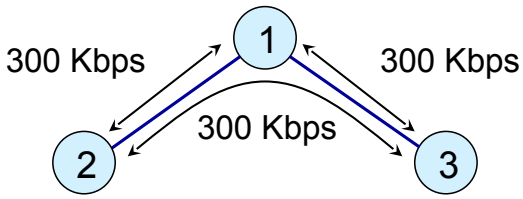
Network Algorithms

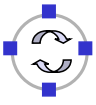
Chapter 2
Modeling Network Design Problems



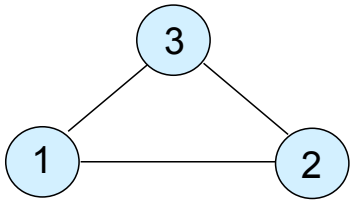
A Simple Network Design Example

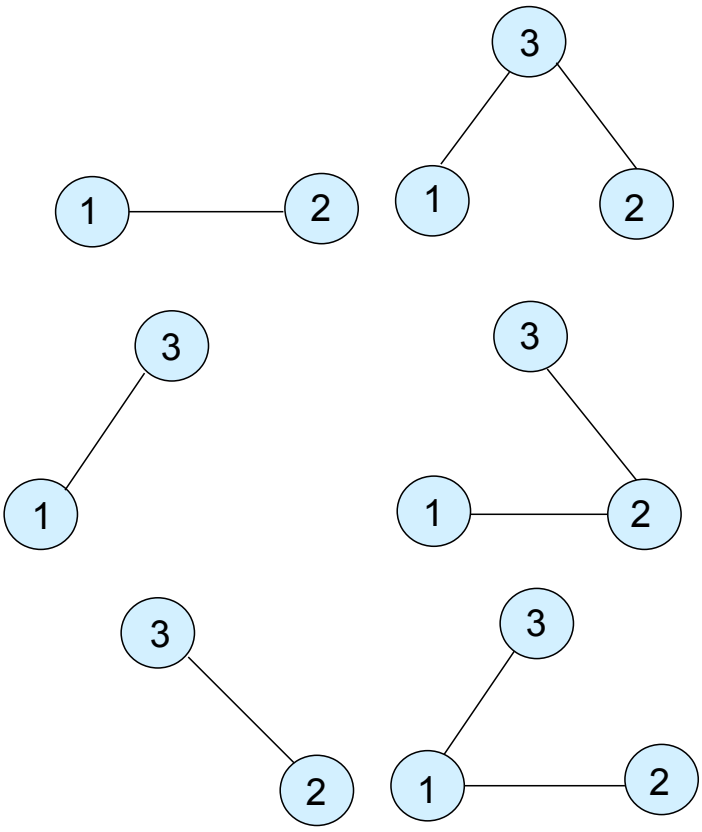


- Consider the example above with three nodes and the demands as specified between the nodes
  - An intuitive option to satisfy the demands could be to install one T1 link (1.54 Mbps) between each node
  - In this case, link utilization would be 300Kbps / 1.54 Mbps ~ 19.5% on each link
  - However, we could also only install two links (1-2 and 1-3) and have traffic between nodes 2 and 3 routed over node 1
  - Link utilization would be ~ 39% in this case, which is still low enough
  - Advantage: this would require only 2/3 of link installation cost



Modeling Design Problems in Link-Path Formulation (1)





- We consider again a simple network, this time with links between each node
- Let us assume that we have undirected **demands** between nodes:
  - 1 and 2: $\hat{h}_{12} = 5$
  - 1 and 3: $\hat{h}_{13} = 7$
  - 2 and 3: $\hat{h}_{23} = 8$
- See on the right all possible paths between nodes

Modeling Design Problems in Link-Path Formulation (2)

- The demand between nodes 1 and 2, denoted as <1, 2> can be routed over two possible paths: 1-2 and 1-3-2
- We use $\hat{x}$ with appropriate indices to denote the unknown **demand path-flow variables** to specify the following constraints:

$$\hat{x}_{12} + \hat{x}_{132} = 5 \quad (= \hat{h}_{12})$$

$$\hat{x}_{13} + \hat{x}_{123} = 7 \quad (= \hat{h}_{13})$$

$$\hat{x}_{23} + \hat{x}_{213} = 8 \quad (= \hat{h}_{23})$$

- Furthermore, all path-flows are non-negative: $\hat{x} \geq 0$ for all paths
- We also need to consider link capacities denoted with $\hat{c}$
- In our example three path-flows make use of the link 1-2:

$$\hat{x}_{12} + \hat{x}_{123} + \hat{x}_{213} \leq \hat{c}_{12}$$

Modeling Design Problems in Link-Path Formulation (3)

- In a similar way, we obtain the following two inequalities:

$$\hat{x}_{132} + \hat{x}_{13} + \hat{x}_{213} \leq \hat{c}_{13}$$

$$\hat{x}_{132} + \hat{x}_{123} + \hat{x}_{23} \leq \hat{c}_{23}$$

- Let us further assume that the capacities of the first two links is 10 and the third link is 15: $\hat{c}_{12} = \hat{c}_{13} = 10, \quad \hat{c}_{23} = 15$

- All in all, we obtain the following set of constraints:

$$\begin{array}{rrrrrrrr}
\hat{x}_{12} & +\hat{x}_{132} & & & & & = & 5 \\
 & & \hat{x}_{13} & +\hat{x}_{123} & & & = & 7 \\
 & & & & \hat{x}_{23} & +\hat{x}_{213} & = & 8 \\
\hat{x}_{12} & & & +\hat{x}_{123} & & +\hat{x}_{213} & \leq & 10 \\
 & \hat{x}_{132} & +\hat{x}_{13} & & & +\hat{x}_{213} & \leq & 10 \\
 & \hat{x}_{132} & & +\hat{x}_{123} & +\hat{x}_{23} & & \leq & 15
\end{array}$$

$$\hat{x}_{12},\ \hat{x}_{132},\ \hat{x}_{13},\ \hat{x}_{123},\ \hat{x}_{23},\ \hat{x}_{213} \geq 0$$

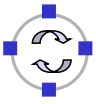


Modeling Design Problems in Link-Path Formulation (4)

- In fact, this system has multiple (continuously many) solutions and defines the set of all **feasible solutions**, i.e. feasible flows $\hat{x}$
- This raises the question, which specific feasible solution is of best interest?
- However, this depends on what is the goal of our network design, e.g.:
  - Minimize the total routing cost (if links are annotated with a link cost)
  - Minimize congestion of the most congested link
- Let us assume, we would like to minimize the total routing cost and the cost of routing one unit of traffic over one link is set to 1 for all links
- This results in the following **objective function**:

$$\mathbf{F} = \hat{x}_{12} + 2\hat{x}_{132} + \hat{x}_{13} + 2\hat{x}_{123} + \hat{x}_{23} + 2\hat{x}_{213}$$

- Note that flows routed over two links are weighted with factor 2



Modeling Design Problems in Link-Path Formulation (5)

- All in all, we have formulated the following optimization problem **(Problem 1)**:

minimize

$$\mathbf{F} = \hat{x}_{12} + 2\hat{x}_{132} + \hat{x}_{13} + 2\hat{x}_{123} + \hat{x}_{23} + 2\hat{x}_{213}$$

subject to

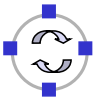
$$\begin{array}{rrrrrrcl}
\hat{x}_{12} & +\hat{x}_{132} & & & & & = & 5 \\
 & & \hat{x}_{13} & +\hat{x}_{123} & & & = & 7 \\
 & & & & \hat{x}_{23} & +\hat{x}_{213} & = & 8 \\
\hat{x}_{12} & & & +\hat{x}_{123} & & +\hat{x}_{213} & \leq & 10 \\
 & \hat{x}_{132} & +\hat{x}_{13} & & & +\hat{x}_{213} & \leq & 10 \\
 & \hat{x}_{132} & & +\hat{x}_{123} & +\hat{x}_{23} & & \leq & 15
\end{array}$$

$$\hat{x}_{12},\ \hat{x}_{132},\ \hat{x}_{13},\ \hat{x}_{123},\ \hat{x}_{23},\ \hat{x}_{213} \geq 0$$



Modeling Design Problems in Link-Path Formulation (6)

- Such an optimization task is called a **multi-commodity flow problem**
- The term multi-commodity comes from the fact that
  - there are multiple demands (or commodities)
  - which need to be routed in the network (to be satisfied) and
  - which compete for available link capacities (resources)
- As all constraints and the optimization function are linear, this is also known as a **linear programming problem**
- The way we formulated the problem is called the **link-path formulation** of the problem
- In our simple example, the optimal solution (marked with a * in superscript) is easy to find, as the cost of multi-link paths is higher than on single link paths and our demands can be satisfied by using only direct links: $\hat{x}_{12}^* = 5, \quad \hat{x}_{13}^* = 7, \quad \hat{x}_{23}^* = 8$
- The total cost of this solution is: $\mathbf{F} = 20$



Modeling Design Problems in Link-Path Formulation (7)

- In most cases, it is not as easy as this to find an optimal solution
- Let us, for example, consider a (rather strange) cost function that has twice the cost for using direct links than for paths over two links (in reality such situations sometimes occur in air travel networks):

$$\mathbf{F} = 2\hat{x}_{12} + \hat{x}_{132} + 2\hat{x}_{13} + \hat{x}_{123} + 2\hat{x}_{23} + \hat{x}_{213}$$

- We are tempted to route all demand over two-link paths as they are cheaper than single-link paths
  - However, this way we would exceed link capacities
  - So, is there no solution to this problem?
- Of course there is, we have only changed the objective function, not the constraints, so the solution to the original problem is still feasible, even though not optimal
- Without explaining how to obtain it (will be done later), the optimal solution (with cost **F** = 25) is:

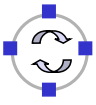
$$\hat{x}^*_{12} = 0, \quad \hat{x}^*_{132} = 5, \quad \hat{x}^*_{13} = 1, \quad \hat{x}^*_{123} = 6, \quad \hat{x}^*_{23} = 4, \quad \hat{x}^*_{213} = 4$$



