ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

Algorithmen für drahtlose Netzwerke

Routing als Flussproblem

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Datenflüsse in Netzwerken

- **Motivation:**
  - Optimiere Datenfluss von Quelle zum Ziel
- **Definition:**
  - (Single-commodity) Maximale Flussproblem
  - Gegeben
    - ein Graph G=(V,E)
    - eine Kapazitätsfunktion w:E→R^+_0,
    - Quellenmenge S und Zielmenge T
  - Finde maximalen Fluss von S zu T

- **Ein Fluss ist eine Funktion f : E → R_0^+ so dass**
  - für alle e ∈ E: f(e) ≤ w(e)
  - für alle e ∉ E: f(e) = 0
  - für alle u,v ∈ V: f(u,v)≥0

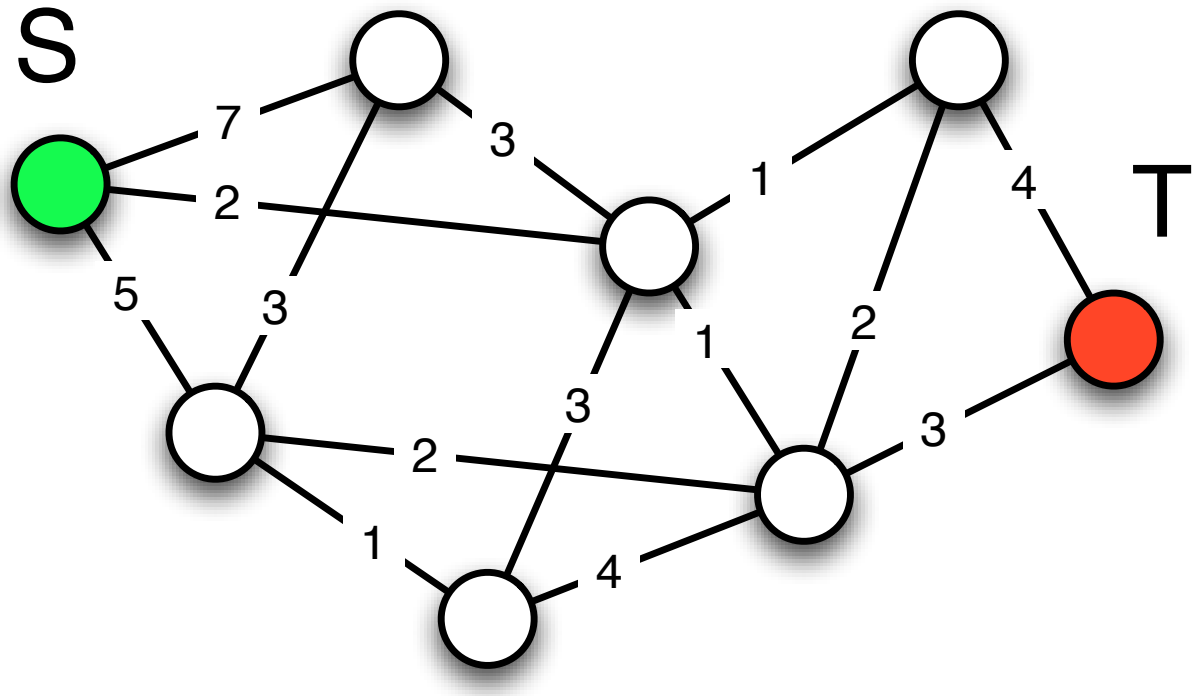
$$\forall u \in V \setminus (S \cup T)$$