Modeling Design Problems in Link-Path Formulation (8)

- From this, we can learn two important lessons:
  - Changing the objective function usually affects the optimal solution to a problem, and can have significant impact on the difficulty of finding it (sometimes quite dramatically!)
  - We need to formulate the right goal (objective function) for a particular network – otherwise our optimal solution might not be meaningful

- So far, we have used the notions of links and paths to describe the network optimization problem (thus the name link-path formulation)
- Link-path formulation is appropriate for networks with undirected links as well as with directed links
- Next, we will look at another way to represent network optimization problems



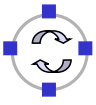
Modeling Design Problems in Node-Link Formulation (1)

- We assume now that links and demands are directed
- Consider a fixed demand pair and a fixed node
- Instead of tracing all path flows realizing the demand, we consider the total link flow for the demand on each link (many of these flows will be zero)
- If we look at this from the point of view of a fixed node that is not an end point of the flow
  - we see flows coming in and going out of that node
  - the total incoming flow is equal to the total outgoing flow (*flow conservation law*)
- If you consider the source node of a demand, the sum of the outgoing flows minus the sum of incoming flows is equal to the demand
- Similar, for a sink node the sum of incoming flows minus the sum of outgoing flows is equal to the demand volume

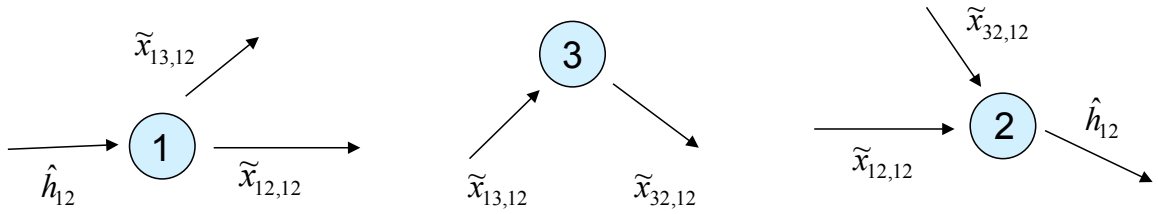


Modeling Design Problems in Node-Link Formulation (2)

- Back to our example network, we substitute each undirected link with two directed links ("arcs"): 1-2 is substituted by 1→2 and 2→1
- Likewise, we need to replace our undirected demands by directed demands:
  - In this example, we do not consider both directions of the demands:
    <1, 2> is replaced by <1:2>
  - For this demand, there are two outlets: 1→2 and 1→3
  - We represent the flows to be allocated on these links with $\tilde{x}_{12,12}$ and $\tilde{x}_{13,12}$ (flow over arc 1→2 for demand <1:2>, and flow over arc 1→3 for demand <1:2>)



Modeling Design Problems in Node-Link Formulation (3)



- Consider demand <1:2>
  - For the source of the flow, node 1, we obtain the following equation:

$$-\hat{h}_{12} - \tilde{x}_{21,12} - \tilde{x}_{31,12} + \tilde{x}_{12,12} + \tilde{x}_{13,12} = 0$$

  - Node 3 is a transit node for the flow:

$$-\tilde{x}_{13,12} - \tilde{x}_{23,12} + \tilde{x}_{31,12} + \tilde{x}_{32,12} = 0$$

  - Node 2 is the sink of the flow:

$$-\tilde{x}_{12,12} - \tilde{x}_{32,12} + \hat{h}_{12} + \tilde{x}_{21,12} + \tilde{x}_{23,12} = 0$$



Modeling Design Problems in Node-Link Formulation (4)

- In our equations we explicitly also included potential “backflows”, e.g. flows $\hat{x}_{21,12}$ and $\hat{x}_{12,12}$ run in opposite directions
- Having such opposite flows would not make much sense from a practical view, so that we can safely set the “back directions” to 0
- Thus, we obtain the following flow conservation equalities for demand <1:2>:

$$\begin{array}{rrrcl} \tilde{x}_{12,12} & +\tilde{x}_{13,12} & & = & \hat{h}_{12} \\ & -\tilde{x}_{13,12} & +\tilde{x}_{32,12} & = & 0 \\ -\tilde{x}_{12,12} & & -\tilde{x}_{32,12} & = & -\hat{h}_{12} \end{array}$$

- This system of equations is dependent and one equation can be eliminated (this is a general fact)
- For demand <1:3> we obtain the following equations:

$$\begin{array}{rrrcl} \tilde{x}_{12,13} & +\tilde{x}_{13,13} & & = & \hat{h}_{13} \\ -\tilde{x}_{12,13} & & +\tilde{x}_{23,13} & = & 0 \\ & -\tilde{x}_{13,13} & -\tilde{x}_{23,13} & = & -\hat{h}_{13} \end{array}$$



Modeling Design Problems in Node-Link Formulation (5)

- For demand <2:3> we obtain the following equations:

$$\begin{array}{rrrcl} \tilde{x}_{21,23} & & +\tilde{x}_{23,23} & = & \hat{h}_{23} \\ -\tilde{x}_{21,23} & +\tilde{x}_{13,23} & & = & 0 \\ & -\tilde{x}_{13,23} & -\tilde{x}_{23,23} & = & -\hat{h}_{23} \end{array}$$

- Furthermore, we have to model capacity constraints of our links, e.g. all flows on link 1→2 should not exceed its capacity:

$$\tilde{x}_{12,12} + \tilde{x}_{12,13} \leq \hat{c}_{12}$$

- We can write down similar equations for all other capacity constraints
- If we want to minimize the overall routing cost for realizing the flows with cost of routing one unit of flow over one link set to 1 (like in our first link-path formulation), we use the following objective function:

$$\mathbf{F} = \tilde{x}_{12,12} + \tilde{x}_{13,12} + \tilde{x}_{32,12} + \tilde{x}_{12,13} + \tilde{x}_{13,13} + \tilde{x}_{23,13} + \tilde{x}_{21,23} + \tilde{x}_{13,23} + \tilde{x}_{23,23}$$



Modeling Design Problems in Node-Link Formulation (6)

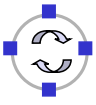
minimize **(Problem 2)**

$$\mathbf{F} = \tilde{x}_{12,12} + \tilde{x}_{13,12} + \tilde{x}_{32,12} + \tilde{x}_{12,13} + \tilde{x}_{13,13} + \tilde{x}_{23,13} + \tilde{x}_{21,23} + \tilde{x}_{13,23} + \tilde{x}_{23,23}$$

subject to

$$\begin{array}{rrrrrrrrrcl} \tilde{x}_{12,12} & +\tilde{x}_{13,12} & & & & & & & & = & \tilde{h}_{12} \\ & -\tilde{x}_{13,12} & +\tilde{x}_{32,12} & & & & & & & = & 0 \\ -\tilde{x}_{12,12} & & -\tilde{x}_{32,12} & & & & & & & = & -\tilde{h}_{12} \\ & & & \tilde{x}_{12,13} & +\tilde{x}_{13,13} & & & & & = & \tilde{h}_{13} \\ & & & -\tilde{x}_{12,13} & & +\tilde{x}_{23,13} & & & & = & 0 \\ & & & & -\tilde{x}_{13,13} & -\tilde{x}_{23,13} & & & & = & -\tilde{h}_{13} \\ & & & & & & \tilde{x}_{21,23} & & +\tilde{x}_{23,23} & = & \tilde{h}_{23} \\ & & & & & & -\tilde{x}_{21,23} & +\tilde{x}_{13,23} & & = & 0 \\ & & & & & & & -\tilde{x}_{13,23} & -\tilde{x}_{23,23} & = & -\tilde{h}_{23} \\ \tilde{x}_{12,12} & & & +\tilde{x}_{12,13} & & & & & & \leq & \tilde{c}_{12} \\ & & & & & & \tilde{x}_{21,23} & & & \leq & \tilde{c}_{21} \\ & \tilde{x}_{13,12} & & & +\tilde{x}_{13,13} & & & +\tilde{x}_{13,23} & & \leq & \tilde{c}_{13} \\ & & & & & \tilde{x}_{23,13} & & & +\tilde{x}_{23,23} & \leq & \tilde{c}_{23} \\ & & \tilde{x}_{32,12} & & & & & & & \leq & \tilde{c}_{32} \end{array}$$

all $\tilde{x}$ non-negative



Reflection on Notations

- So far, we have used a certain notation to represent nodes, links, demands and path flows from a *node-identifier based perspective*
- While this works well for small examples, in larger networks this may become cumbersome:
  - Between some pairs of nodes (i, j) there might be no demand: $\hat{h}_{ij} = 0$
  - Between most pairs of nodes (i, j) there will be no links: $\hat{c}_{ij} = 0$
  - In a node-identifier based notation, all such cases have to be explicitly included in the problem formulation
- Furthermore, the notation of candidate paths is rather clumsy:
  - All nodes in a candidate path are included in the variables indices
  - This allows for no clean notation of candidate paths in case that candidate paths with different lengths have to be supported
  - The notation can not handle multiple links between a given pair of nodes (multi-graph case)
  - Likewise handling multiple demands between a pair of nodes is impossible
  - There is virtually no compact way of writing summations over paths



Link-Demand-Path-Identifier-Based Notation (1)

- In order to avoid these problems, we develop yet another notation:
  - Each non-zero demand <i, j> (undirected) or <i:j> (directed) is explicitly assigned a demand label index (assignments are stored in a table)

$$\hat{h}_{12} \longleftrightarrow h_1, \quad \hat{h}_{13} \longleftrightarrow h_2, \quad \hat{h}_{23} \longleftrightarrow h_3$$

  - The same is done with all existing links (v, w)

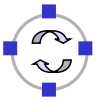
$$\hat{c}_{12} \longleftrightarrow c_1, \quad \hat{c}_{13} \longleftrightarrow c_2, \quad \hat{c}_{23} \longleftrightarrow c_3$$

  - Candidate paths for each demand d are numbered from 1 to P_d with P_d being the total number of candidate paths for demand d
    - E.g., for the demand h_1 = <1, 2> we have a total number of P_1 = 2 candidate paths
    - The respective path flow variables are indexed by composed indices consisting of the demand identifier and the candidate path identifier

$$\hat{x}_{12} \longleftrightarrow x_{11}, \quad \hat{x}_{132} \longleftrightarrow x_{12}$$

$$\hat{x}_{13} \longleftrightarrow x_{21}, \quad \hat{x}_{123} \longleftrightarrow x_{22}$$

$$\hat{x}_{23} \longleftrightarrow x_{31}, \quad \hat{x}_{213} \longleftrightarrow x_{32}$$



Link-Demand-Path-Identifier-Based Notation (2)

- Thus, demand and capacity labels each have a single index
- This notation allows us to rewrite our original problem to **(Problem 3)**:

$$\textbf{minimize}$$

$$F = x_{11} + 2x_{12} + x_{21} + 2x_{22} + x_{31} + 2x_{32}$$

$$\textbf{subject to}$$

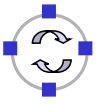
$$\begin{array}{cccccccc}
x_{11} & +x_{12} & & & & & = & h_1 \\
 & & x_{21} & +x_{22} & & & = & h_2 \\
 & & & & x_{31} & +x_{32} & = & h_3 \\
x_{11} & & & +x_{22} & & +x_{32} & \leq & c_1 \\
 & x_{12} & +x_{21} & & & +x_{32} & \leq & c_2 \\
 & x_{12} & & +x_{22} & +x_{31} & & \leq & c_3
\end{array}$$

$$x_{11}, x_{12}, x_{21}, x_{22}, x_{31}, x_{32} \geq 0$$



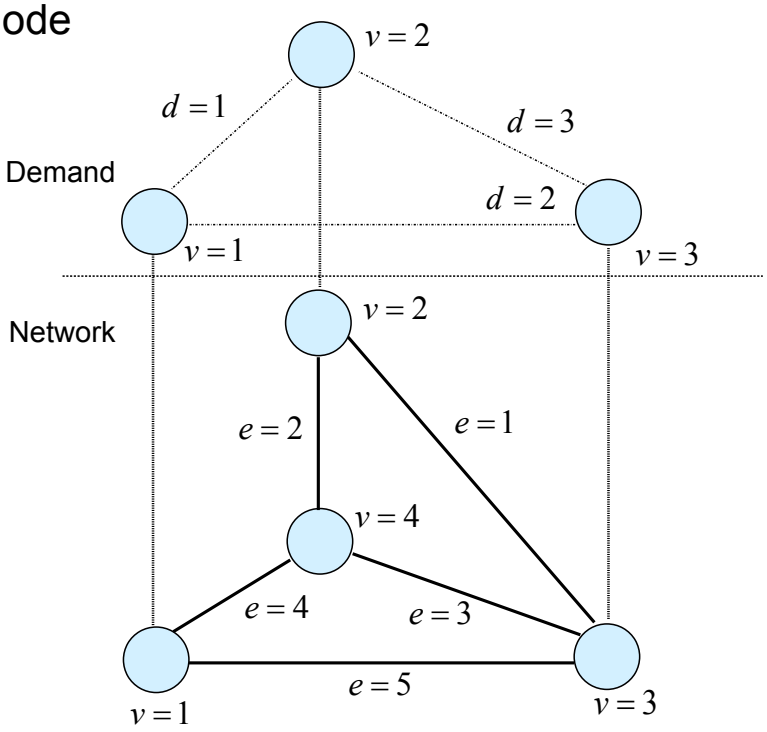
Link-Demand-Path-Identifier-Based Notation (3)

- Note, that this formulation still represents the original problem (Problem 1) and both are given in link-path representation. The only difference lies in the simplified indexing of the variables and constants.
- More definitions on notation:
  - An undirected demand between nodes v and w is denoted as <v,w>
  - A directed demand between nodes v and w is denoted as <v:w>
  - An undirected link between nodes v and w is denoted as v-w
  - A directed link between nodes v and w is denoted as v→w
  - An n-hop path between two nodes v_1 and v_{n+1} is an interlacing sequence of nodes and links (v_1, e_1, v_2, e_2, …, v_n, e_n, v_{n+1})
  - In "node representation" we write undirected paths as v_1-v_2-…-v_{n+1} and directed paths as v_1→v_2→…→v_{n+1}
  - In "link representation" we write undirected paths as {e_1, e_2, …, e_n} and directed paths as (e_1, e_2, …, e_n)
  - We denote nodes with v, links with e, demands with d and paths with p
  - The total numbers of nodes, links, demands, paths is given by V, E, D, P



Dimensioning Problems (1)

- Consider the following example of a network with four nodes, in which there exist demands between three nodes and one node (node 4) is a pure transit node

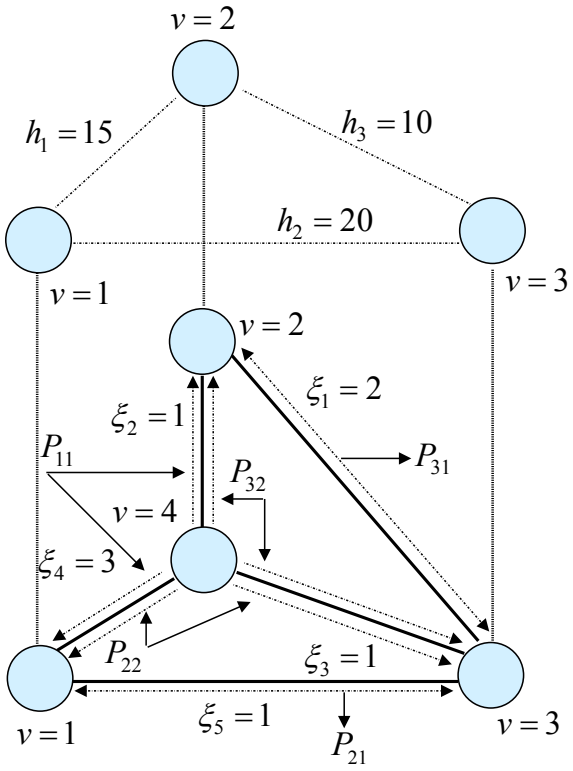


Dimensioning Problems (2)

- The figure on the preceding slide shows in the upper part the **demand view** of the dimensioning problem and in the lower part the **network view**:
  - Demand 1 is between nodes 1 and 2 etc.
  - Link 1 is between nodes 2 and 3
  - Both links and demands are undirected (no arrows)
  - As node 4 has no demand (neither as source nor as sink), it is not shown in the upper part of the figure
- Concerning link capacities:
  - When a capacity for a link e is given, we usually represent it by constant c_e
  - However, in a dimensioning problem capacities are unknown and will therefore be represented by variables y_e
  - Demand and link capacities are represented in generic **demand volume units (DVU)** or **link capacity units (LCU)**, respectively
  - You can think of DVU and LCU in terms of pps or Mbps, but it really does not matter as long as all units are equal

Dimensioning Problems (3)

- The refined view on the left adds further details to our problem:
  - Volumes of individual demands h_i are specified
  - Costs ξ_i of links e_i are given
  - Potential candidate paths for demands P_{ij} are specified
  - Note that demand 1 has only one candidate path P_{11} even if others would be possible
  - Demands 2 and 3 have 2 candidate paths each (P_{21}, P_{22} and P_{31}, P_{32})
  - The flow on path P_{ij} is denoted by the path flow variable x_{ij}



Dimensioning Problems (4)

- This allows us to write the following equations for path flow variables:

$$x_{11} = 15$$

$$x_{21} + x_{22} = 20$$

$$x_{31} + x_{32} = 10$$

- If we denote the number of candidate paths for demand d with P_d, we can – due to our link-demand-path-identifier based notation – actually write these equations in a more concise and uniform way as:

$$\sum_{p=1}^{P_d} x_{dp} = h_d, \quad d = 1, \ldots, D$$

- When it is clear that we sum over all candidate paths for demands:

$$\sum_{p} x_{dp} = h_d, \quad d = 1, \ldots, D$$

Dimensioning Problems (5)

- We call the vector of all flows (path flow variables) the **flow allocation vector** or short **flow vector**:

$$\begin{aligned}\mathbf{x} &= (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_D) \\ &= (x_{11}, x_{12}, \ldots, x_{1P_1}, \ldots, x_{D1}, x_{D2}, \ldots, x_{DP_D}) \\ &= (x_{dp} : d = 1, 2, \ldots, D; p = 1, 2, \ldots, P_d)\end{aligned}$$

- In this notation, we represent vectors with bold letters **x** and scalar components of vectors with normal letters x
- The only exception of this rule is the value of the objective function **F** which is a scalar, but nevertheless represented with a bold letter
- We need a second set of constraints for modeling that for each link e its capacity c_e or y_e is not exceeded
  - Recall, if the capacities are known, they are specified as constants c_e
  - If capacities are unknown (dimensioning problem) we write them as variables y_e



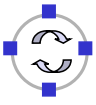
Dimensioning Problems (6)

- For our four node network example, we obtain the following **capacity constraints**:

$$\begin{array}{rrrrcl} & & x_{31} & & \leq & y_1 \\ x_{11} & & & +x_{32} & \leq & y_2 \\ & x_{22} & & +x_{32} & \leq & y_3 \\ x_{11} & +x_{22} & & & \leq & y_4 \\ & x_{21} & & & \leq & y_5 \end{array}$$

- In order to write down the capacity constraints we need to know the relationship between links and paths (link-path incidence relation δ_{edp})

[gray].9 e \ P_{dp}	$P_{11} = \{2,4\}$	$P_{21} = \{5\}$	$P_{22} = \{3,4\}$	$P_{31} = \{1\}$	$P_{32} = \{2,3\}$
1	0	0	0	1	0
2	1	0	0	0	1
3	0	0	1	0	1
4	1	0	1	0	0
5	0	1	0	0	0



Dimensioning Problems (7)