$$\sum_{v \in V} f(v,u) = \sum_{v \in V} f(u,v)$$

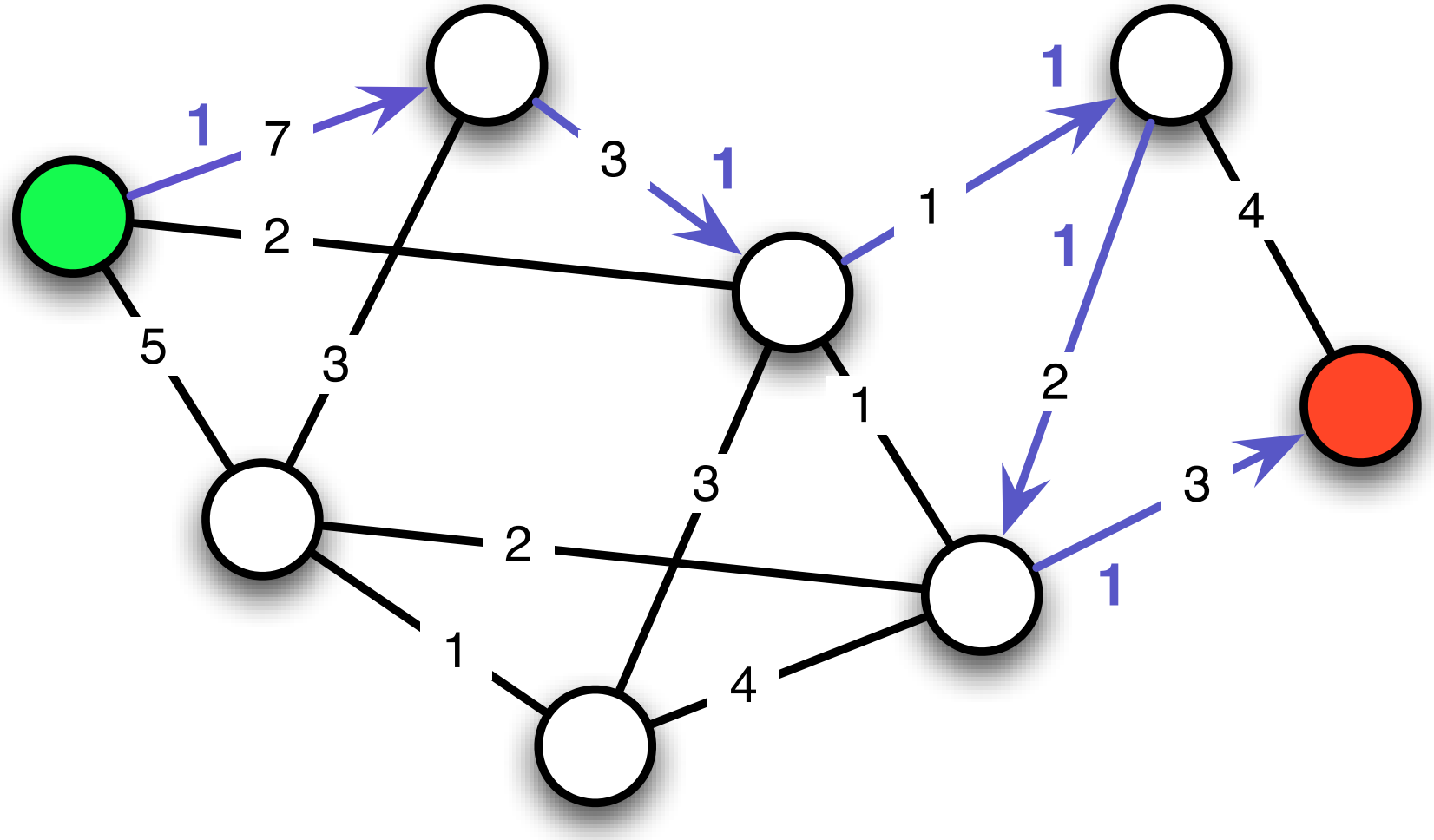
- **Maximiere Fluss:**

$$\sum_{u \in S} \sum_{v \in V} f(u,v)$$

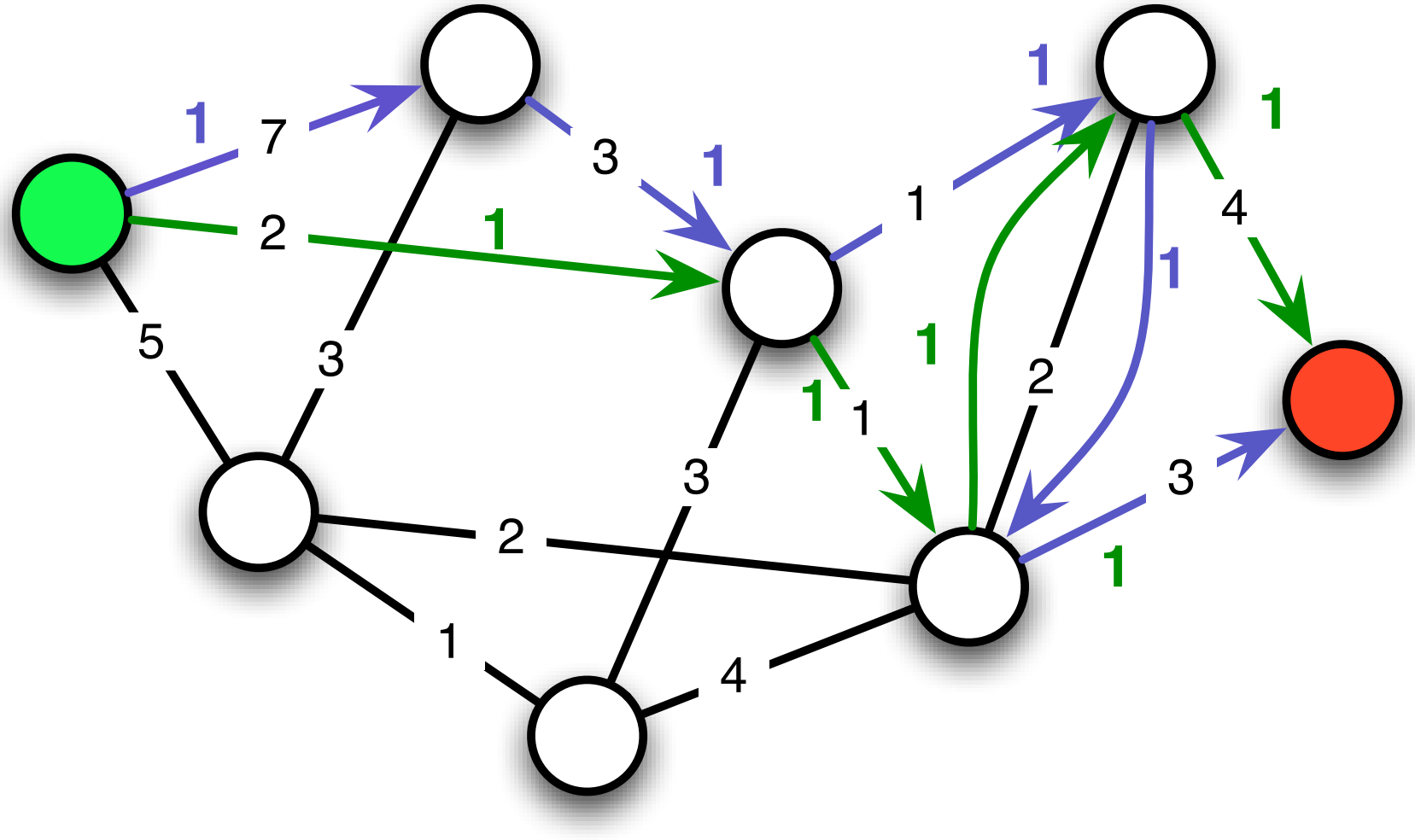
Datenflüsse in Netzwerken



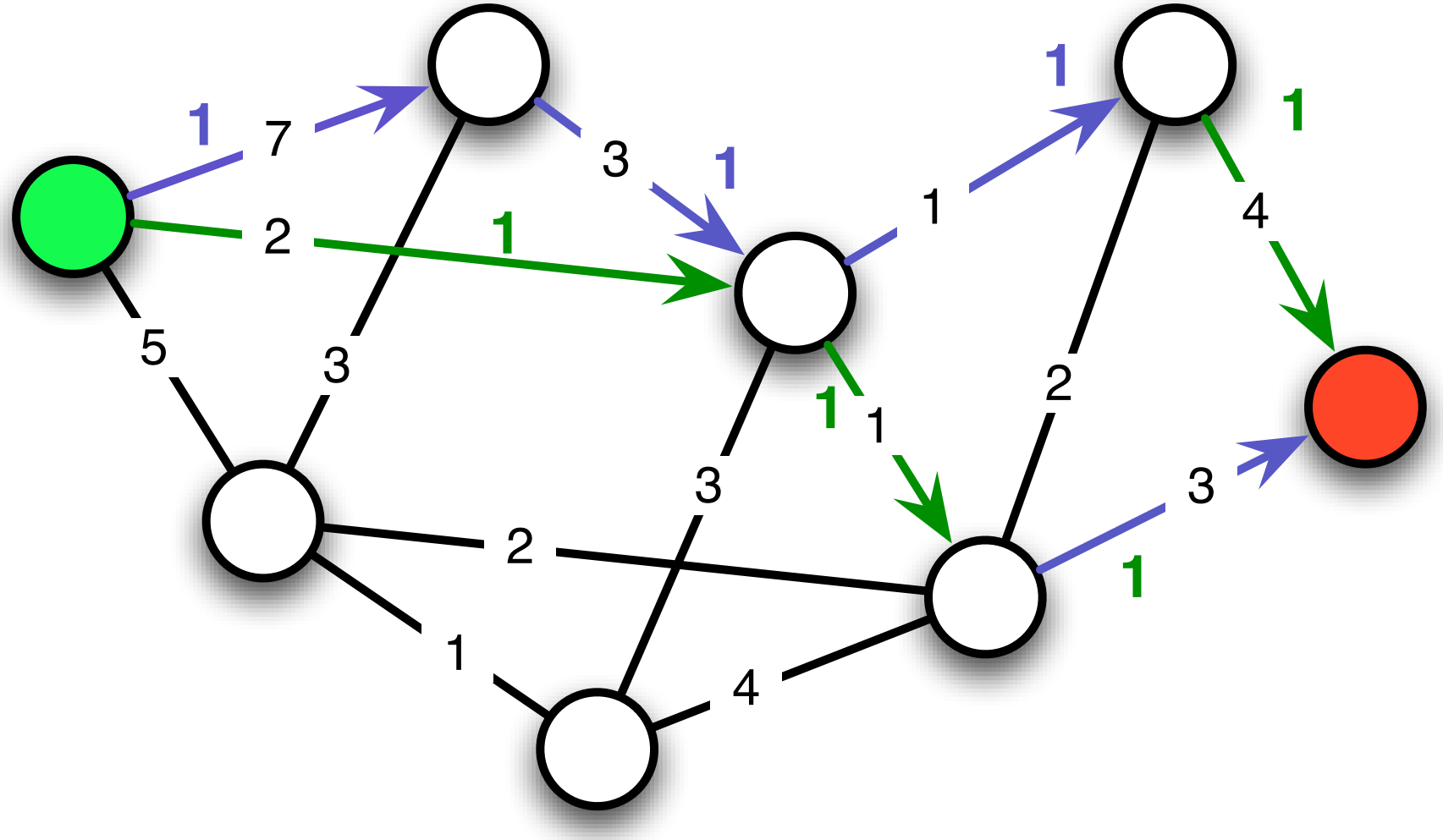
Datenflüsse in Netzwerken



Datenflüsse in Netzwerken

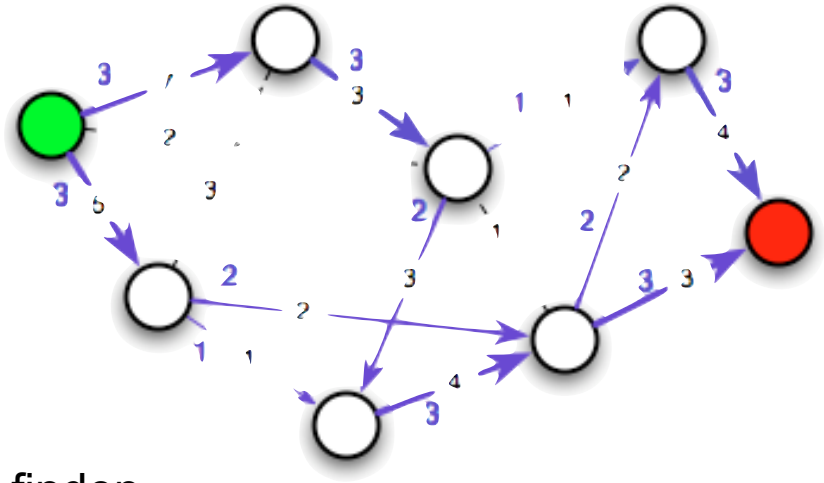


Datenflüsse in Netzwerken



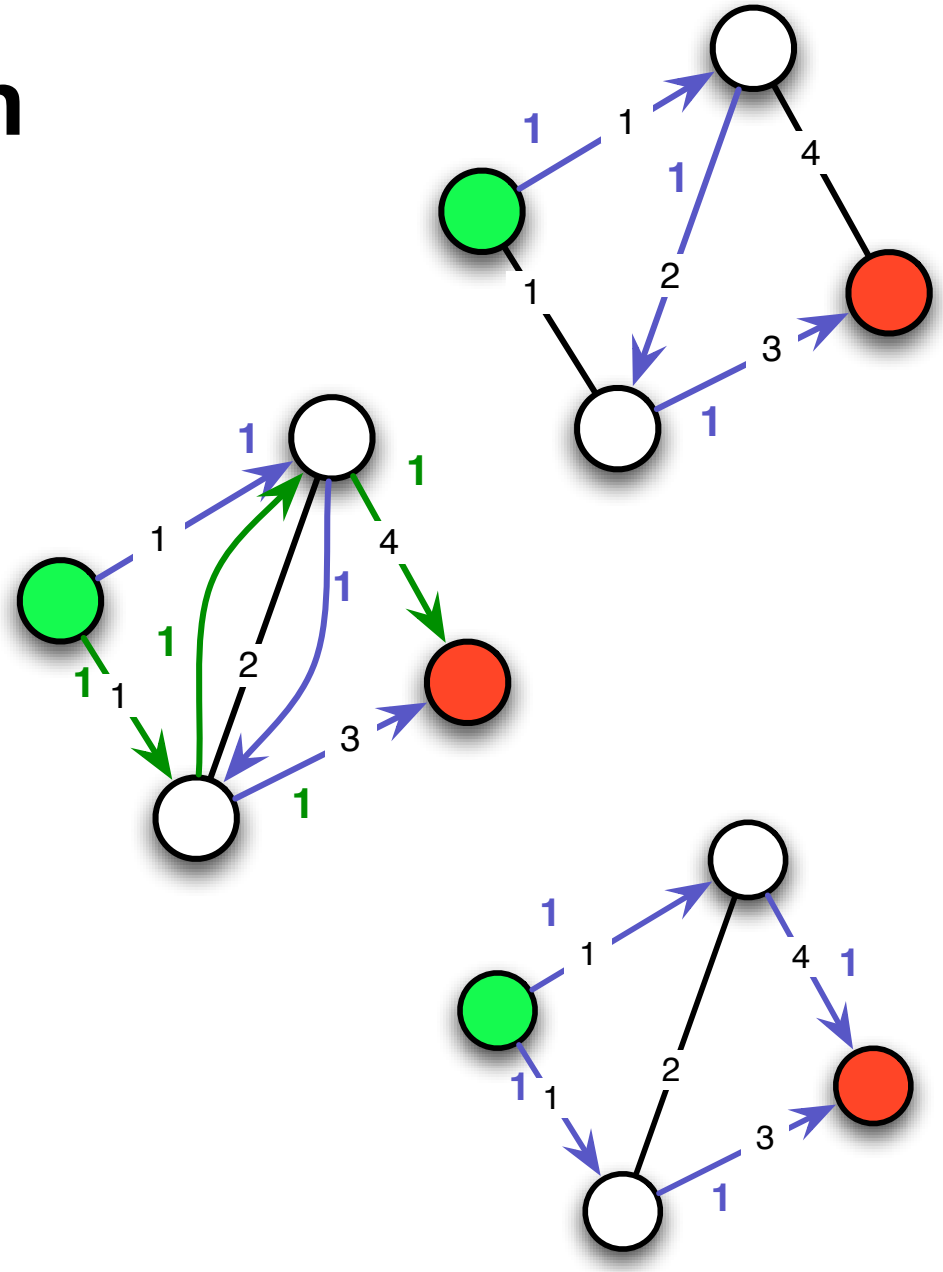
Berechnung des maximalen Flusses

- **Jedes natürliches Röhrensystem löst das reelle Maximale Fluss-Problem natürlich**
- **Algorithmen zur Lösung**
  - Lineare Programmierung
    - für reelle Zahlen
    - Der Fluss beschreibt die Gleichungen eines linearen Optimierungsproblems
    - Simplex-Algorithmus (oder Ellipsoid-Methode) kann das lineare Gleichungssystem lösen
  - Ford-Fulkerson
    - auch für ganze Zahlen
    - Solange offene Pfade, die den Fluss erhöhen existieren, werden diese gewählt
      - offener Pfad: Pfad der den Fluss erhöht
  - Edmonds-Karp
    - Spezieller Fall von Ford-Fulkerson
    - Breitensuche wird verwendet um Pfade zu finden



Ford-Fulkerson

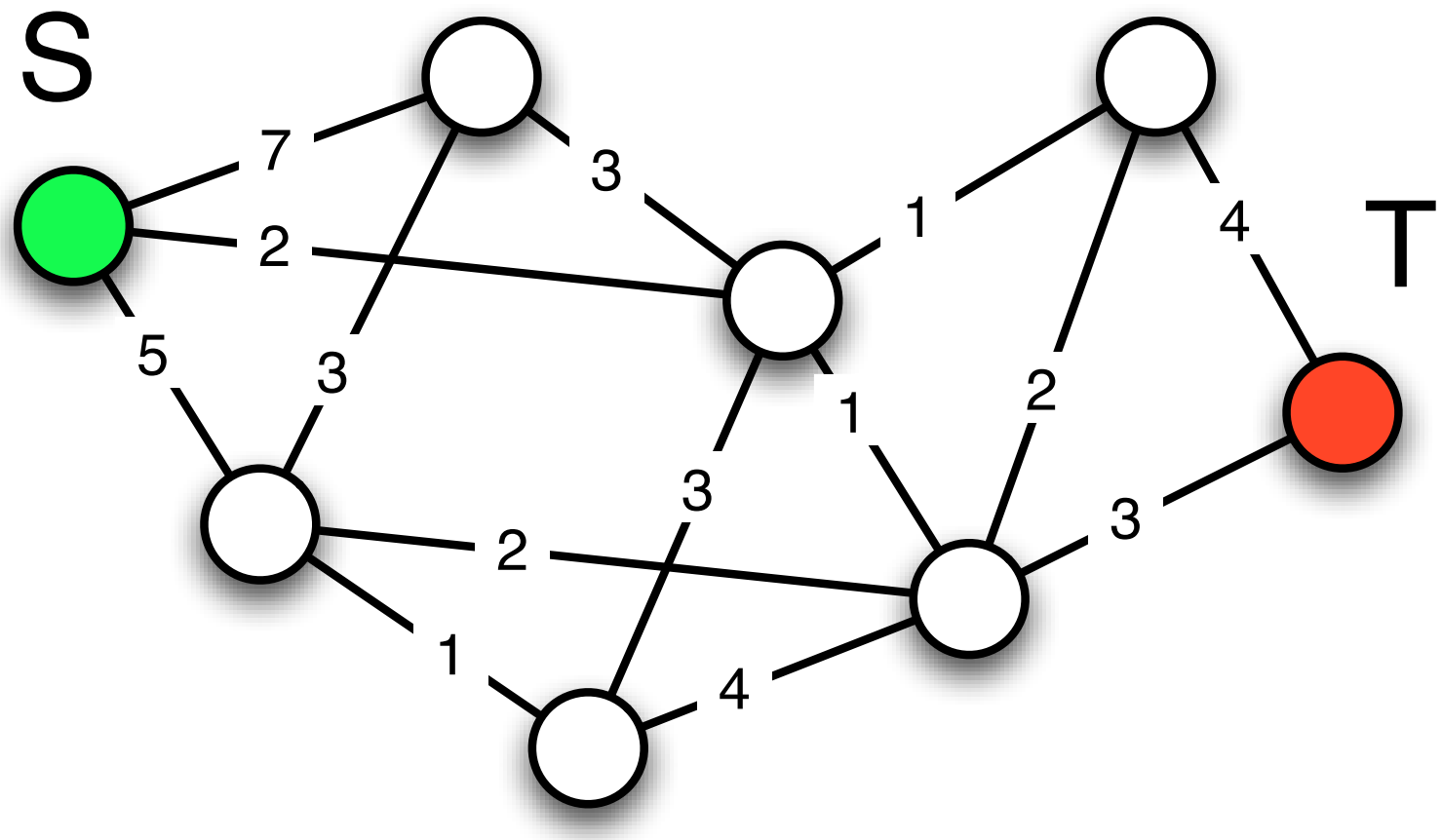
- **Finde einen Pfad von einem Quellknoten zu einem Senkenknoten, wobei auf jeder Kante**
  - die Kapazität nicht ausgenutzt worden ist oder
  - der vorhandene Fluss dadurch reduziert wird
- **Berechne den maximalen Fluss der auf diesen Pfad hinzugefügt werden kann**
  - durch das Maximum
- **Füge den Fluss auf dem Pfad hinzu**
- **Wiederhole diese Schritte bis kein solcher Pfad gefunden werden kann**