- The table for the link-path incidence relation δ_{edp} contains a 1 whenever a link e is used for satisfying a demand d over a path p, and 0 when the link is not used in that path
- Thus, it is defined as:

$$\delta_{edp} = \begin{cases} 1 & \text{if link } e \text{ belongs to path } p \text{ for demand } d \\ 0 & \text{otherwise} \end{cases}$$

- Note, that δ_{edp} is not a variable, as it is already fixed which paths are to be considered as candidate paths for demands d
- This notation allows us to write the load $\underline{y}_e$ on link e in a compact way:

$$\underline{y}_e = \underline{y}_e(\mathbf{x}) = \sum_{d=1}^{D} \sum_{p=1}^{P_d} \delta_{edp} x_{dp}$$

- With this, we can specify our capacity constraints as:

$$\sum_d \sum_p \delta_{edp} x_{dp} \leq y_e, \quad e = 1, 2, \ldots, E$$



Dimensioning Problems (8)

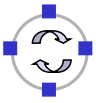
- Please note the difference of the actual link loads $\underline{y}_e$ (determined by path flow variables x_{dp} of a solution) and the capacity variables y_e (computed as part of the solution)
- If we denote the cost of a link e by ξ_e and we are interested in **minimizing the capacity cost**, we obtain the following objective function:

$$\mathbf{F} = \sum_e \xi_e y_e \quad (= 2y_1 + y_2 + y_3 + 3y_4 + y_5)$$

- The general formulation of our simple dimensioning problem is:

minimize
objective/cost function: $F = \sum_e \xi_e y_e$
subject to

$$\begin{array}{rcl} \text{demand constraints:} & \sum_p x_{dp} = h_d, & d = 1, 2, \ldots, D \\ \text{capacity constraints:} & \sum_d \sum_p \delta_{edp} x_{dp} \leq y_e, & e = 1, 2, \ldots, E \\ \text{constraints on variables:} & \mathbf{x} \geq 0, \mathbf{y} \geq 0 & \end{array}$$



Dimensioning Problems (9)

- All in all, we obtain the following problem formulation for our example **(Problem 4)**:

$$\textbf{minimize}$$
$$F = 2y_1 + y_2 + y_3 + 3y_4 + y_5$$
$$\textbf{subject to}$$

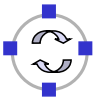
$$\begin{aligned}
x_{11} &= 15 \\
x_{21} + x_{22} &= 20 \\
x_{31} + x_{32} &= 10 \\
x_{31} &\le y_1 \\
x_{11} + x_{32} &\le y_2 \\
x_{22} + x_{32} &\le y_3 \\
x_{11} + x_{22} &\le y_4 \\
x_{21} &\le y_5
\end{aligned}$$

$$x_{11}, x_{21}, x_{22}, x_{31}, x_{32} \ge 0, \quad y_1, y_2, y_3, y_4, y_5 \ge 0$$



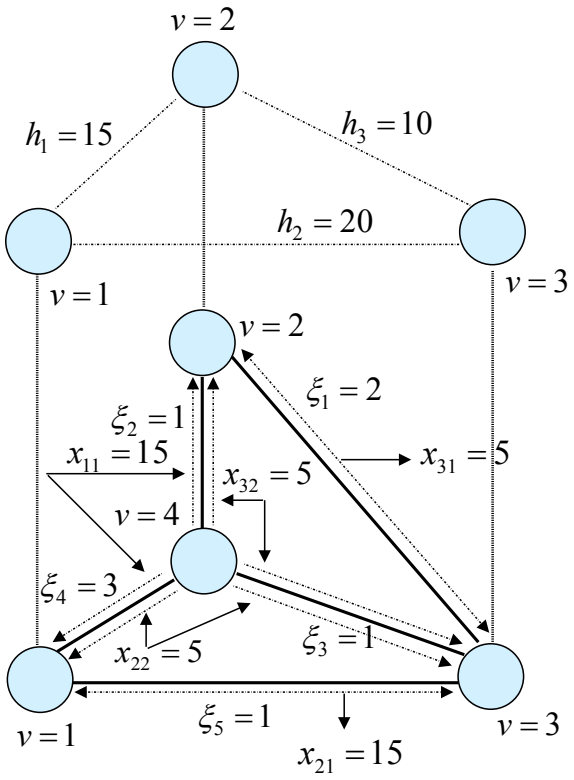
Dimensioning Problems (10)

- Even though in linear programming all variables are usually written on the left-hand side of equations or constraints, we keep the y_i on the right hand side for clarity of presentation
- Let us compare Problem 4 to our first example Problem 1:
  - In Problem 1, we minimized the total routing cost in a network for which link capacities were given to carry a given demand using flow variables
  - In Problem 4, we minimized the total link capacity cost under the constraint that the network needs to meet a given traffic demand. Link capacities were not given, but variables for which optimal values were to be computed
  - Problems that have variable link capacities are referred to as **dimensioning problems** or **uncapacitated problems**
- When variables can take continuous values, then for any optimal solution the capacity constraints become equalities
  - For each link, its load is equal to its capacity because otherwise we would have to pay for unused capacity (implying the solution is not optimal)



Dimensioning Problems (11)

- This figure shows a set of values for the flow variables x_{ij}
- If we calculate the link loads for this flow allocation vector, we obtain:
  $y_1 = 5,\ y_2 = 20\ \ y_3 = 10,\ y_4 = 20,\ y_5 = 15$
- The total cost for this feasible solution according to our cost function is:
  $F = 2 \cdot 5 + 20 + 10 + 3 \cdot 20 + 15 = 115$
- It is easy to see that this is not an optimal solution, as it uses the rather expensive path P_{22} (cost 4) for parts of demand h_2 which could also completely be routed over the cheaper path P_{21} (cost 1)



Dimensioning Problems (12)

- In general, the cost of a path P_{dp} is given by:

$$\zeta_{dp} = \sum_e \delta_{edp} \xi_e, \quad d = 1, 2, \ldots, D \quad p = 1, 2, \ldots, P_d$$

- In our example, it is profitable to move all the flow from path P_{22} to path P_{21} which results in a saving of $x_{22} \cdot (\zeta_{22} - \zeta_{21}) = 5 \cdot 3 = 15$
- Note that path flow x_{11} is trivially optimal as there is only one (considered) candidate path P_{11} for demand h_1
- Also, the two flows x_{31} and x_{32} are optimal as they both have the same cost $\zeta_{31} = \zeta_{32} = 2$
- Thus for demand 3 any split of its volume $h_3 = 10$ among the two available paths leads to an optimal solution
- These considerations lead to describing the optimal solution (x*, y*) in the following way (see next slide)

Dimensioning Problems (13)

$$x_{11}^* = 15$$
$$x_{21}^* = 20, \quad x_{22}^* = 0$$
$$x_{31}^* = a, \quad x_{32}^* = 10 - a \quad \text{for any } 0 \le a \le 10$$
$$y_1^* = 10 - a, \quad y_2^* = 15 + a, \quad y_3^* = a, \quad y_4^* = 15, \quad y_5^* = 20$$
$$\mathbf{F}^* = 100.$$

- Thus, we can make the following statement in regard to obtaining the optimal solution to problems like Problem 4:

Shortest-Path Allocation Rule for Dimensioning Problems:

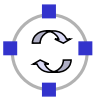
For each demand, allocate its entire demand to its shortest path with respect to link costs and candidate paths.

If there is more than one shortest path for a given demand, then the demand volume can be arbitrarily split among the shortest paths.



Dimensioning Problems (14)

- Note that the optimal solution to Problem 4 is not unique, but there are many solutions with the same optimal cost (F = 100)
- While the shortest path rule works for the simple dimensioning problem considered so far, this must not necessarily be the case if further constraints are to be taken into account
- The list of considered candidate paths has a determining influence on the optimal solution:
  - Let us assume that we also have $P_{12} = (e_1, e_5)$ with overall path cost $\xi_{e5} + \xi_{e5} = 3$ as a candidate path for demand 1
  - In this case path P_{12} is 1 unit cheaper than P_{11} and we can save 15 units of cost by routing all traffic of demand 1 over path P_{12}
  - Starting with a “good” set of candidate paths is thus essential for obtaining good solutions, so that “path preprocessing”, i.e. initializing and augmenting path lists is an important part of the network design process



Dimensioning Problems (15)

- There are cases, in which further constraints are imposed on the problem:
  - One example are so called **non-bifurcated flows**, in which each demand has to be satisfied by exactly one path flow
  - We could as well imagine cases where demands have to be partitioned among several (node-disjoint) paths in order to allow for graceful degradation in case of node or link failures
  - When link capacities are given we speak of **capacitated design problems**
  - Depending on demands and capacities it may happen that non-bifurcated solutions are not possible even though bifurcated solutions exist

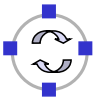


Capacitated Problems

- In some cases link capacities are given, and we are looking for a solution that satisfies the specified demands while staying within capacity bounds
- Such problems can be formulated in the following general notation (**Problem 5**):

$$\begin{aligned} \text{demand constraints:} \quad & \textstyle\sum_p x_{dp} = h_d, && d = 1, 2, \ldots, D \\ \text{capacity constraints:} \quad & \textstyle\sum_d \sum_p \delta_{edp} x_{dp} \leq c_e, && e = 1, 2, \ldots, E \\ \text{constraints on variables:} \quad & x \geq 0. \end{aligned}$$

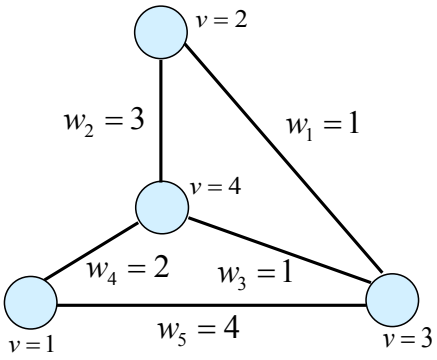
- Sometimes there might even be no objective function, so that any solution that satisfies the constraints is acceptable
- If flow routing cost is to be minimized, these problems are similar to problem 1



Shortest-Path Routing (1)