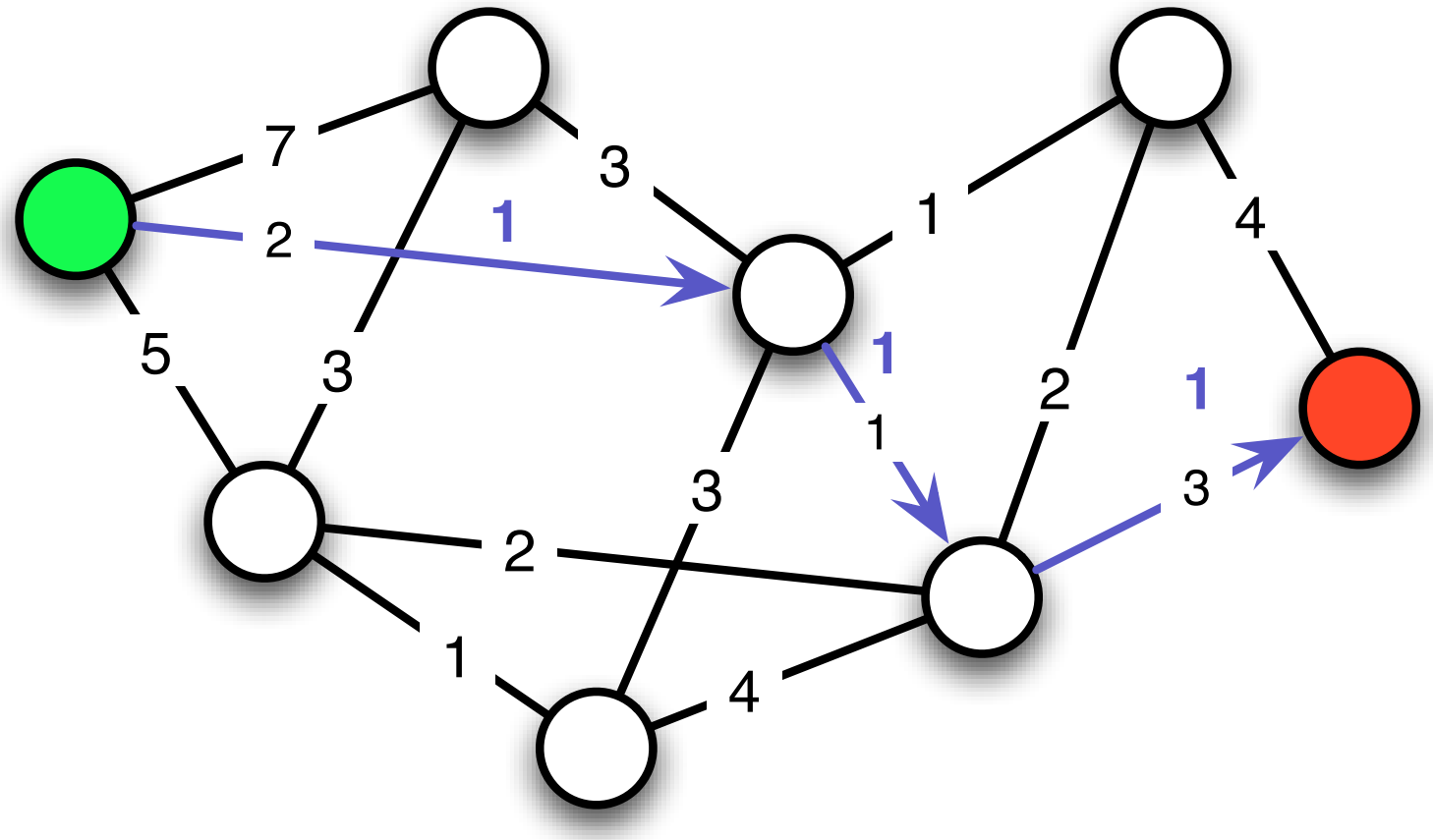
Edmunds-Karp

- **Pfadsuche für Ford-Fulkerson-Algorithmus**
- **Wähle den kürzesten augmentierenden Pfad**
  - Berechnung durch Breitensuche
- **Führt zu Laufzeit $O(|V| |E|^2)$**
  - während bei Ford-Fulkerson exponentielle Laufzeit möglich wäre

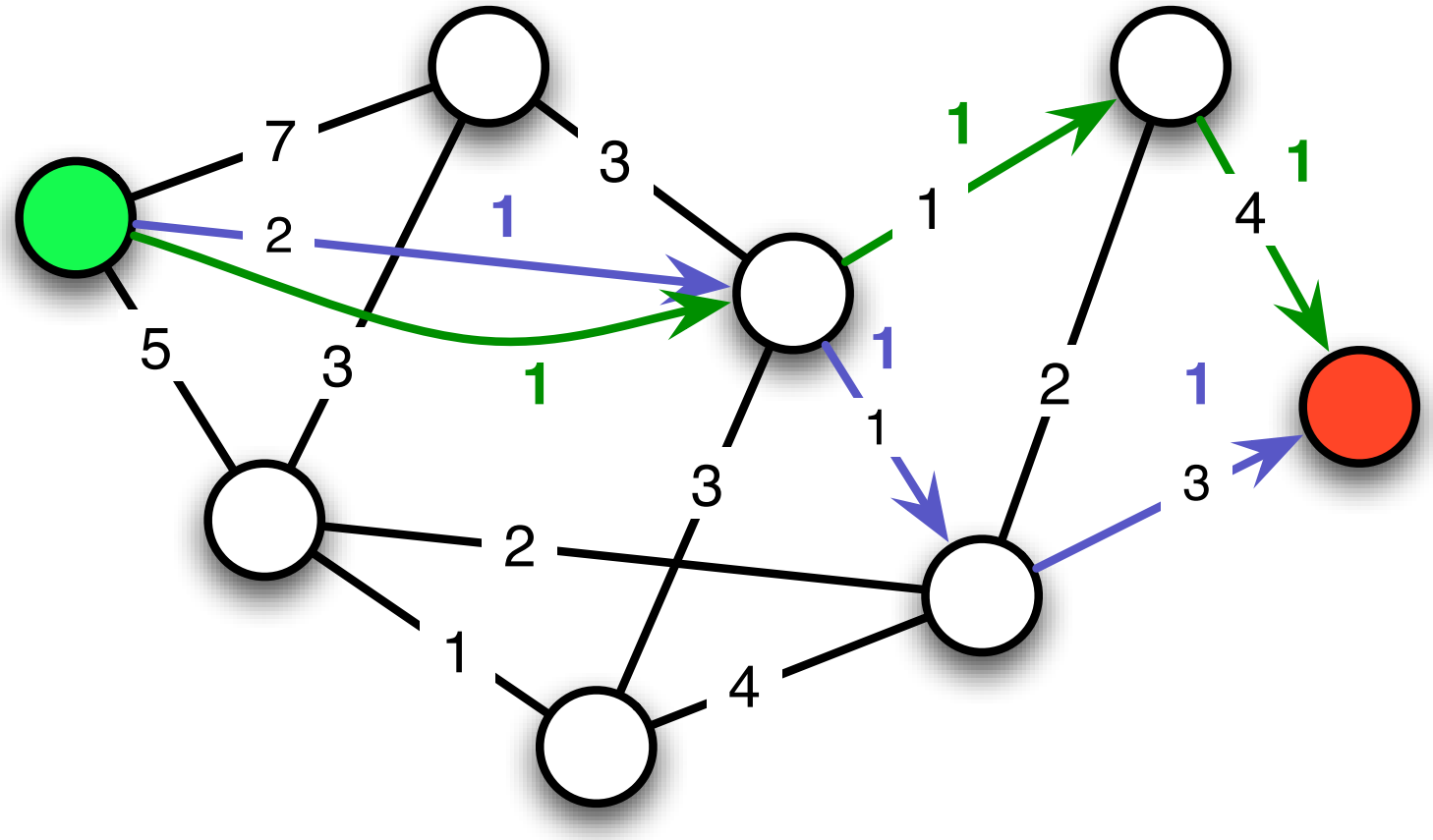
Beispiel



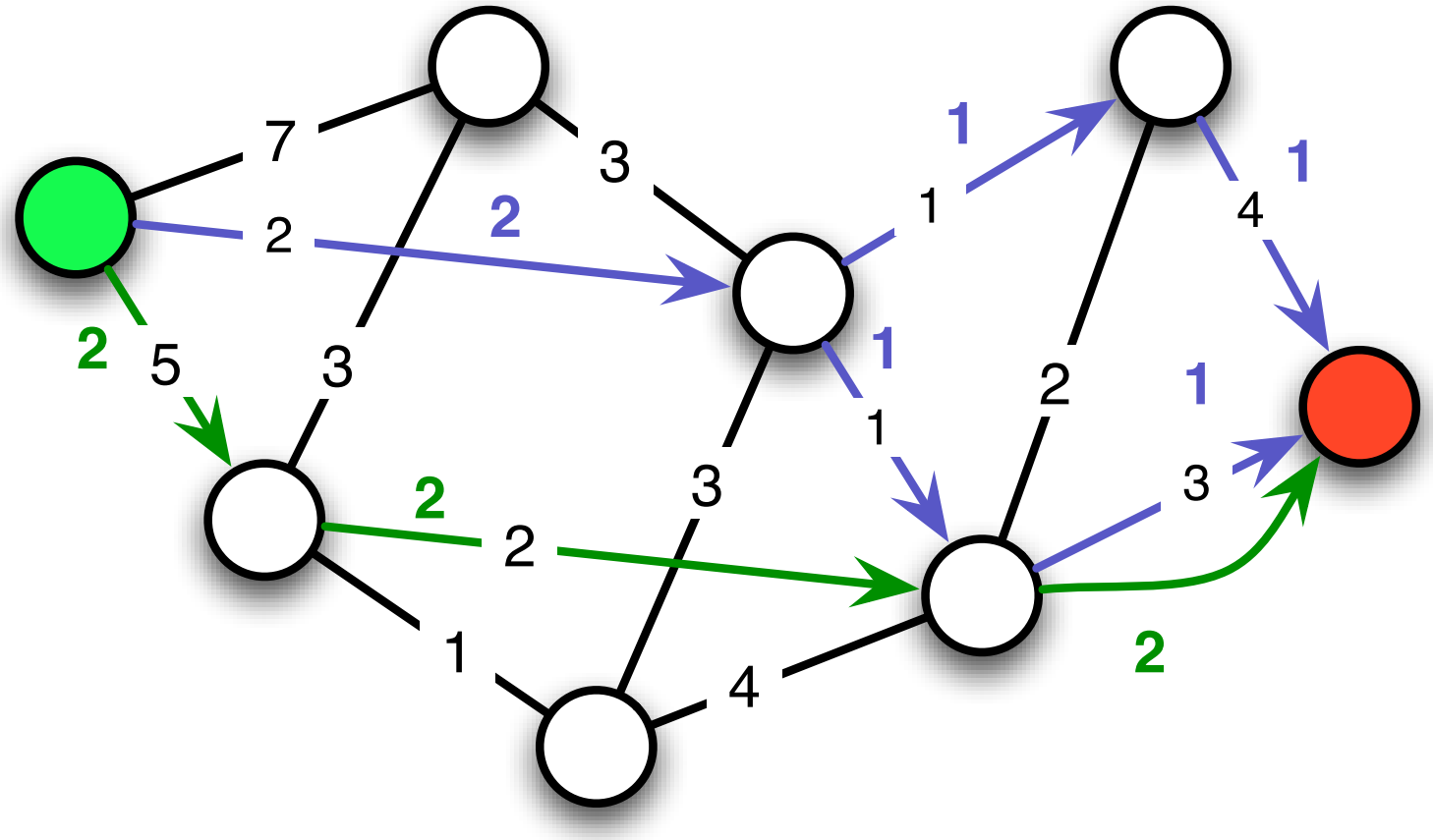
Beispiel



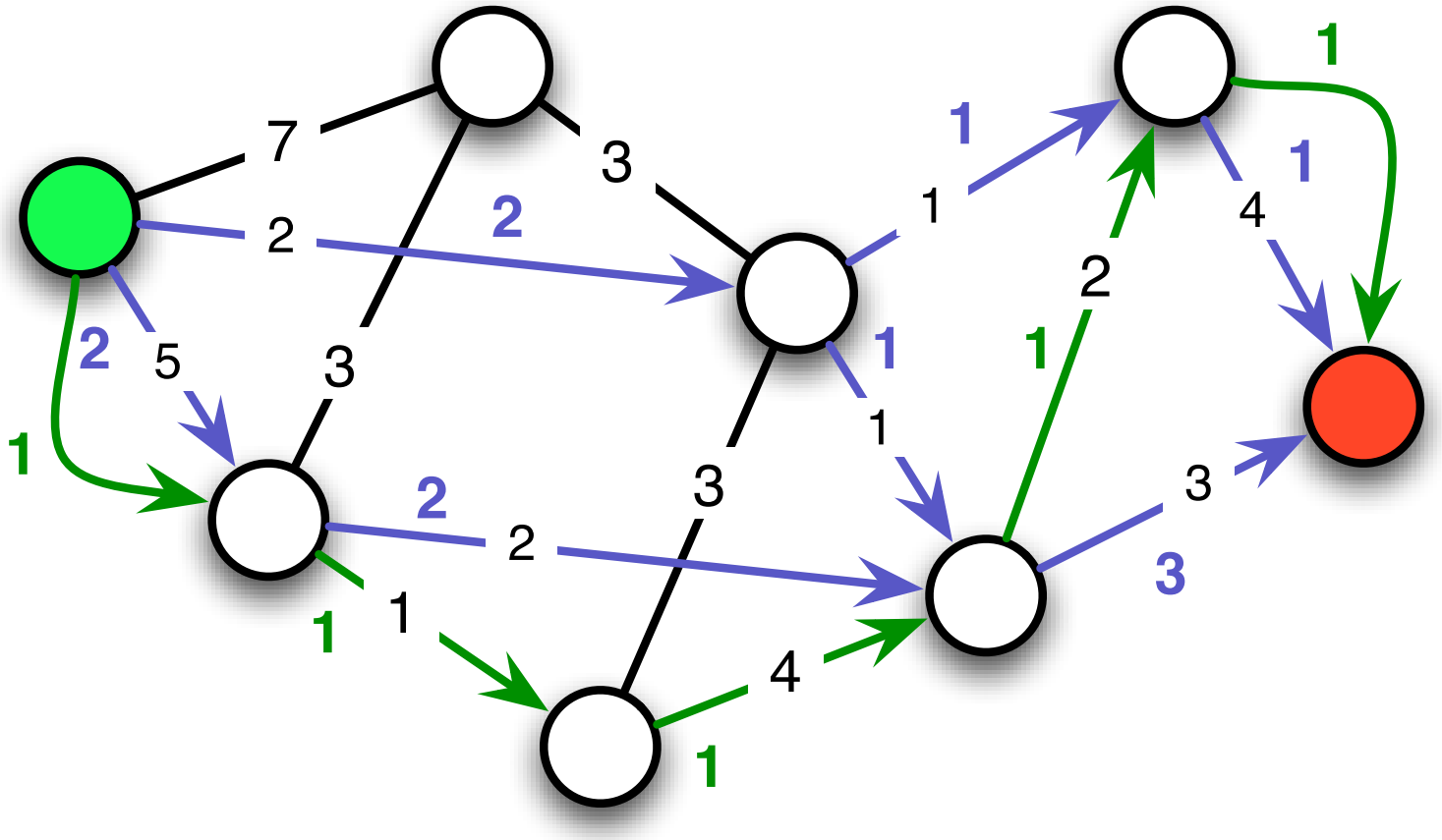
Beispiel



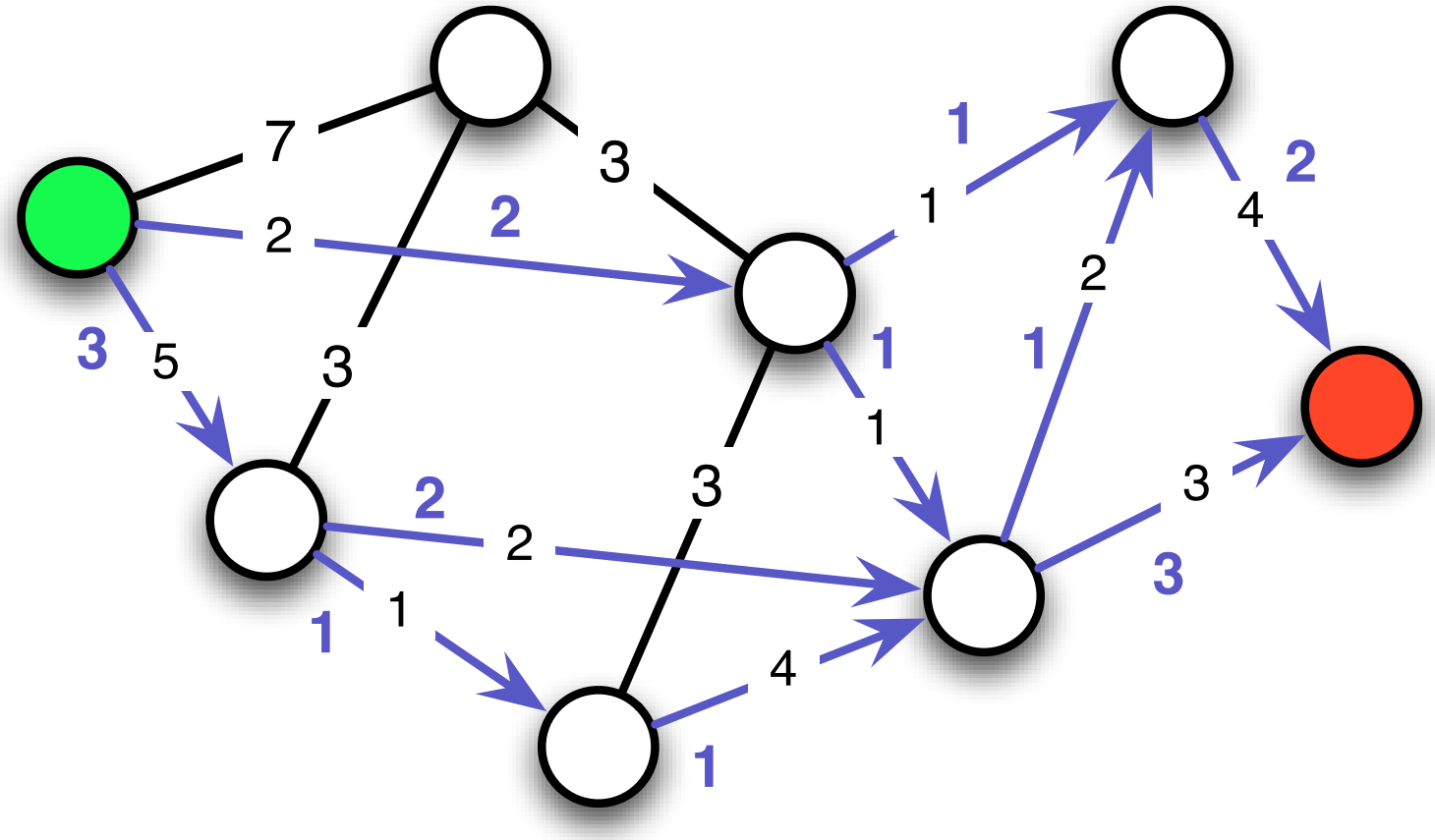
Beispiel



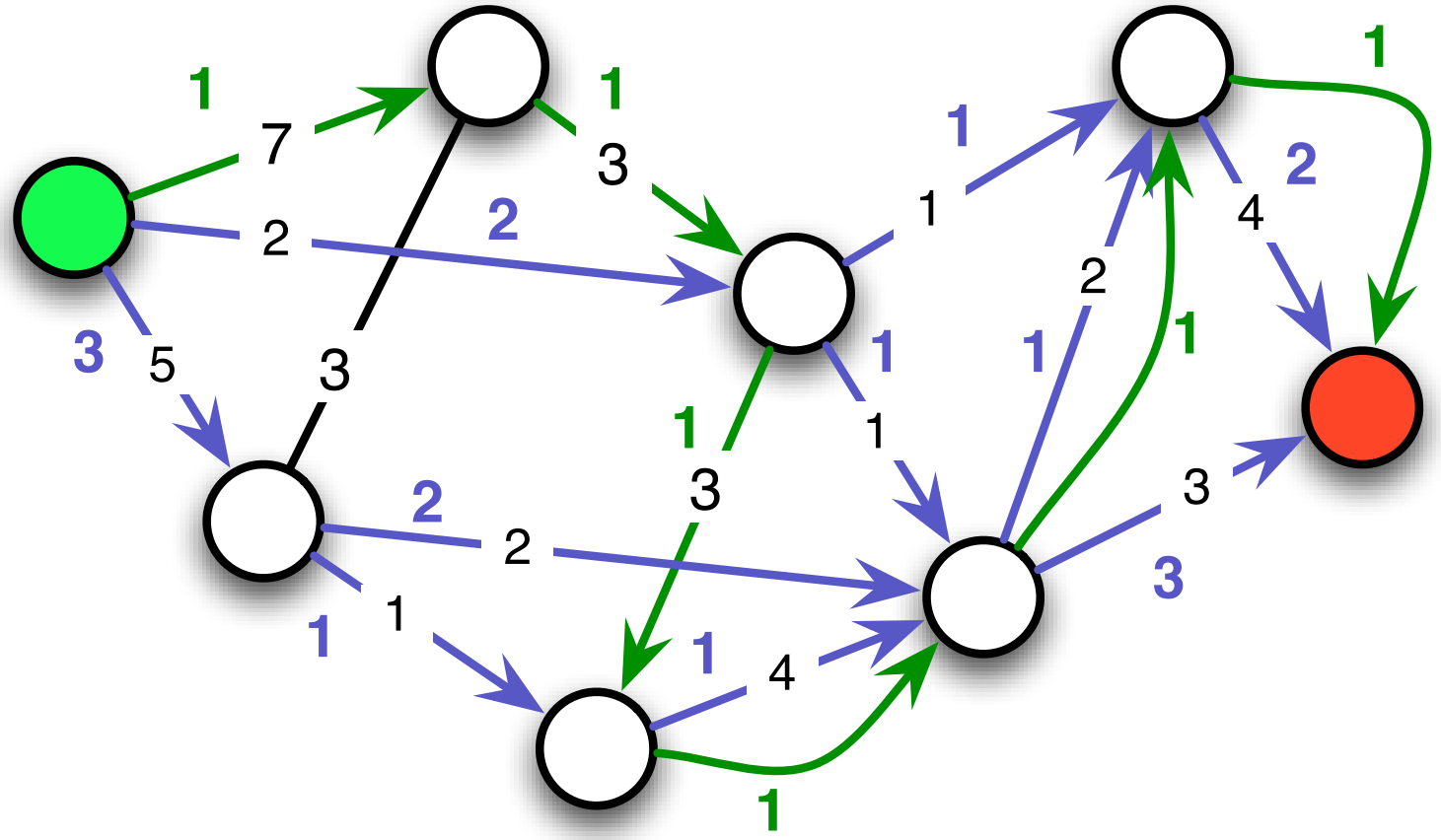
Beispiel



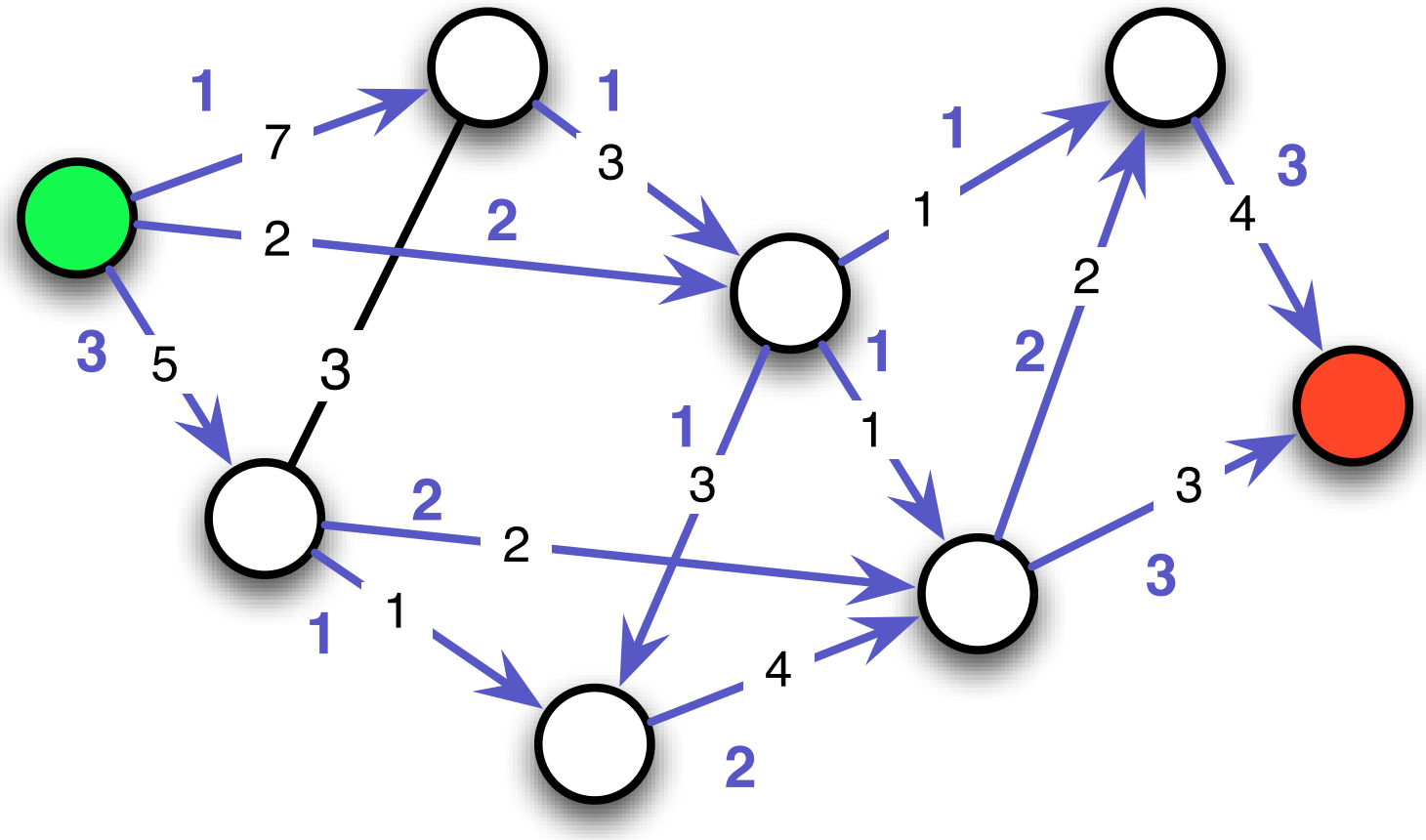
Beispiel



Beispiel

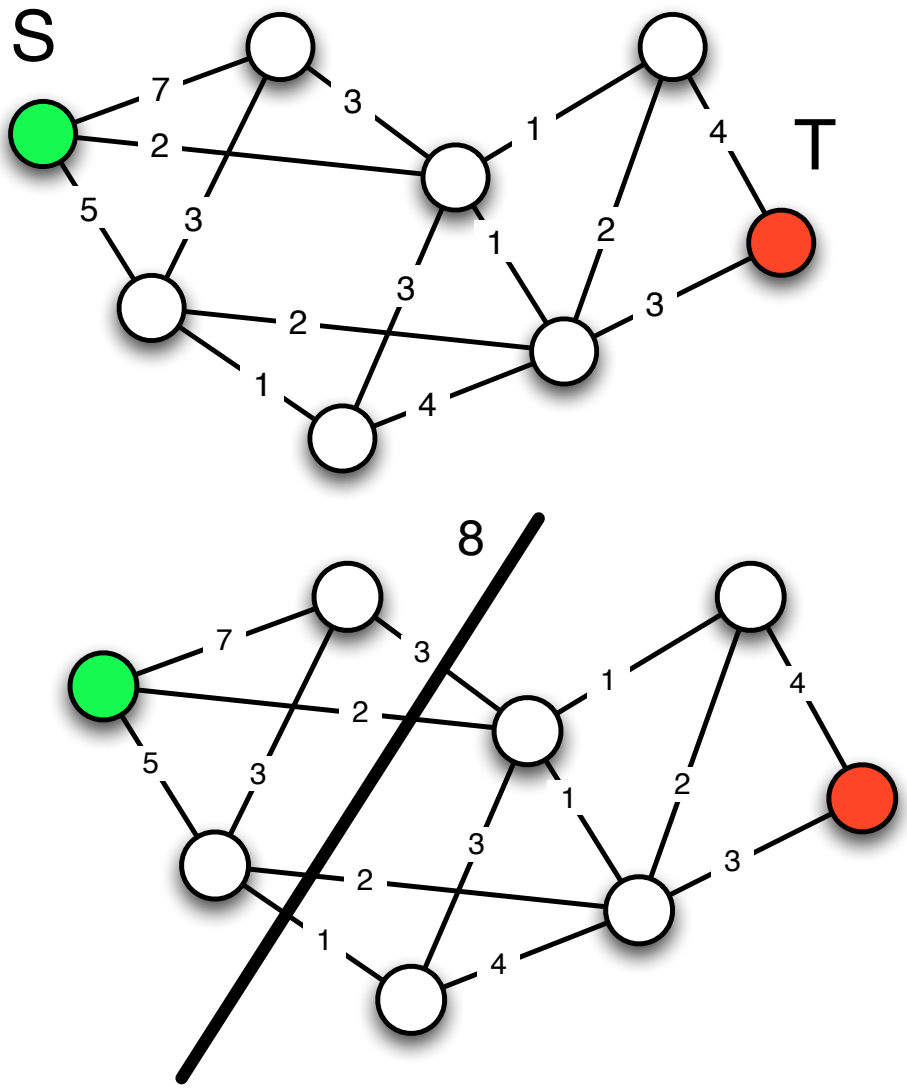