- Shortest-path routing in network design means, that we have to anticipate that demands will only be routed on their shortest paths
- The path "length" is determined by adding up link costs w_e according to some weight system $\mathbf{w} = (w_1, w_2, \ldots, w_E)$
- Let us assume the weight system (1, 3, 1, 2, 4) for our example network
  - With shortest-path routing the entire demands would be routed over their respective shortest path:

$$\begin{array}{llll} d=1 & \mathcal{P}_{13}=\{1,3,4\} & \text{path length } 4 & x_{13}=15 \\ d=2 & \mathcal{P}_{21}=\{1\} & \text{path length } 1 & x_{21}=10 \\ d=3 & \mathcal{P}_{32}=\{3,4\} & \text{path length } 3 & x_{32}=10 \end{array}$$



  - However, if we assume capacities **c** = (5, 10, 10, 5, 30) this flow vector **x(w)** is not feasible, as link load would be **y(w)** = (25, 0, 35, 35, 0)

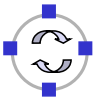


Shortest-Path Routing (2)

- Clearly, if we change the link capacities to be equal to the computed link loads of the shortest-path routing solution, this solution becomes (trivially) feasible
- In general, however, we are interested in a solution that allows to respect the given link capacities and instead looks for the appropriate weight system

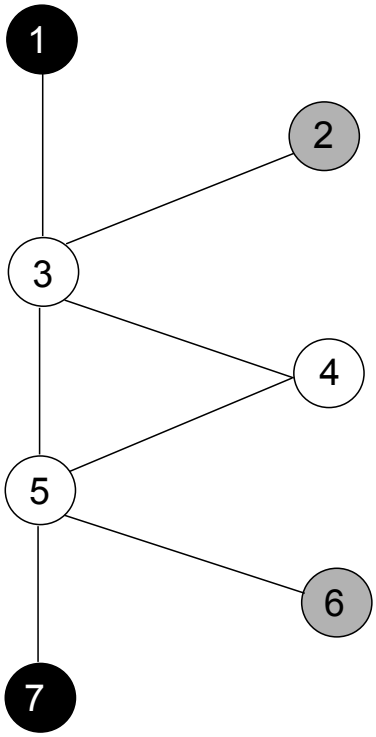
Single Shortest Path Allocation Problem
For given link capacities **c** and demand volumes **h**, find a link weight system **w** such that the resulting shortest paths are unique and the resulting flow allocation vector is feasible

- This usually is a complex problem:
  - A non-bifurcated solution may not exist even when bifurcated solutions do
  - Non-bifurcated solutions are usually hard to determine (if they exist)
  - Even when a single-path flow solution exists, it might be impossible to find a weight system to induce this solution



Shortest-Path Routing (3)

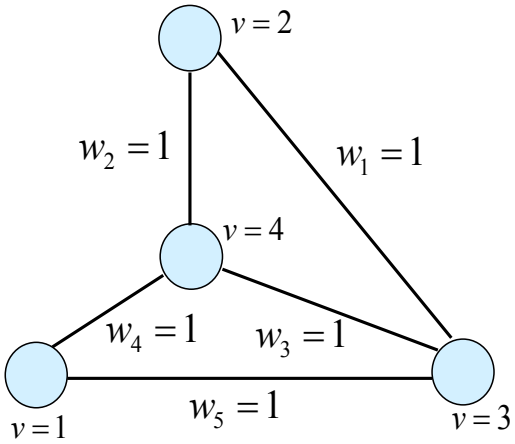
- Consider the following network:
  - Demands d_1 between nodes 1 and 7, and demand d_2 between nodes 2 and 6 both have volume $h_1 = h_2 = 1$
  - All link capacities $c_i = 1$
  - Two candidate paths for d_1: $P_{11} = 1 \rightarrow 3 \rightarrow 5 \rightarrow 7$ and $P_{12} = 1 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 7$
  - Two candidate paths for d_2: $P_{21} = 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$ and $P_{22} = 2 \rightarrow 3 \rightarrow 5 \rightarrow 6$
  - We are looking for a non-bifurcated (single-path) solution to Problem 5 (link capacities given, no objective function)
  - $x_{11} = 1$ and $x_{21} = 1$ is a valid solution
  - However, trying to find a weight system to induce this solution is not obvious (actually, it is not possible for single shortest paths)

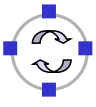




Shortest-Path Routing (4)

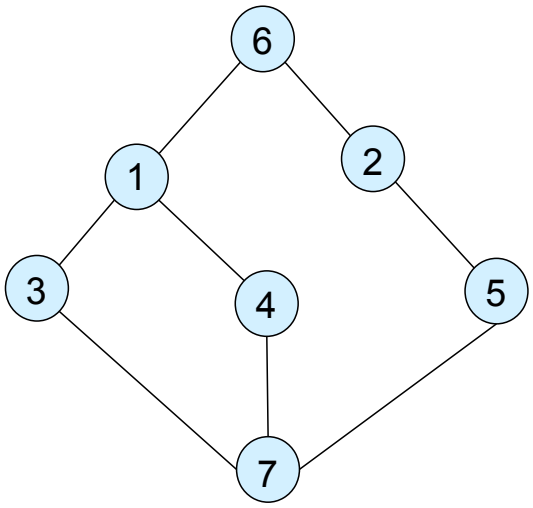
- Let us go back to our four node network and set all link weights to 1 (like in standard Internet routing)
- Consider demand d_1 between nodes 2 and 1: there are two shortest paths $P_{11} = \{2, 4\}$ and $P_{12} = \{1, 5\}$
- This raises the question over which path the demand should be routed
  - Often, a rule for splitting demand volumes among multiple shortest paths is desired
  - One such rule, used in OSPF routing, is the **"equal-cost multi-path"** rule (**ECMP**): for a fixed destination the goal of ECMP is to equally split the outgoing demand volume over all outgoing next hops with equal cost





Shortest-Path Routing (5)

- However, this simple rule may fail if link weights are not set appropriately
- Assume that in the network on the left all link weights $w_i = 1$
  $\Rightarrow$ there are 3 shortest paths from 6 to 7
- For the directed demand <6:7> paths 6→1→3→7 and 6→1→4→7 each carry 25% and path 6→2→5→7 carries 50% of the traffic volume if ECMP is used
- As in OSPF routing is directional, the directed demand <7:6> is equally split to the three paths in opposite direction

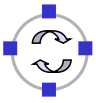


- Determining weight systems under the ECMP rule encounters similar difficulties like when determining weight systems for the single-shortest-path allocation case (~non-bifurcated)

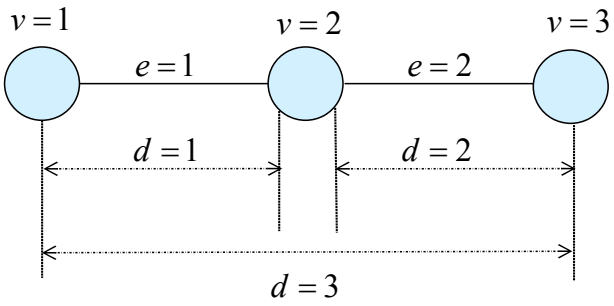


Fair Networks (1)

- In the Internet, traffic demands are often not fixed but **elastic**, that is each demand can consume any bandwidth assigned to its path
- In such cases, we have capacity constraints as in Problem 5, but no particular values h_d for demands are assumed
- In order to deal with potential capacity bottlenecks, we are interested in assigning demand volumes such that fairness is assured among the different demands
  - As an obvious initial solution, we could assign each demand volume on its lower bound
  - If this does not satisfy capacity constraints, then there is no feasible solution at all
  - However, if feasibility is assured, we would like to carry more than the minimum required bandwidth of demands while at the same time giving a **fair share** of bandwidth to all flows
- Furthermore, we want to understand implications of our assignments on network throughput (= sum of path flows x_{dp})



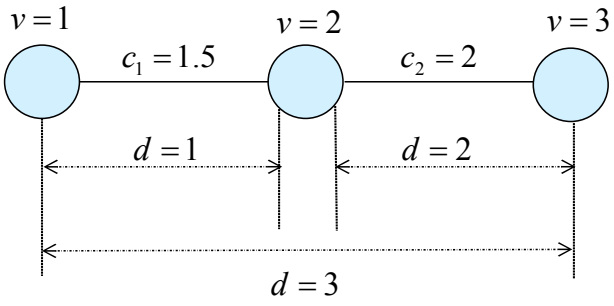
Fair Networks (2)



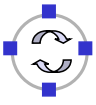
- Consider the above example network with three nodes, three demands, only one candidate path per demand, and both link capacities being $c_1 = c_2 = 1.5$
- The best-known general fairness criterion is **Max-Min Fairness (MMF)**, also called **equity**:
  - If no lower bounds are specified, assign the same maximal value to all demands
  - If there is still capacity left, assign the same maximal value to all demands that can still make use of that capacity
- Under MMF, we obtain $x_{11} = x_{21} = x_{31} = 0.75$ ($\Rightarrow$ all capacity used)



Fair Networks (3)



- Let us assume now, that c_2 has been increased to 2:
  - After the initial MMF allocation link 2 still has 0.5 LCU available
  - As only demand 2 can make use of this unused capacity, we can increase h_2 to 1.25
  - Fair from the user's point of view (nobody else can use this capacity)
- Let us look at network throughput:
  - The first MMF solution leads to throughput $x_{11} + x_{21} + x_{31} = 2.25$
  - The second MMF solution (with $c_2 = 2$) leads to throughput 2.75
  - A rather unfair solution (for $c_2 = 1.5$) would be $x_{11} = 1.5$, $x_{21} = 1.5$, $x_{31} = 0$, leading to throughput 3.0



Fair Networks (4)

- This last unfair solution would be obtained if we used the objective function $\mathbf{F} = \sum_d \sum_p x_{dp}, \quad d = 1, 2, \ldots, D \quad p = 1, 2, \ldots, P_d$
- Clearly, setting all path flow variables of one flow to 0 is not desirable if we aim to assure some kind of fairness in the network
  - However, if a demand is routed over a longer path, it uses more link resources than demands that are routed over shorter paths
- We would like to find a compromise between MMF and greedily maximizing network throughput:
  - One such fair allocation principle is called Proportional Fairness (PF) and it is realized by maximizing the following logarithmic revenue function

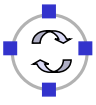
    $\mathbf{R}(\mathbf{x}) = \sum_d r_d \, ln\left(\sum_p x_{dp}\right), \quad d = 1, 2, \ldots, D \quad p = 1, 2, \ldots, P_d$

    with r_d being the revenue associated with demand d
    (e.g. r_i = 1 if all demands are of equal importance)
  - Note, that this is not a linear function
  - This function ensures that no demand is allocated an overall path flow sum of 0 and makes assigning (unfairly) high values “unattractive”



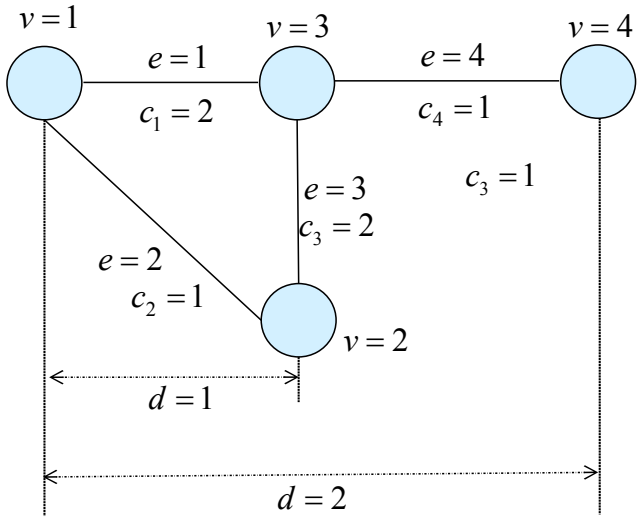
Fair Networks (5)

- The above PF problem can be solved by introducing a linear approximation of the logarithmic function and solving the resulting linear programming problem (explained later)
- Solving the MMF capacitated problem is more complicated:
  - In general, it is not enough to find a flow allocation vector that maximizes the minimal flow X_d over all demands d
  - If such a flow vector X is found then in general some link capacities might still be free and can be used to increase flow allocations for at least a subset of demands
  - However, depending on the initial flow allocation vector, we may be able to extend some flows or not (leading to different overall network throughput)



Fair Networks (6)

- Applying solution x^1 leaves room for further increase in flow for demand 1
  - In this case $x^1_{12} = 1$ can be additionally routed over path {1, 3}
- Applying solution x^2 does not leave room for further improvement:
  - Link 3 is exhausted, routing demand 2 over this link was not a good decision
  - Remaining capacity of link 1 can not be used
- The final total flow allocation vector is $X = (X_1, X_2) = (2, 1)$



$$Solution\ x^1 : x^1_{11} = 1\ on\ path\ \{2\}, x^1_{21} = 1\ on\ path\ \{1,4\}$$

$$Solution\ x^2 : x^2_{11} = 1\ on\ path\ \{1,3\}, x^2_{21} = 1\ on\ path\ \{2,3,4\}$$



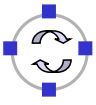
Topological Design (1)

- When installing a link in a network, usually a fixed cost has to be invested (e.g. cabling cost) independent of the capacity of the link
- In order to account for this, we have to model this "opening cost" κ_e in the objective function (which is to be minimized):

$$\mathbf{F} = \sum_e \xi_e y_e + \sum_e \kappa_e u_e$$

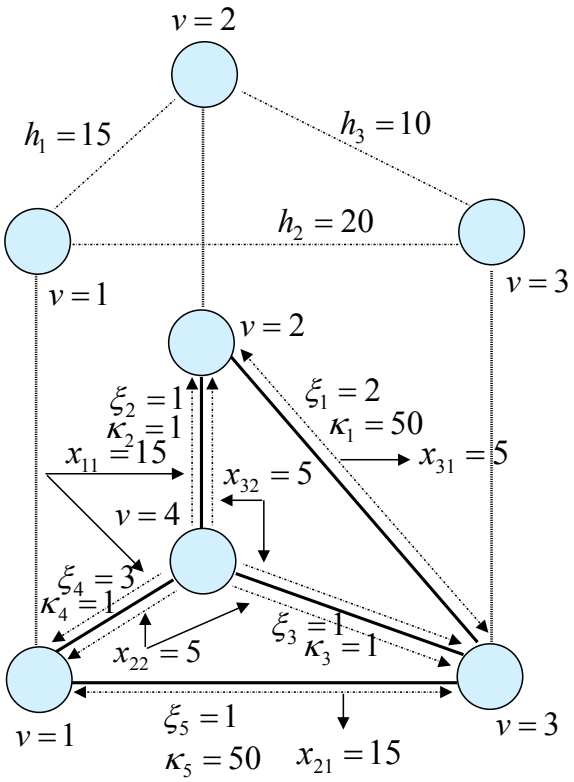
  where the binary variable u_e is 1 if link e is installed and 0 if not
- In order to force the capacity y_e of link e to be 0 whenever the link e is not installed, we introduce the following additional constraints:

$$y_e \leq \Delta u_e \quad e = 1, \ldots, E \quad \text{and } \Delta \text{ being a large constant}$$



Topological Design (2)

- Assume, that the installation costs κ are (50, 1, 1, 1, 50)
- Then the optimal network topology becomes a tree consisting of links {2, 3, 4} with link installation costs 3 and capacity dependent costs summing up to 160
- The difficulty of solving such problems is considerable and comparable to the case of uncapacitated problems with modular links (= links of various fixed capacity values)
- A variation of such problems considers node locations



Restoration Design (1)

- So far, we have considered the network to be in normal operational state, that is no link or node failure has been taken into account
- Let us now assume the following failure model:
  - Links can be either fully functional or completely failed
  - No more than one link fails at a time
  - For our example network consisting of four nodes and five links, we can therefore model the following 6 **failure states** or **failure situations:**
  - In failure state 0 no link, in failure state s link s has failed ($1 \leq s \leq 5$)
- Suppose, we want to solve the restoration design problem (RDP) for our four network example (with two candidate paths for demand 1), so that all demands can be fully satisfied in all failure states:
  - In order to account for failure situations, we introduce an additional index s to our path flow variables x_{dps} referring now to that particular flow in case of failure state s
  - E.g. for demand 1 we obtain: $x_{11s} + x_{12s} = 15 \quad s = 0, \ldots, S$ (note that due to our assumptions S = E in this example)

Restoration Design (2)

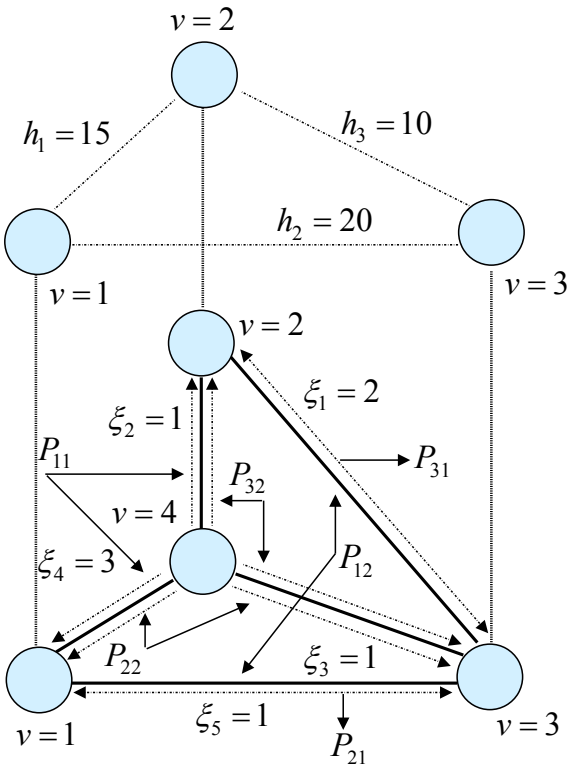
- Furthermore, we have to reformulate our capacity constraints:

$$\begin{aligned} s=0: &\quad x_{120}+x_{310} &&\le y_1 \\ s=1: &\quad x_{121}+x_{311} &&\le 0 \\ s=2: &\quad x_{122}+x_{312} &&\le y_1 \\ &\ldots \end{aligned}$$

- We introduce notation α_{es} to denote if link e is up (1) or not (0) and obtain the following:

$$\sum_d \sum_p \delta_{edp} x_{dps} \le \alpha_{es} y_e$$

$$s = 0,1,\ldots,S \quad e = 1,2,\ldots,E$$



Restoration Design (3)

- If we can fully re-arrange flows, we obtain the following optimal flows:

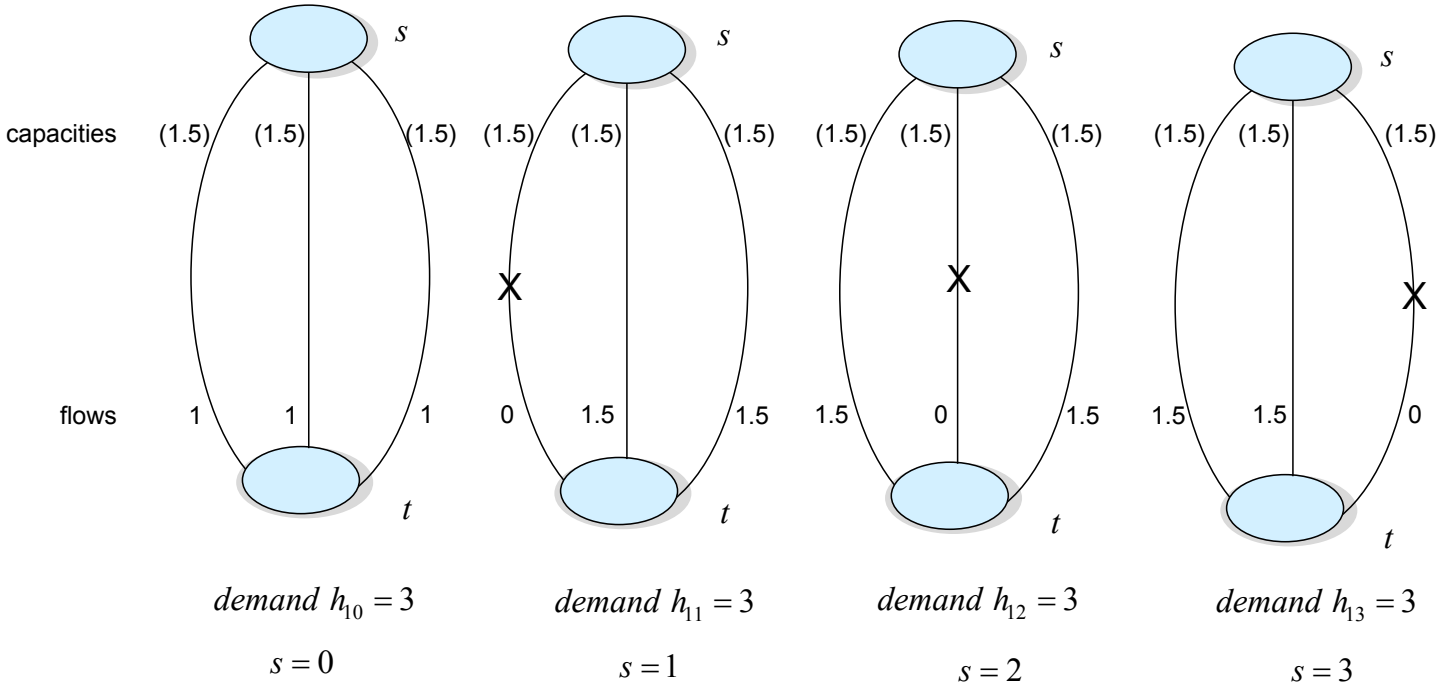
$$\begin{aligned} s=0: &\quad x^*_{110}=0 & x^*_{120}=15 & x^*_{210}=20 & x^*_{220}=0 & x^*_{310}=0 & x^*_{320}=10 \\ s=1: &\quad x^*_{111}=15 & x^*_{121}=0 & x^*_{211}=20 & x^*_{221}=0 & x^*_{311}=0 & x^*_{321}=10 \\ s=2: &\quad x^*_{112}=0 & x^*_{122}=15 & x^*_{212}=20 & x^*_{222}=0 & x^*_{312}=10 & x^*_{322}=0 \\ s=3: &\quad x^*_{113}=0 & x^*_{123}=15 & x^*_{213}=20 & x^*_{223}=0 & x^*_{313}=10 & x^*_{323}=0 \\ s=4: &\quad x^*_{114}=0 & x^*_{124}=15 & x^*_{214}=20 & x^*_{224}=0 & x^*_{314}=0 & x^*_{324}=10 \\ s=5: &\quad x^*_{115}=15 & x^*_{125}=0 & x^*_{215}=0 & x^*_{225}=20 & x^*_{315}=0 & x^*_{325}=10 \end{aligned}$$

- The optimum capacity y^*_e of link e implied by the above flows is computed as the maximum load of the link e over all states s:

$$e^*_1 = 25 \quad e^*_2 = 25 \quad e^*_3 = 30 \quad e^*_4 = 35 \quad e^*_5 = 35$$

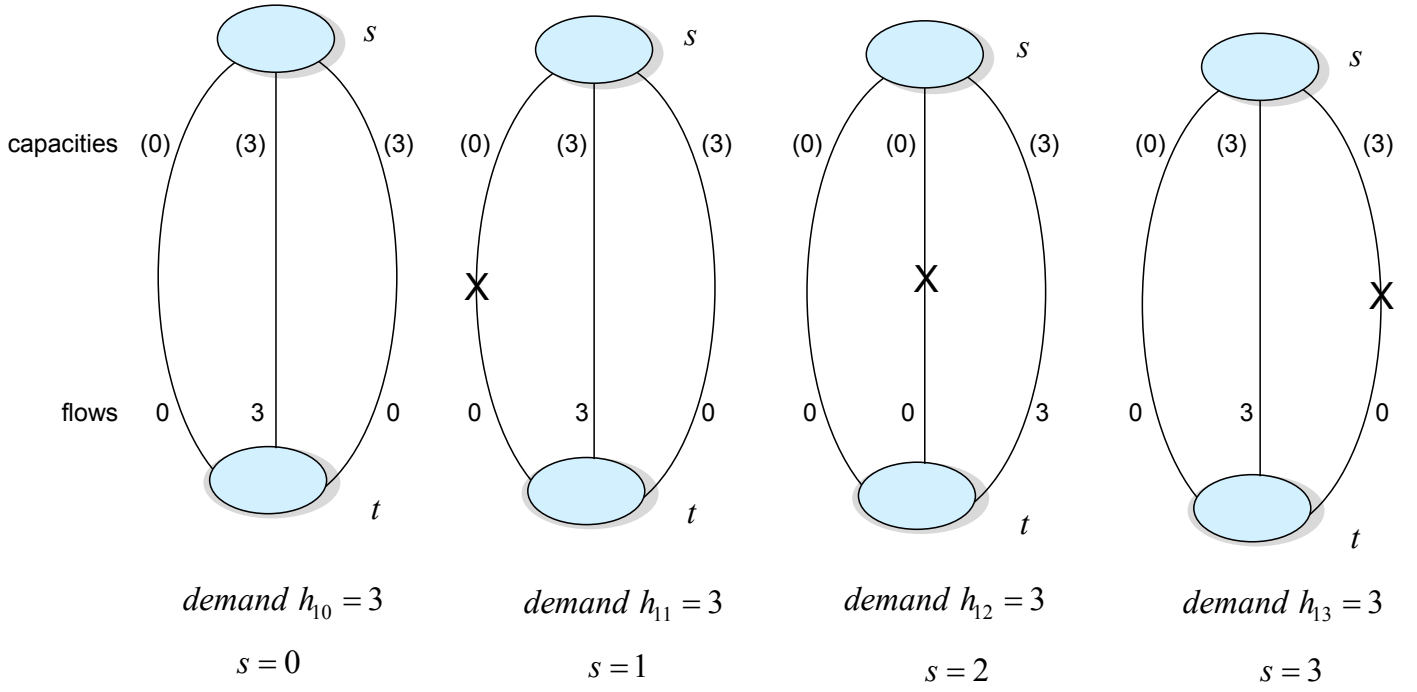
- This results in the optimal cost **F*** = 245
- Thus, a robust network can be considerably more expensive than the cheapest network without failure considerations (**F*** = 85)
- In this example, we obtained only non-bifurcated flows which must not always be the case

Restoration Design (4)



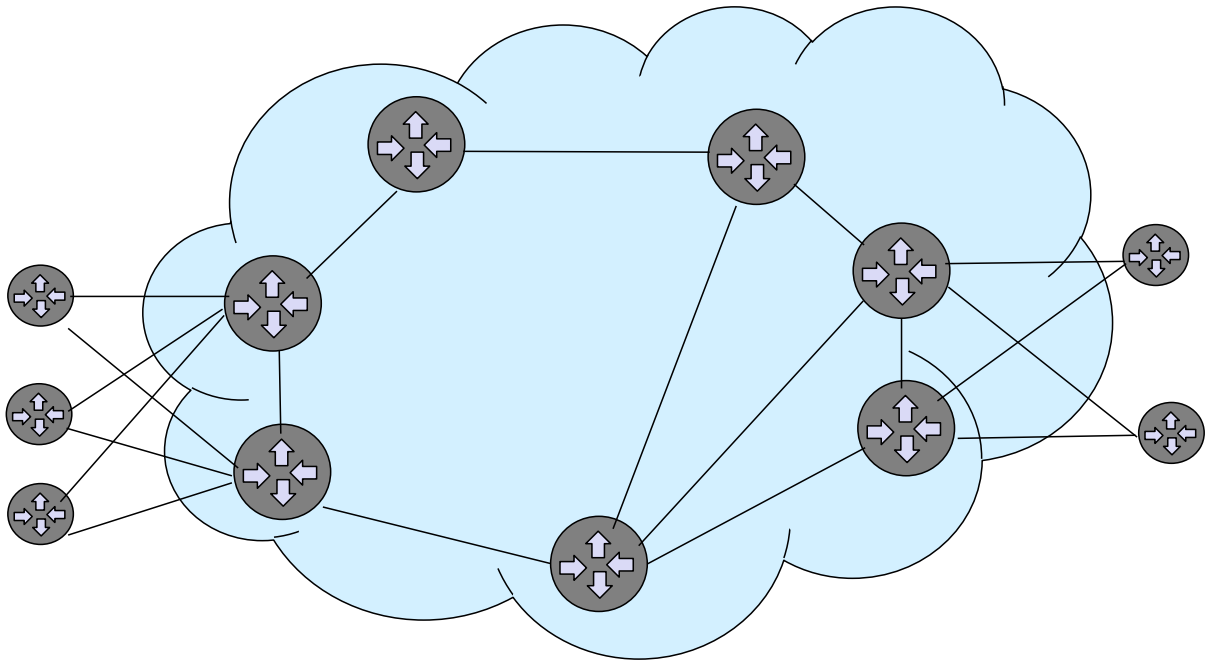
- Assume the above network with two nodes, three links and one demand:
  - If we allow for bifurcated solutions we obtain cost **F*** = 4.5

Restoration Design (5)



- For the optimal non-bifurcated solution we obtain cost **F*** = 6
- Our example considered the **path protection mechanism** (other mechanisms are possible, e.g. link protection)

Intra-Domain Traffic Engineering for IP-Networks (1)

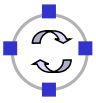


- Let us consider now, optimizing intra-domain routing in the Internet
- Intra-domain routing is operated by an ISP that has control over the network topology, routing algorithm, and link weight system



Intra-Domain Traffic Engineering for IP-Networks (2)