Beispiel



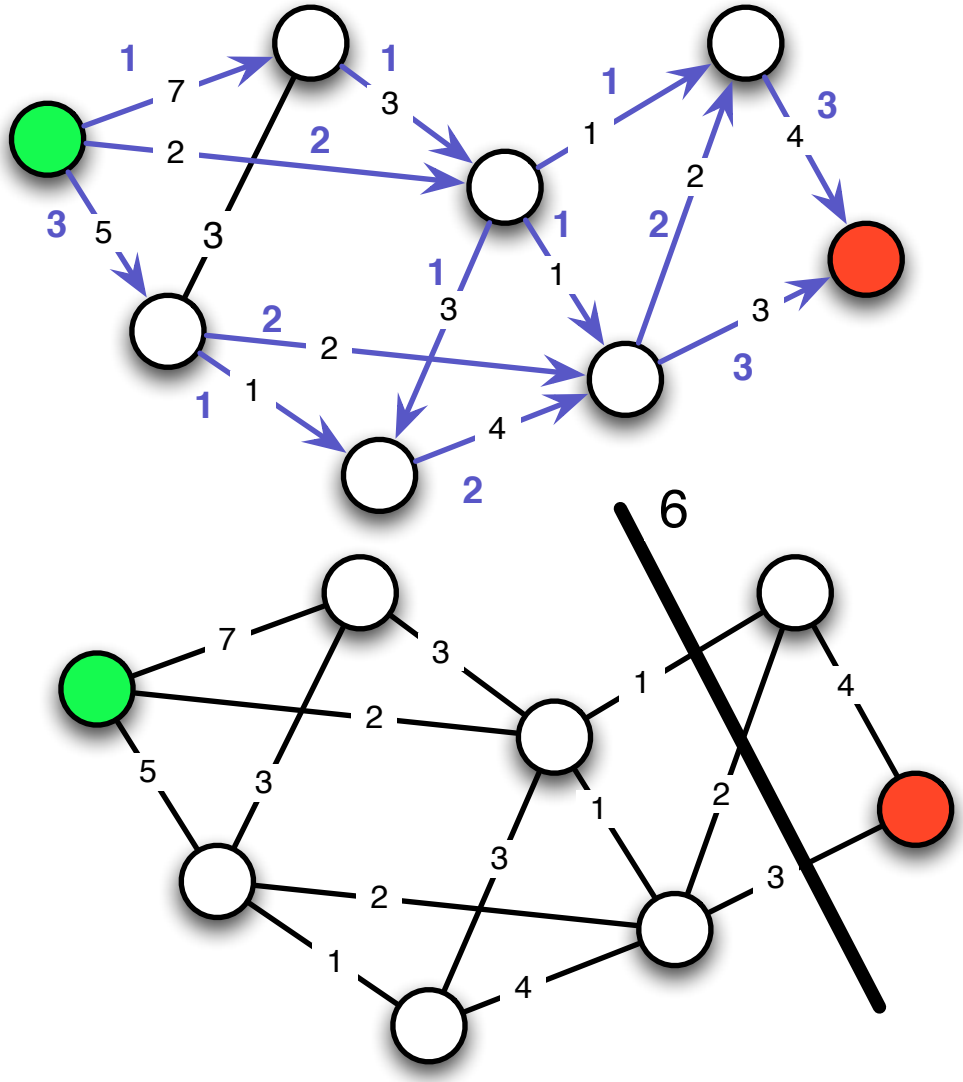
Minimaler Schnitt in Netzwerken

- **Motivation**
  - Finde Flaschenhals im Netzwerk
- **Definition**
  - Min Cut problem
  - Gegeben:
    - Graph G=(V,E)
    - Kapazitätsfunktion w: E → R+0,
    - Quellen S und Senken T
  - Finde minimalen Schnitt zwischen S und T
- **Ein Schnitt C ist eine Menge von Kanten**
  - so dass kein Pfad eines Knoten von S zu einem Knoten aus T existiert, der nicht eine Kante von C benutzte in T
- **Die Größe des Schnitts C ist:** $\sum_{e \in C} w(e)$



Min-Cut-Max-Flow Theorem

- **Theorem**
  - Der minimale Schnitt ist gleich dem maximalen Fluss
- **Algorithmen für minimalen Schnitt**
  - können aus Algorithmen für maximalen Fluss gewonnen werden