- Due to either contracted service level agreements or experience obtained via measurements the ISP knows the demands between nodes of his network
- A common objective of intra-domain routing optimization is to minimize the (average) delay experienced by data packets
  - Recall the relationship between link utilization and average delay
  - Thus, we would like to minimize the maximum utilization over all links
- A commonly used intra-domain routing protocol is OSPF which is based on Dijkstra's algorithm:
  - Dijkstra's algorithm calculates for each destination the shortest path according to some weight system **w**
  - Thus, we have to solve a variant of a shortest path problem, that is identifying a weight system **w**:
    - such that the maximum link utilization of our network is minimized,
    - while satisfying all given demands, and
    - staying within capacity constraints



Intra-Domain Traffic Engineering for IP-Networks (3)

- How can we formulate this problem as a multi-commodity flow problem?
  - We know our demands h_d and link capacities c_e
  - The sum over all path flows for a given demand must meet the demand:

$$\sum_p x_{dp}(\mathbf{w}) = h_d, \quad d = 1, 2, \ldots, D$$

  - Furthermore, the link load $\underline{y}_e$ on link e induced by the link metric system **w** should not be greater than the link capacity:

$$\underline{y}_e(\mathbf{w}) = \sum_d \sum_p \delta_{edp} x_{dp}(\mathbf{w}) \leq c_e, \quad e = 1, 2, \ldots, E$$

  - The maximum r over all link utilizations can be computed as follows:

$$r = max_{e=1,\ldots,E}\{\underline{y}_e(\mathbf{w})/c_e\}$$

  - We now need to ensure that all link loads stay below $c_e r$

$$\underline{y_e}(\mathbf{w}) = \sum_d \sum_p \delta_{edp} x_{dp}(\mathbf{w}) \leq c_e r, \quad e = 1, 2, \ldots, E$$

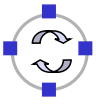


Intra-Domain Traffic Engineering for IP-Networks (4)

- All in all, we obtain the following optimization problem:

$$\begin{aligned}
&\textbf{minimize} \quad \mathbf{F} = r \\
&\textbf{subject to} \\
&\sum_p x_{dp}(\mathbf{w}) = h_d, && d = 1, 2, \ldots, D \\
&\underline{y_e}(\mathbf{w}) = \sum_d \sum_p \delta_{edp} x_{dp}(\mathbf{w}) \leq c_e r, && e = 1, 2, \ldots, E \\
&r = max_{e=1,\ldots,E}\{\underline{y}_e(\mathbf{w})/c_e\} \\
&r \quad \text{continous} \\
&w_e \quad \text{non-negative integers}
\end{aligned}$$

  - For this to work, we need to find k shortest candidate paths for every attempted weight system vector **w**
  - If $r^* < 1$ then no link will be overloaded (congestion is likely to occur if r^* is close to 1)

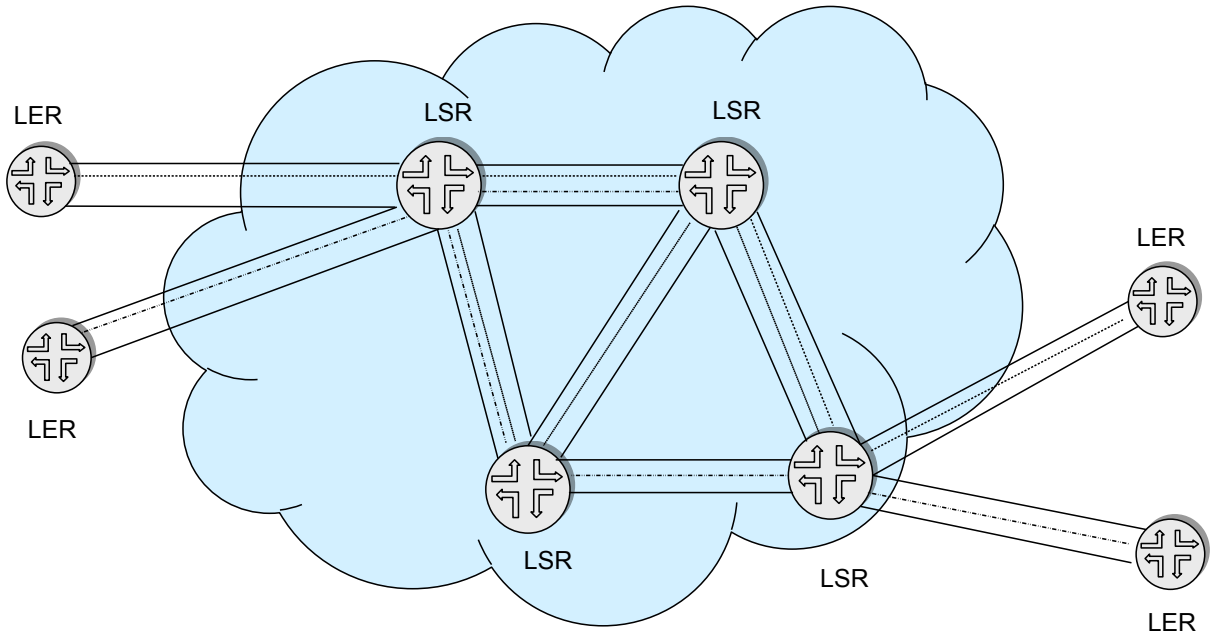


Tunnel Optimization for MPLS Networks (1)

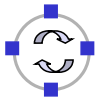
- Multi-Protocol Label Switching (MPLS) is an approach that introduces virtual connections into packet switched networking in order to:
  - speed up processing times in routers, and to
  - allow for traffic engineering
  - For details on basic MPLS principles and protocol functions please refer also to the respective chapter of the Telematics 2 course
- In order to transport traffic in an MPLS network, a so-called label switched path needs to be set up from the source (ingress MPLS node) to the destination (egress MPLS node)
- In order to allow for “similar” traffic to be handled in an aggregated way, tunnels can be set up (by making use of label stacking)
- Tunneling offers promising traffic engineering capabilities:
  - put traffic with same QoS characteristics into one tunnel and treat this in a similar fashion
  - simple re-routing in case of local congestion/link failures



Tunnel Optimization for MPLS Networks (2)



- The figure above illustrates tunnels in an MPLS network consisting of label edge routers (LER) and label switch routers (LSR)
- Note the tunnel aggregation inside the network



Tunnel Optimization for MPLS Networks (3)

- However, in order not to overload routers with too many tunnels (leading to high processing overhead), it is desirable to limit the number of tunnels per router and/or link
- Thus, we face the following optimization challenge:
  - How to carry different traffic classes in an MPLS network through the creation of tunnels in a way that the number of tunnels per node/link is minimized and the load is balanced among routers/links?
- We use basically the same notation as before with one difference:
  - The path-flow variable x_{dp} now denotes the fraction of demand h_d that is routed over path P_{dp}
  - Thus, we obtain the demand constraint: $\sum_p x_{dp} = 1, \quad d = 1, 2, \ldots, D$
- As we are not interested in obtaining path-flows that carry a very low fraction, we:
  - require a lower bound ε for the fraction x_{dp}, and for this
  - introduce the binary variables u_{dp} that are set to 1 if the lower bound is satisfied and 0 otherwise



Tunnel Optimization for MPLS Networks (4)

- We need two constraints to model this:

$$\epsilon\, u_{dp} \leq x_{dp}, \quad d = 1, 2, \ldots, D \quad p = 1, 2, \ldots, P_d$$
$$x_{dp} \leq u_{dp}, \quad d = 1, 2, \ldots, D \quad p = 1, 2, \ldots, P_d$$

- The book (Pioro, page 83) gives the first formula as:

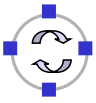
$$\epsilon\, u_{dp} \leq h_d x_{dp}, \quad d = 1, 2, \ldots, D \quad p = 1, 2, \ldots, P_d$$

  Seems to be a typo…

- Furthermore, we have capacity feasibility constraints

$$\sum_d h_d \sum_p \delta_{edp} x_{dp} \leq c_e, \quad e = 1, 2, \ldots, E$$

- The number of tunnels on link e will be: $\sum_d \sum_p \delta_{edp} u_{dp}$
- We aim to minimize the number r representing the maximum number of tunnels over all links



Tunnel Optimization for MPLS Networks (5)

- Altogether, we obtain the following optimization problem:

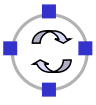
$$\begin{aligned}
&\textbf{minimize} \quad \mathbf{F} = r \\
&\textbf{subject to}
\end{aligned}$$

$$\begin{aligned}
\textstyle\sum_p x_{dp} &= 1, && d = 1, 2, \ldots, D \\
\textstyle\sum_d h_d \sum_p \delta_{edp} x_{dp} &\leq c_e, && e = 1, 2, \ldots, E \\
\epsilon\, u_{dp} &\leq x_{dp}, && d = 1, 2, \ldots, D \quad p = 1, 2, \ldots, P_d \\
x_{dp} &\leq u_{dp}, && d = 1, 2, \ldots, D \quad p = 1, 2, \ldots, P_d \\
\textstyle\sum_d \sum_p \delta_{edp} u_{dp} &\leq r && e = 1, 2, \ldots, E \\
x_{dp} &\text{ continous and non-negative} \\
u_{dp} &\text{ binary} \\
r &\text{ integer}
\end{aligned}$$



Tunnel Optimization for MPLS Networks (6)

- This problem has both continous and discrete variables while the constraints and the objective function are linear
- It is an example for a mixed-integer linear programming problem (MIP)
- Finding exact solutions for MIP problems is more difficult than for (plain) linear optimization problems:
  - Established techniques are branch-and-bound, and branch-and-cut (to be treated later)



Summary

- This chapter has introduced various simple network design problems:
  - Minimizing routing cost
  - Uncapacitated network dimensioning
  - Shortest-path routing
  - Fair networks
  - Topological design
  - Restoration design
  - Delay minimization in IP networks with shortest path routing
  - Minimizing the number of tunnels per link/node in MPLS networks
- We also introduced a notation to formulate such problems as multi-commodity flow problems:
  - Link-path formulation
  - Node-link formulation
  - Refined notation to be able to deal with non-trivial examples
- We are now all curious how to solve these problems! :-)