Multi-Commodity Flow Problem

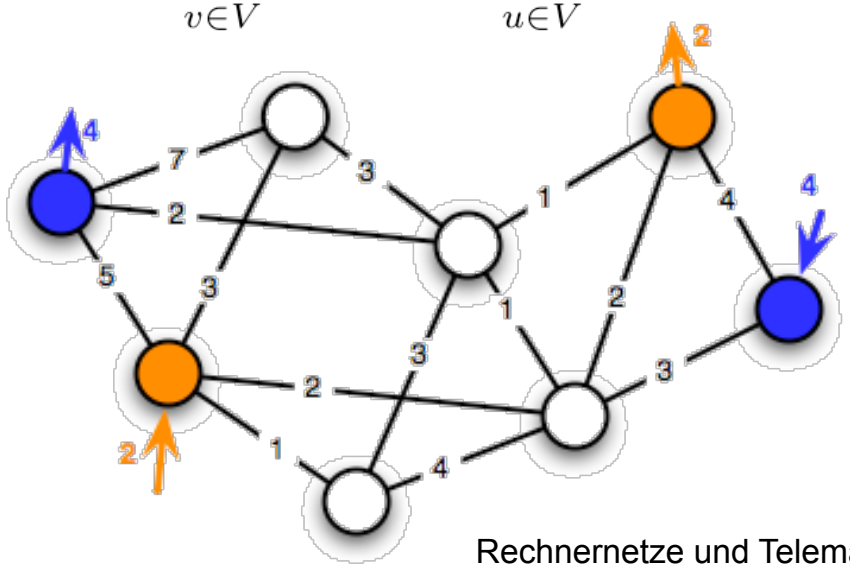
- **Motivation**
  - Theoretisches Modell für Punkt-zu-Punkt-Kommunikation
- **Definition**
  - Multi-Gommodity Flow Problem
  - Gegeben
    - ein Graph G=(V,E)
    - eine Kapazitätsfkt. w: E → R+0,
    - Güter (commodities) K_1, .., K_k:
      - $K_i=(s_i,t_i,d_i)$ mit
      - s_i: Quellknoten
      - t_i: Zielknoten
      - d_i: Bedarf

- **Finde Flüsse $f_1,f_2,...,f_k$ für alle Güter, wobei**
  - Kapazität: $\sum_{i=1}^{k} f_i(u,v) \leq w(u,v)$
  - Flusseigenschaft:

$$\forall v \notin \{s_i, t_i\} \ : \ \sum_{u \in V} f_i(u,v) = \sum_{u \in V} f_i(v,u)$$

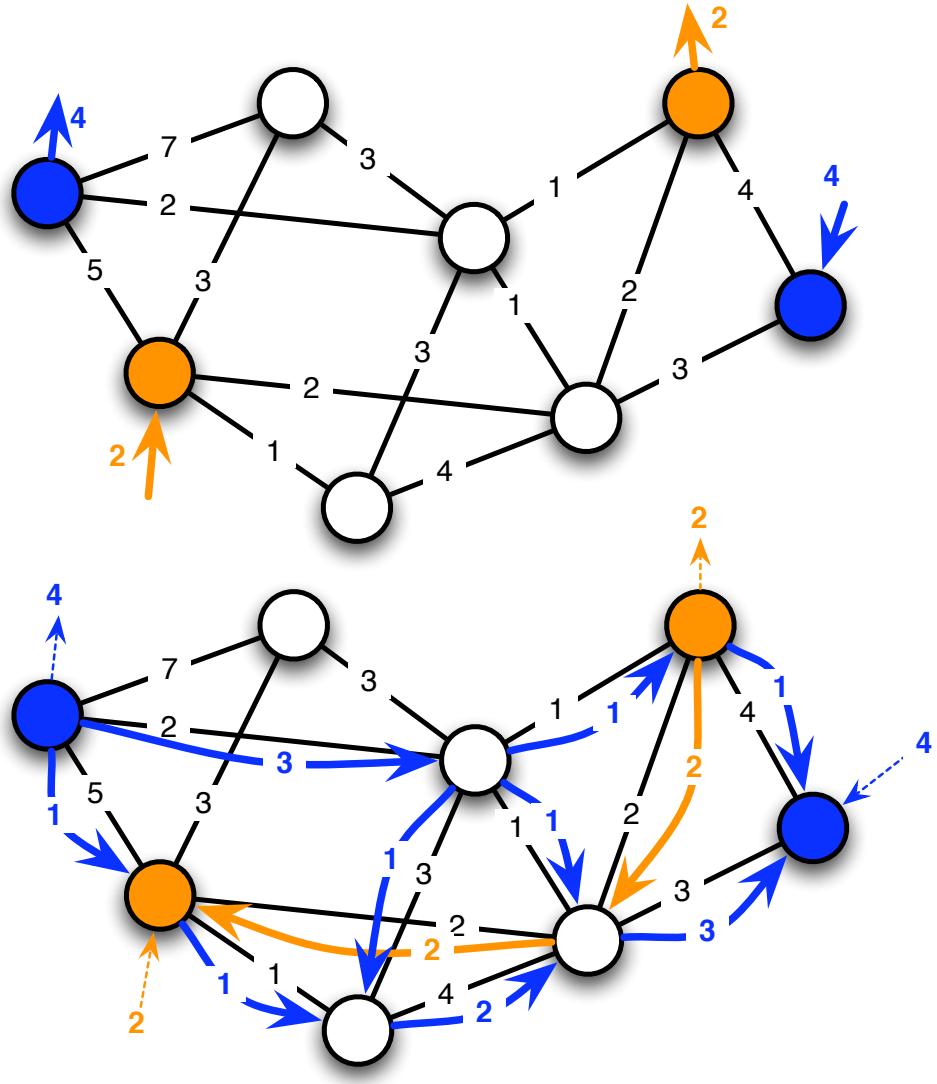
Bedarf:

$$\sum_{v \in V} f_i(s_i,v) = \sum_{u \in V} f_i(u,t_i) = d_i$$



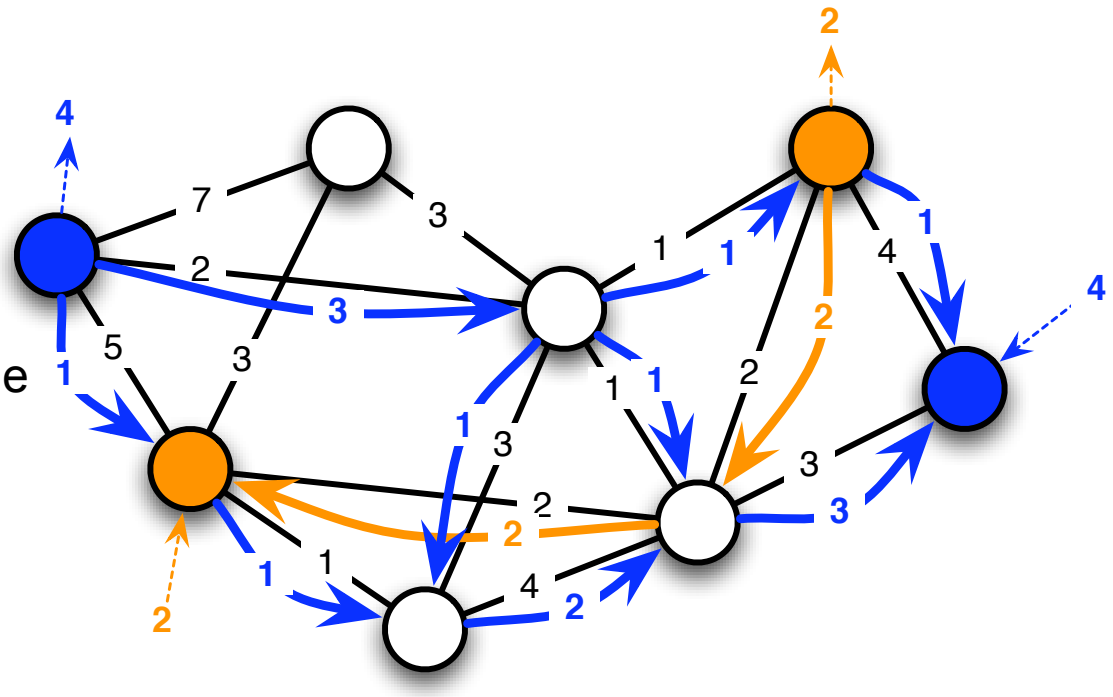
Lösen des Multi-Commodity Flow Problems

- **Multi-Commodity Flow Problem**
- **Optimiere**
  - Summe aller Flüsse oder
  - Maximiere das schlechteste Verhältnis zwischen einer Commodity und dem Bedarf
- **Problem ist in polynomieller Zeit lösbar**
  - für reelle Zahlen
  - durch lineare Programmierung



Komplexität der Multi-Commodity-Flow-Problems

- **Problem ist NP-vollständig**
  - für ganze Zahlen
    - z.B. Pakete
  - schon für zwei Commodities
    - Shai, Itai, Even, 1976
- **Polynomielle Lösung**
  - in Abhängigkeit der Anzahl der Pfade von der Quelle zur Senke
- **Approximation**
  - gute zentrale und verteilte Approximationsalgorithmen sind bekannt (polylogarithmische Approximationsgüte)
- **Schwächere Versionen des Min-Cut-Max-Flow-Theorems existieren**



ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

Algorithmen für drahtlose Netzwerke

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer