Graphs

Part: Flows

Algorithms and Data Structures 2, 340300
Lecture – 2023W
Univ.-Prof. Dr. Alois Ferscha, teaching@pervasive.jku.at

Institute of
Pervasive Computing

**JOHANNES KEPLER
UNIVERSITY LINZ**
Altenberger Straße 69
4040 Linz, Austria
jku.at

Flow Analysis

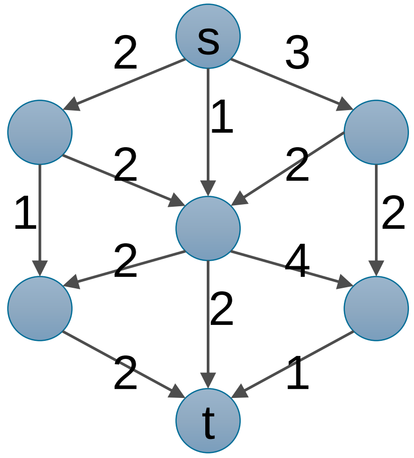
Flows in Networks

Consider a directed, weighted graph, also called **network *N***, where:

The weights are defined as **capacity** of the edge

In the graph there is one distinct vertex, which has **no incoming** edges, the **source *s***

In the graph there is one distinct vertex which has **no outgoing** edges, the **sink *t***



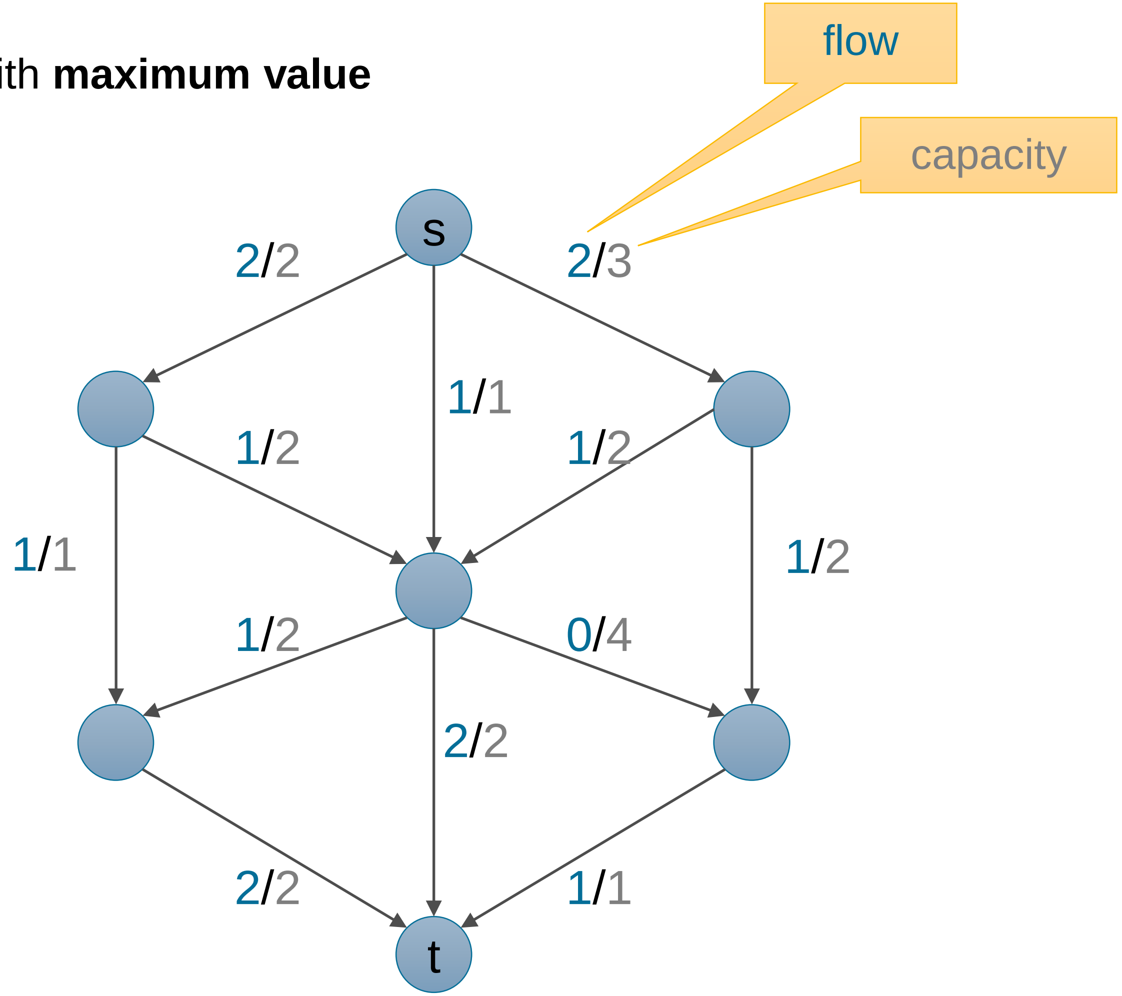
A **flow(u,v)** $\forall$ edge(u,v) in this graph is a function on the edges with

- **0** $\leq$ flow(u,v) $\leq$ **capacity(u,v)** (**Capacity constraint**)
- $\Sigma_{\mathbf{u \in in(v)}}$ flow(u,v) = $\Sigma_{\mathbf{w \in out(v)}}$ flow(v,w) (**Flow conservation**)
- |**flow**| = $\Sigma_{w \in out(s)}$ flow(s,w) = $\Sigma_{u \in in(t)}$ flow(u,t) (**Value of the flow**)

Maximum Flow

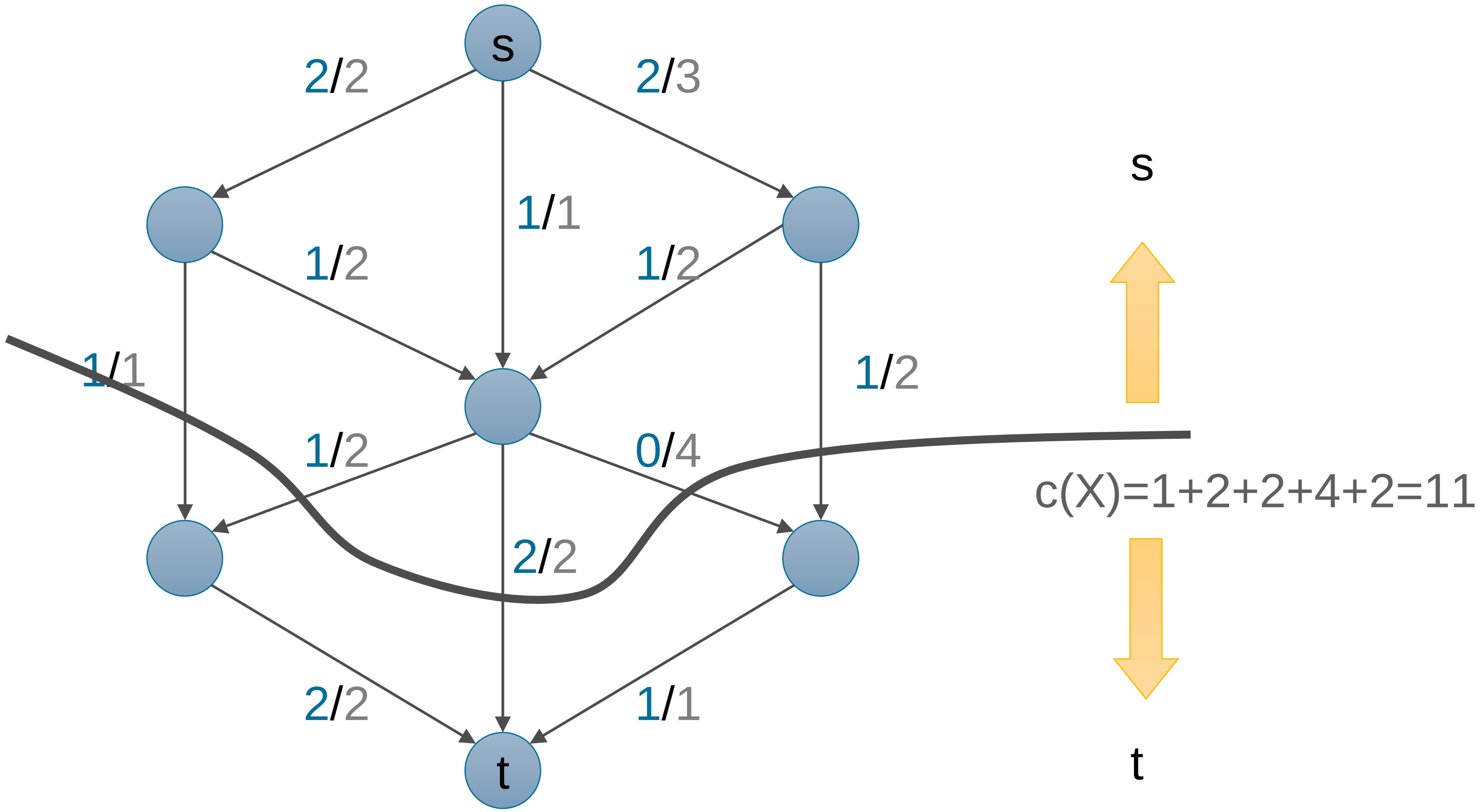
Find the **flow *f*** in a given network *N* with **maximum value**

Example: maxFlow = 5



Maximum flow

Maximum value of the flow is not only determined by *s* and *t*,
but also by each **cut** that **separates *s* from *t*** : **cut**



Cuts in Networks

A **cut $X = (V_s, V_t)$** is a cut through the network that **separates** the set of vertices into **two partitions**.

The **capacity** of a cut is the **sum of the capacities of the "cut" edges**

$$c(X) = \sum_{v \in V_s, w \in V_t} capacity(v, w)$$

We have:

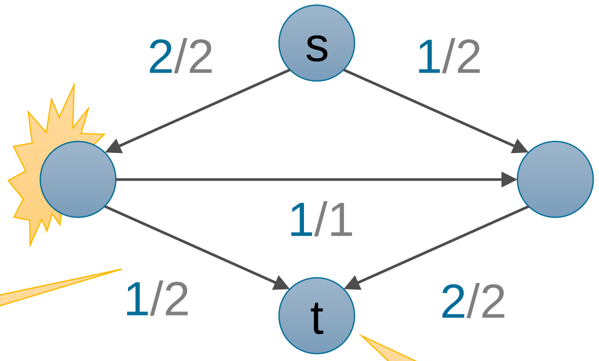
Value of the **maximum flow** = **capacity** of the **minimum cut**

Determination of the minimum cut:

Determine V_s as the set of vertices that are reached on the **augmenting path** in the network, and V_t as the set of **all other vertices** (variant of the **Ford-Fulkerson algorithm**)

Maximum Flow

min cut = 4



Maximum Flow Theorem (Ford/Fulkerson 1956)

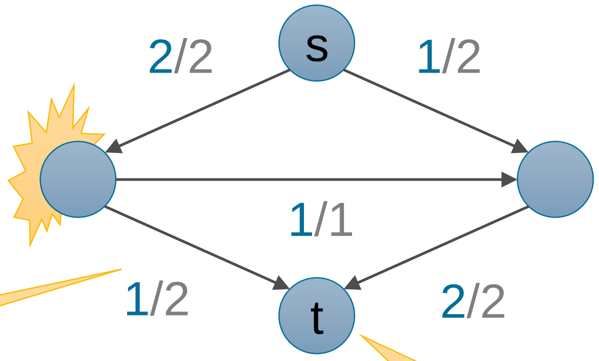
max flow ⇔ min cut

Each path s->t contains at least one **saturated** edge

flow = 3

Maximum Flow

min cut = 4



flow = 3

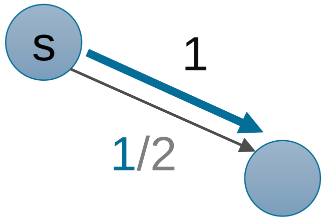
Maximum Flow Theorem (Ford/Fulkerson 1956)

max flow ⇔ min cut

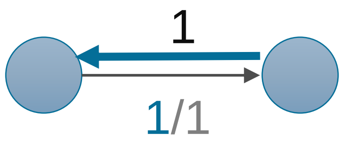
Each path s->t contains at least one **saturated** edge

Definition: augmenting path

- **Forward-edge**:
  flow(u,v) < capacity(u,v)
  → Flow can be **increased**

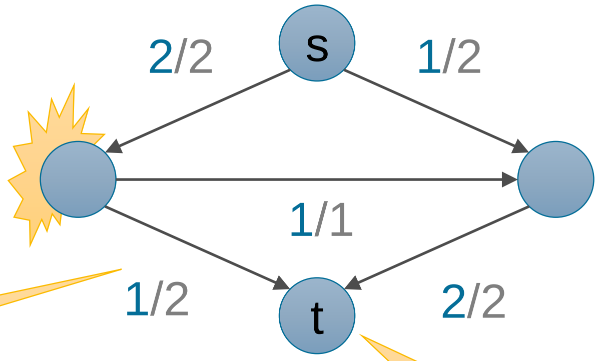


- **Backward-edge**:
  flow(u,v) > 0
  → Flow can be **decreased**



Maximum Flow

min cut = 4



flow = 3

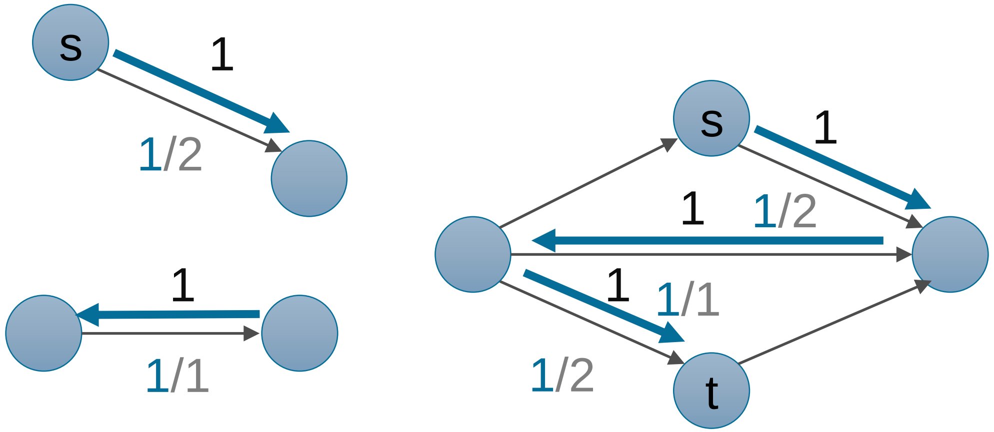
Maximum Flow Theorem (Ford/Fulkerson 1956)

The **flow** in a network is at **maximum**, if and only if the network has **no augmenting path**.

Each path s->t contains at least one **saturated** edge

Definition: augmenting path

- **Forward-edge**:
  flow(*u*,*v*) < capacity(*u*,*v*)
  → Flow can be **increased**
- **Backward-edge**:
  flow(*u*,*v*) > 0
  → Flow can be **decreased**



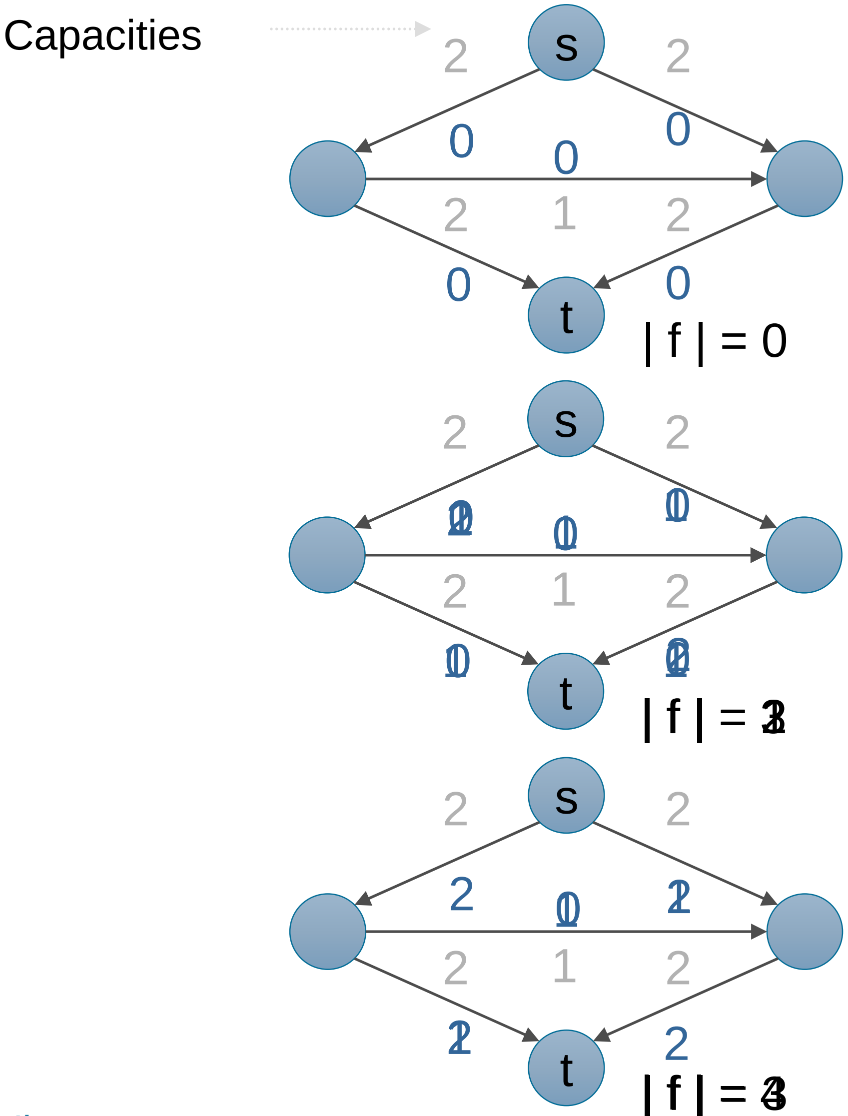
Ford-Fulkerson Algorithm

```
initialize network with null flow;

Method FindFlow
    if augmenting paths exist then
        find augmenting path;
        increase flow;
        recursive call to FindFlow;
```

A path s → t
(regardless of the direction of the arrow)
on which you can increase the flow is
called *augmenting path*

Ford-Fulkerson :: Example



Initialize flow with zero

Increase flow by 1 unit

Increase flow by 1 unit

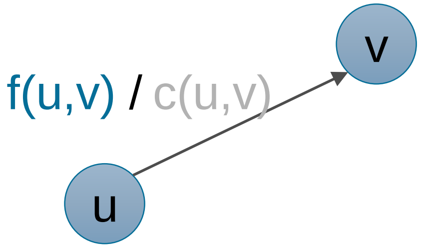
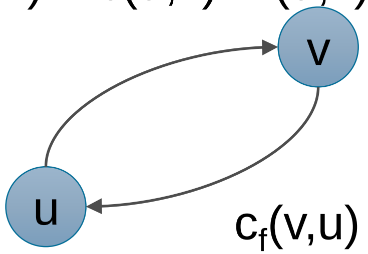
Increase flow by 1 unit

Send further unit over augmented path

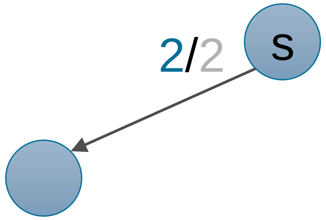
Determination of the Augmenting Path

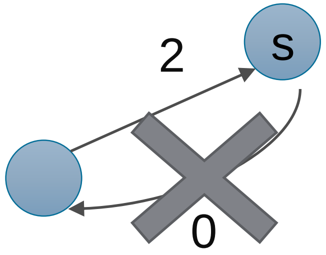
„Residual graph“: describes all possibilities to increase the flow

Consider the **Residual Network** $N_f = (V, Ef, cf, s, t)$ to a network $N = (V, E, c, s, t)$

N	N_f
	$c_f(u,v) = c(u,v) - f(u,v)$  $c_f(v,u) = f(u,v)$ Edges with capacity 0 are removed

An **augmented path** in N corresponds to a directed path from s to t in N_f and can therefore be determined using DFS in N_f



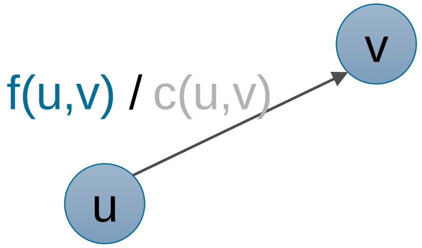


Determination of the Augmenting Path

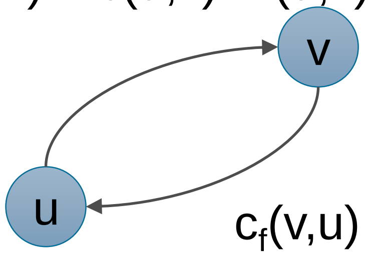
„Residual graph“: describes all possibilities to increase the flow

Consider the **Residual Network** $N_f = (V, Ef, cf, s, t)$ to a network $N = (V, E, c, s, t)$

N | N_f



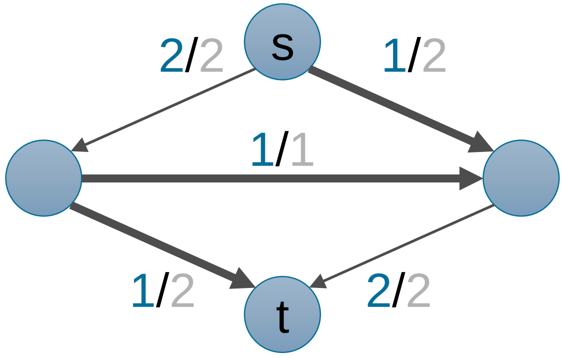
$c_f(u,v) = c(u,v) - f(u,v)$

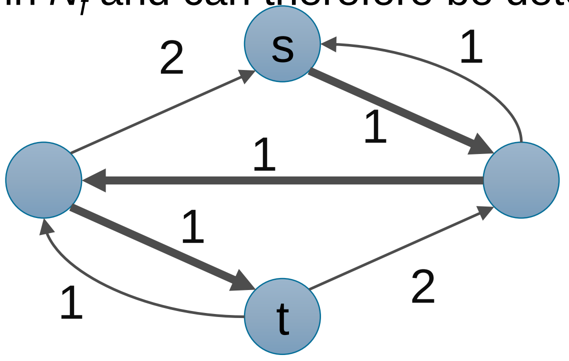


$c_f(v,u) = f(u,v)$

Edges with capacity 0 are removed

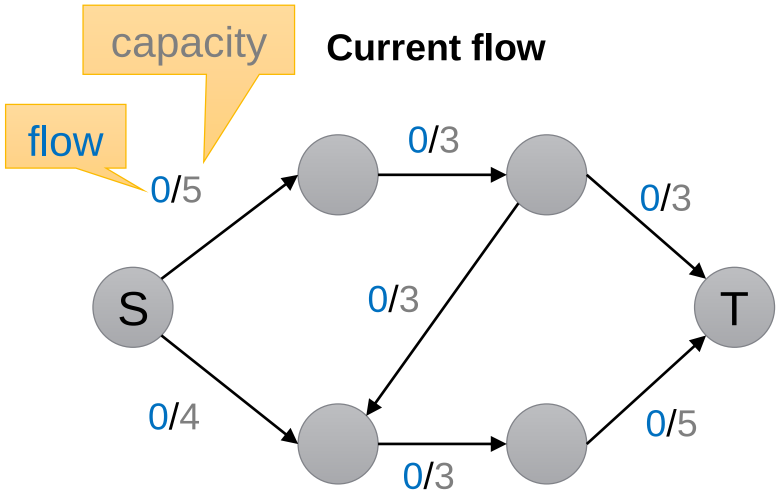
An **augmented path** in *N* corresponds to a directed path from *s* to *t* in N_f and can therefore be determined using DFS in N_f



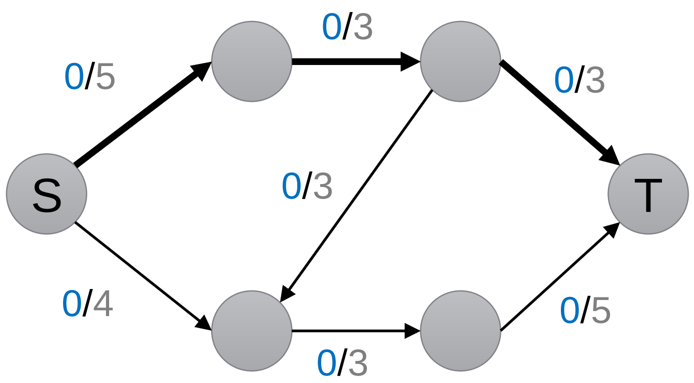


AUGMENTED PATH :: EXAMPLE STEP 1/7

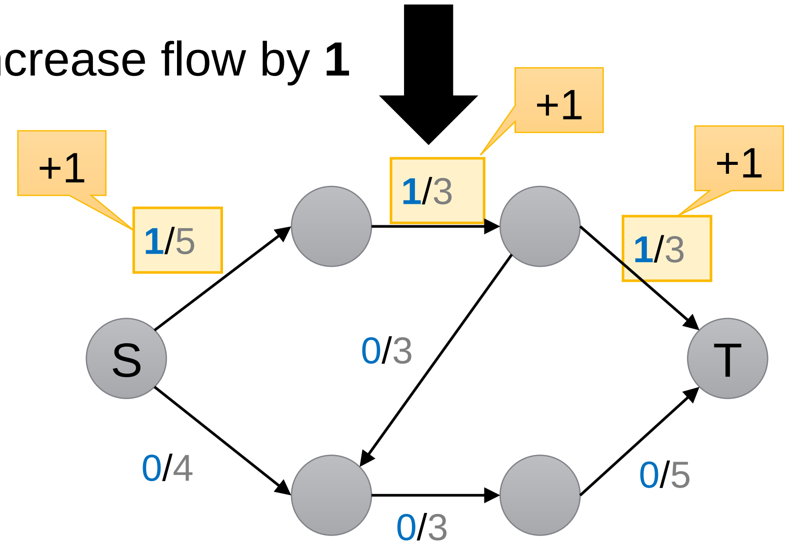
Current flow



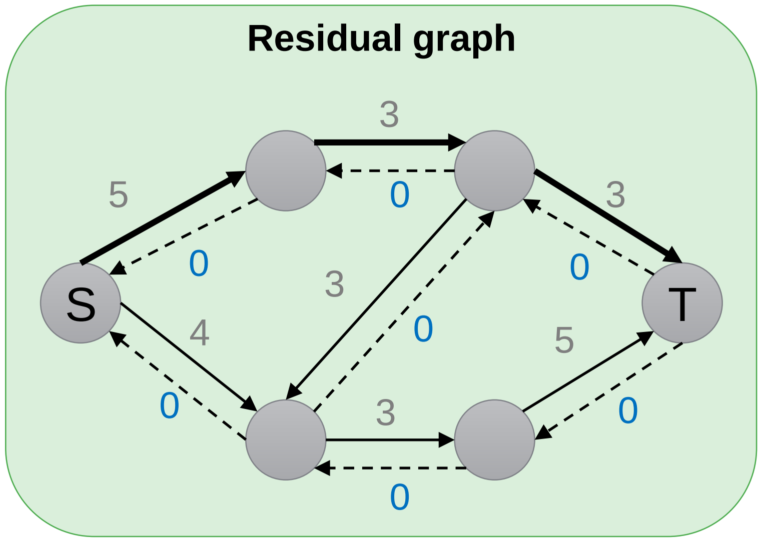
Augmenting path



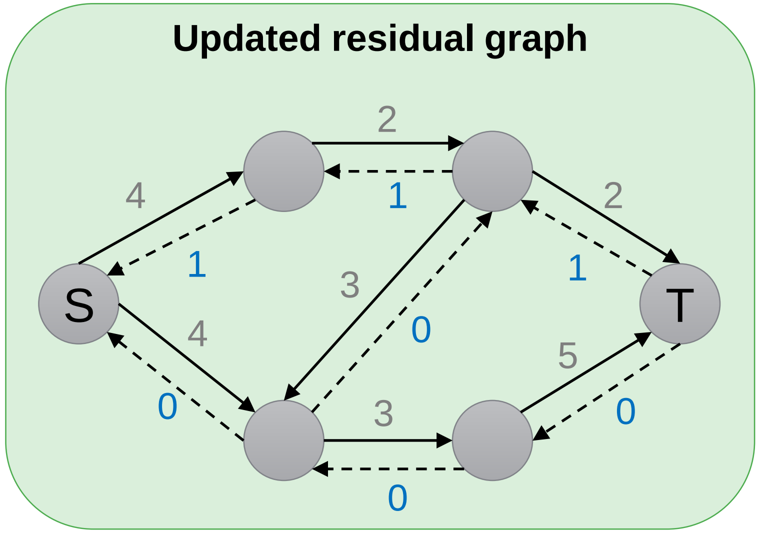
increase flow by **1**



Residual graph



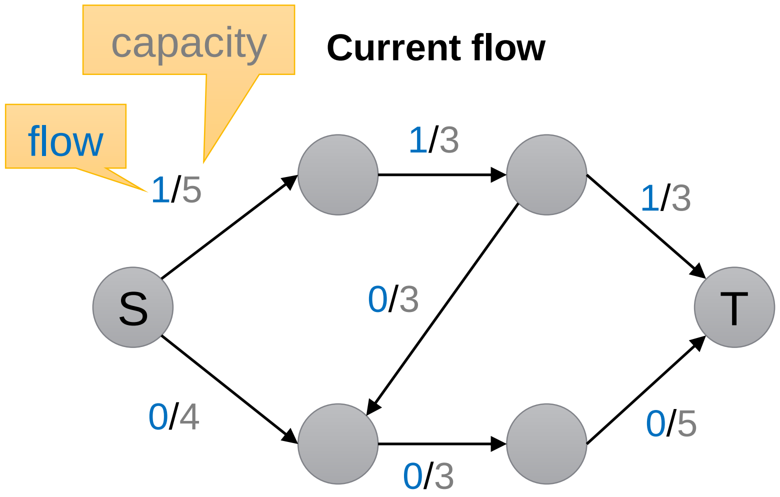
Updated residual graph



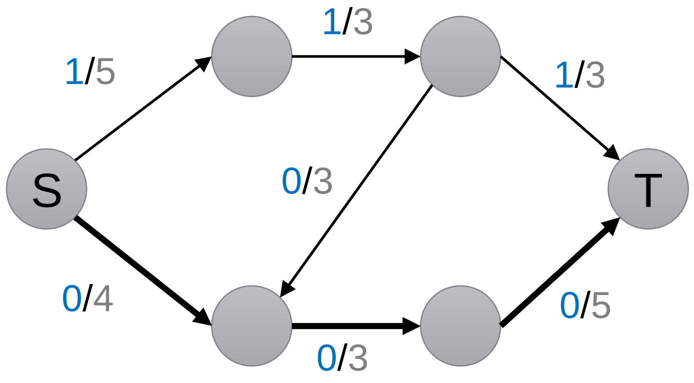
Institute of
Pervasive Computing

AUGMENTED PATH :: EXAMPLE STEP 2/7

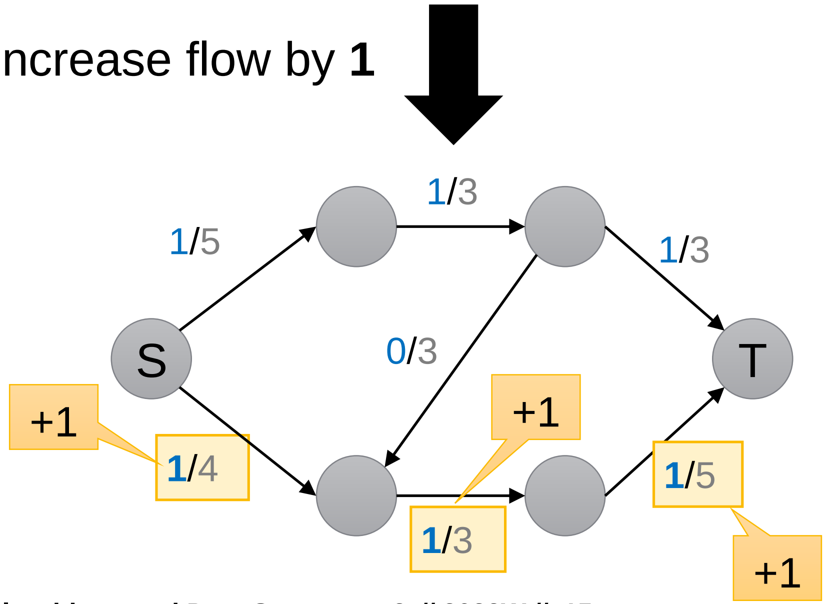
Current flow



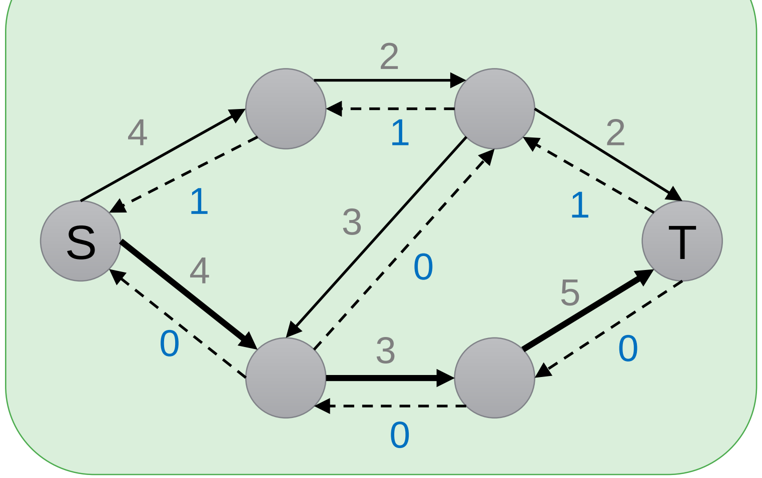
Augmenting path



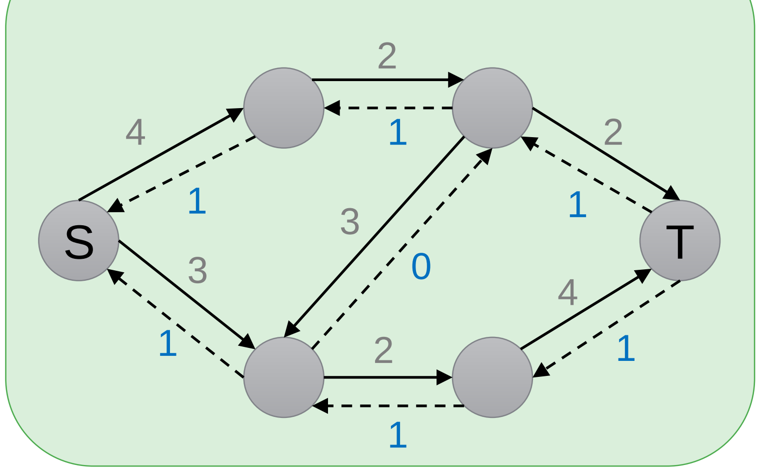
increase flow by **1**



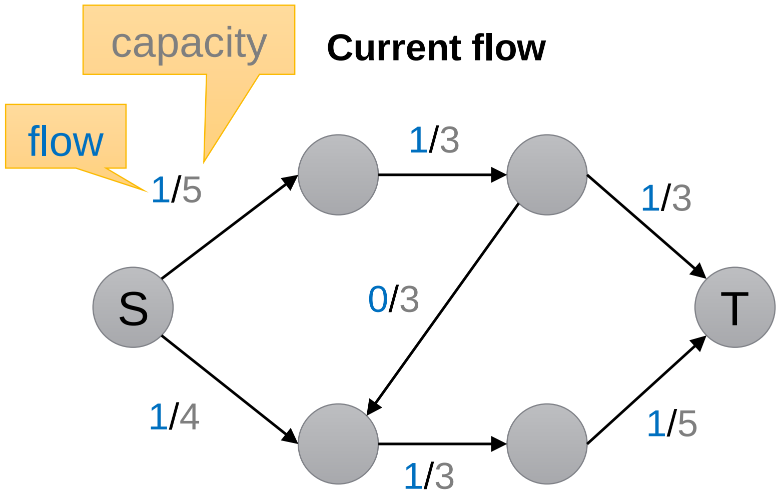
Residual graph

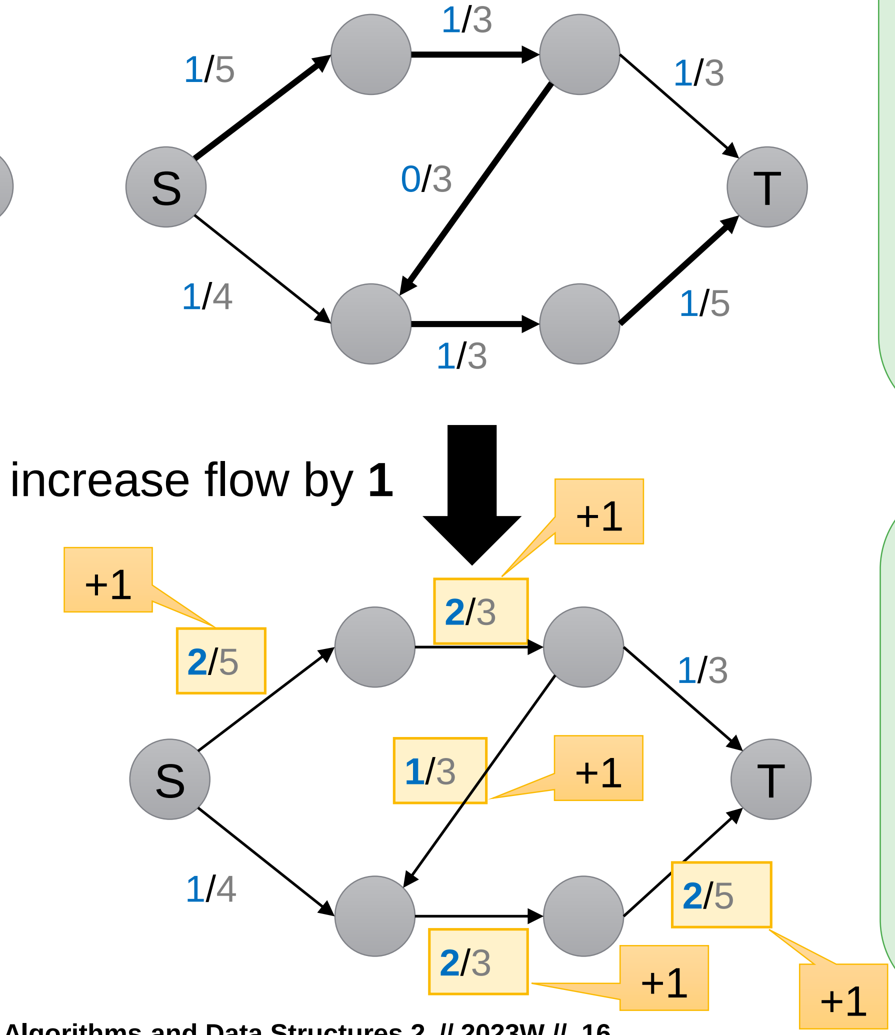


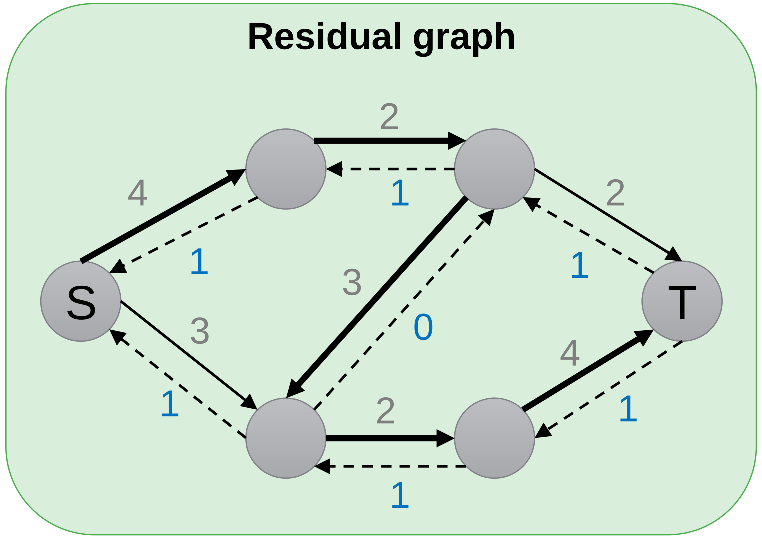
Updated residual graph

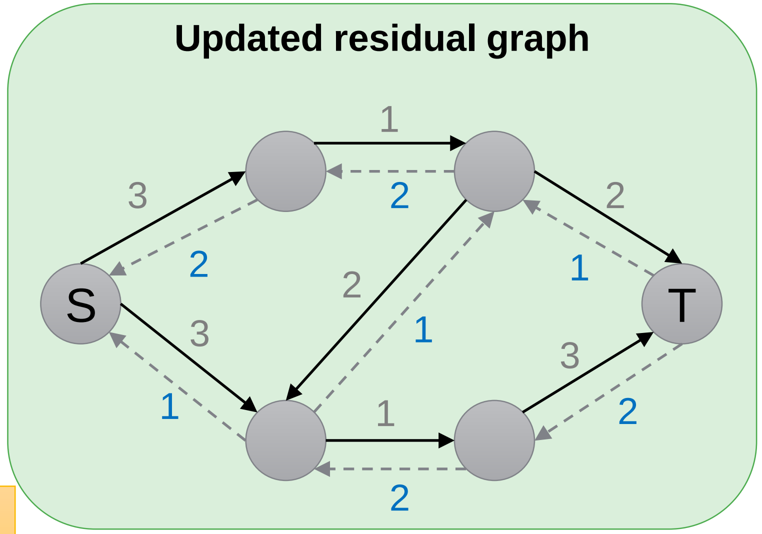


AUGMENTED PATH :: EXAMPLE STEP 3/7



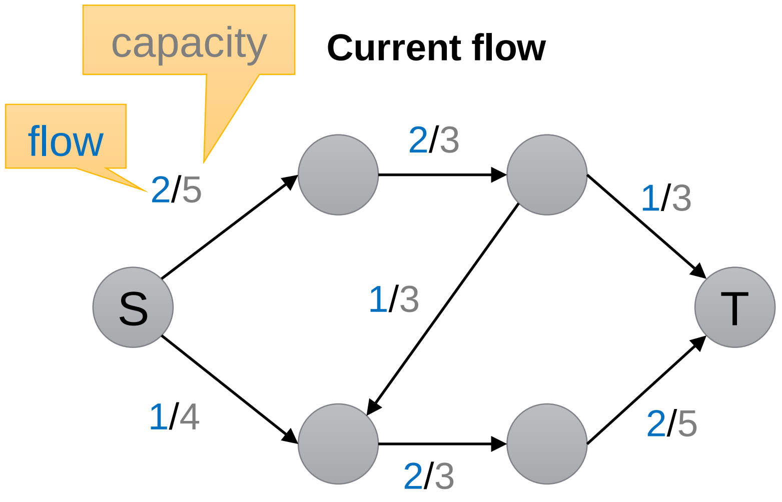




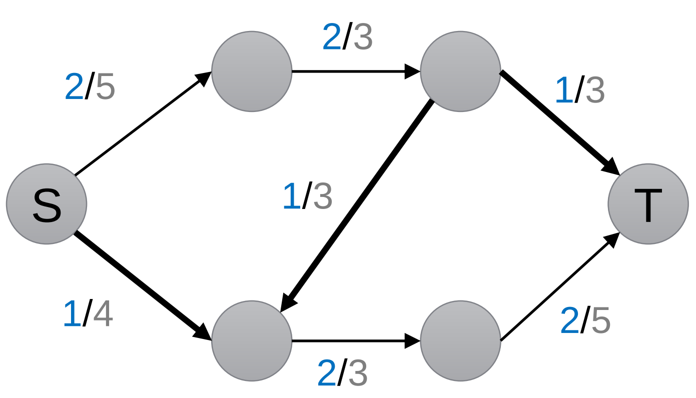


AUGMENTED PATH :: EXAMPLE STEP 4/7

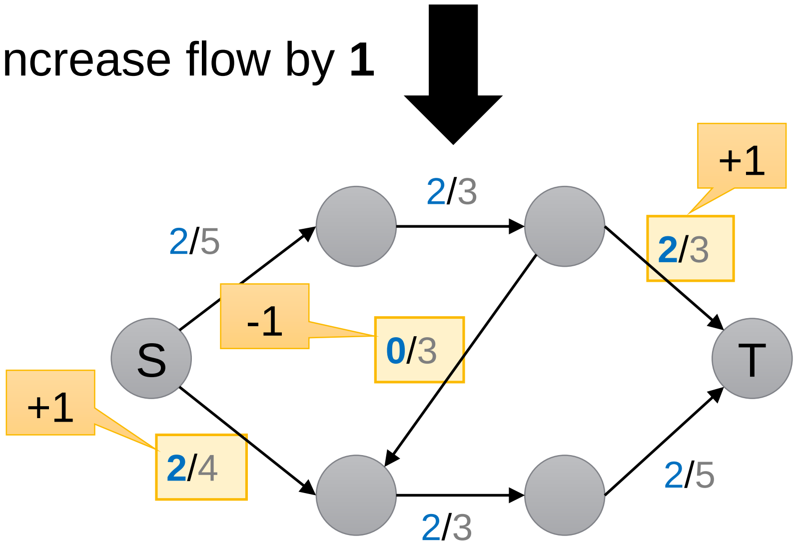
Current flow



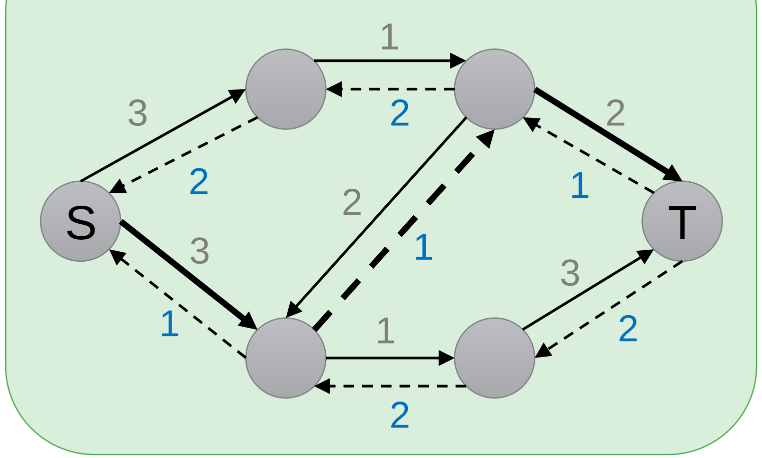
Augmenting path



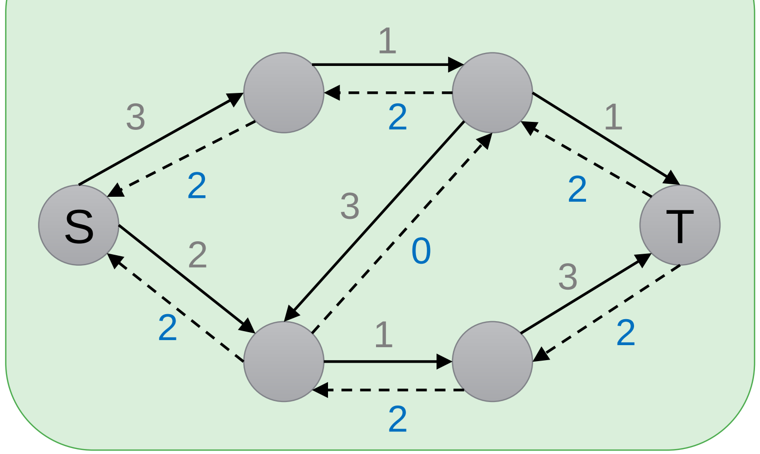
increase flow by **1**



Residual graph

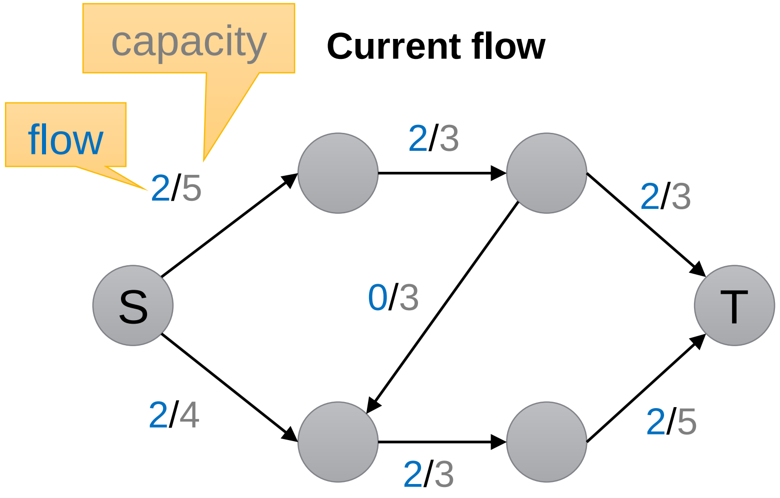


Updated residual graph

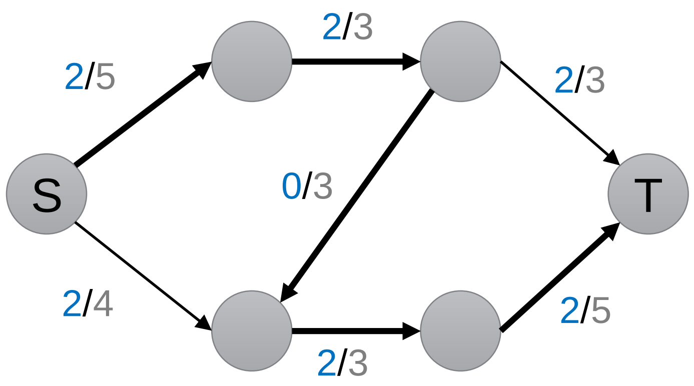


AUGMENTED PATH :: EXAMPLE STEP 5/7

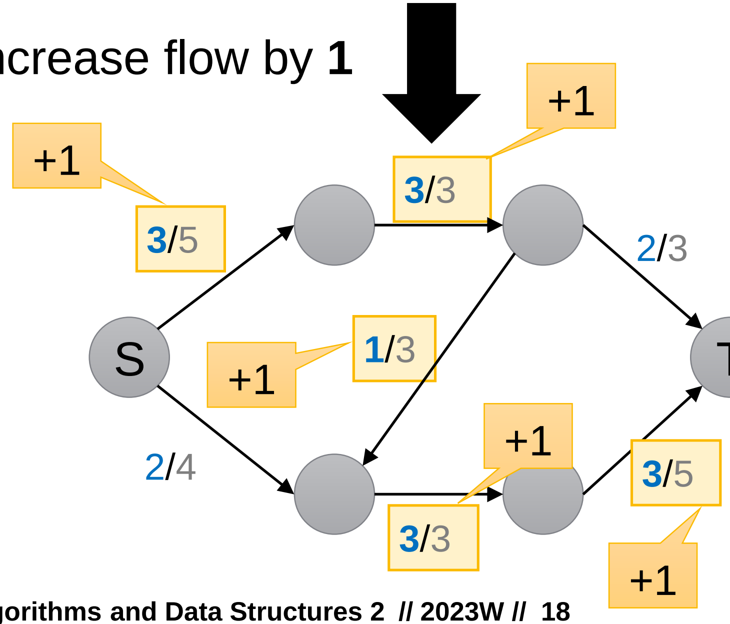
Current flow

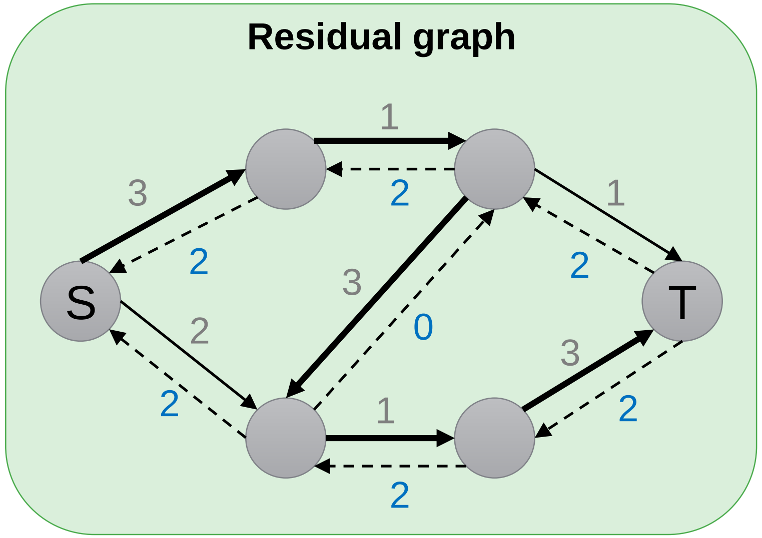


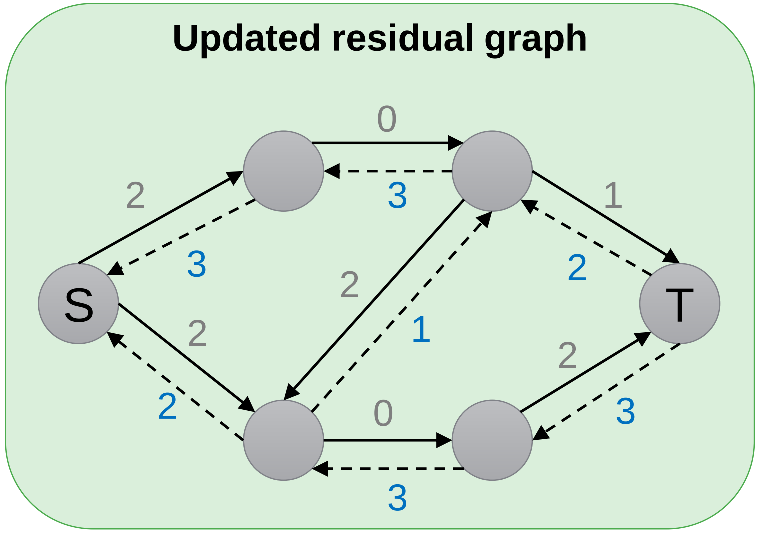
Augmenting path



increase flow by **1**



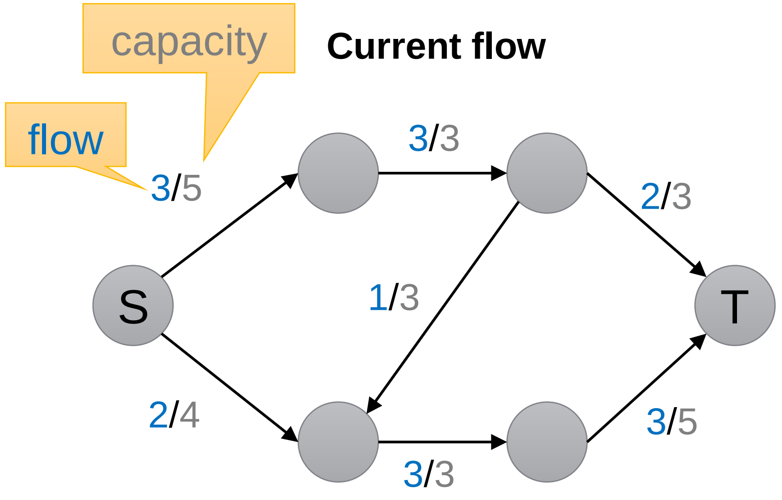


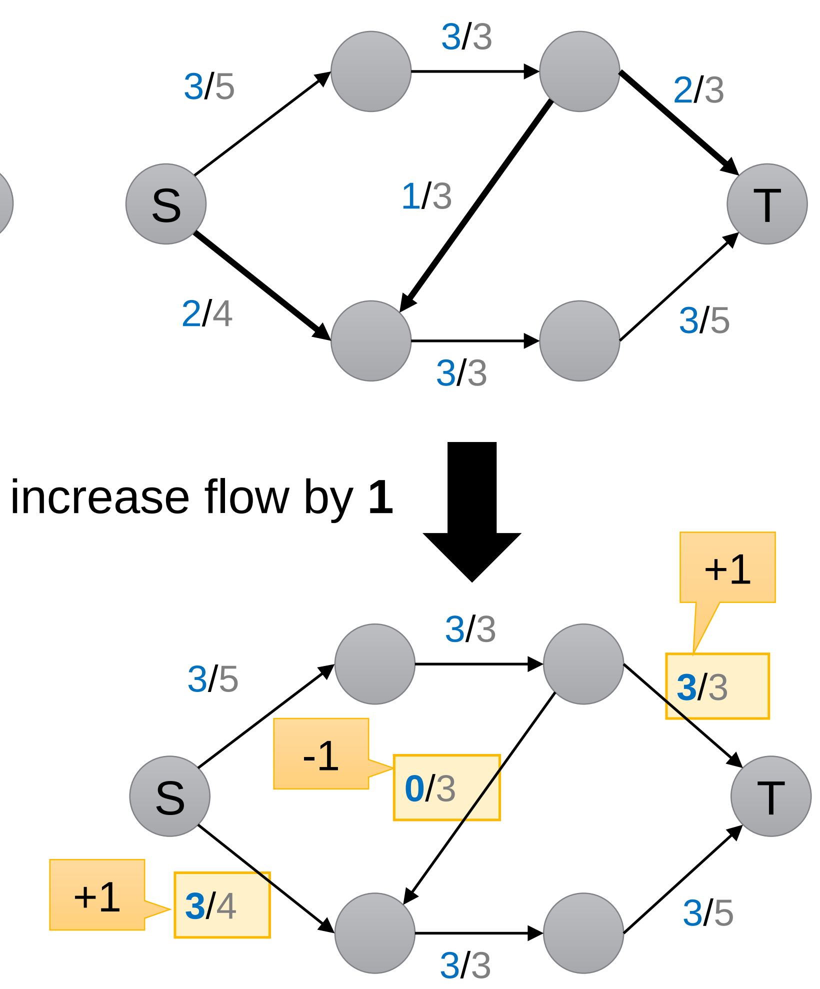


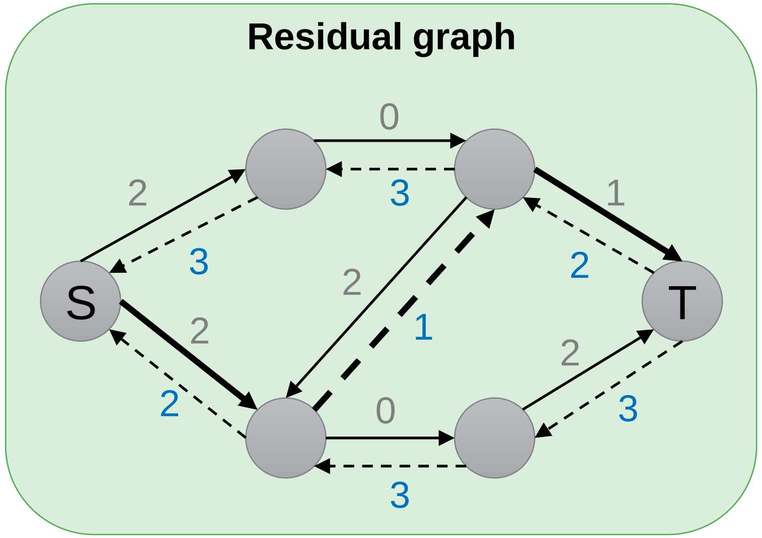
Institute of Pervasive Computing

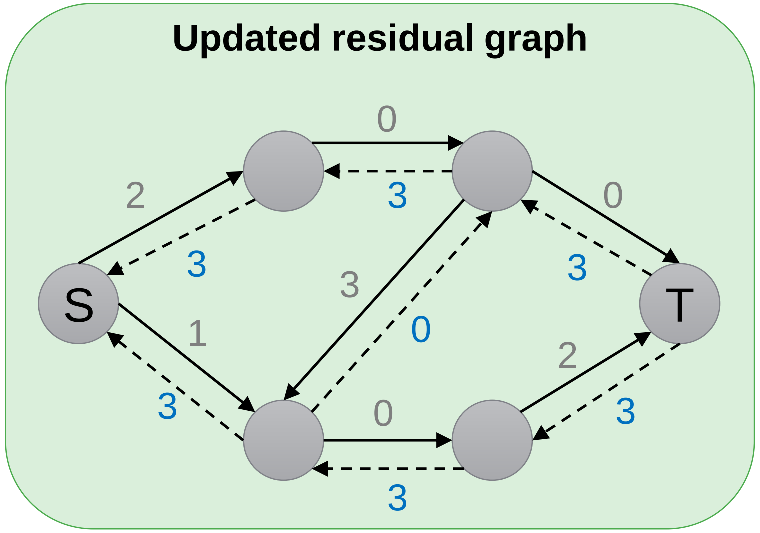
A. Ferscha

AUGMENTED PATH :: EXAMPLE STEP 6/7



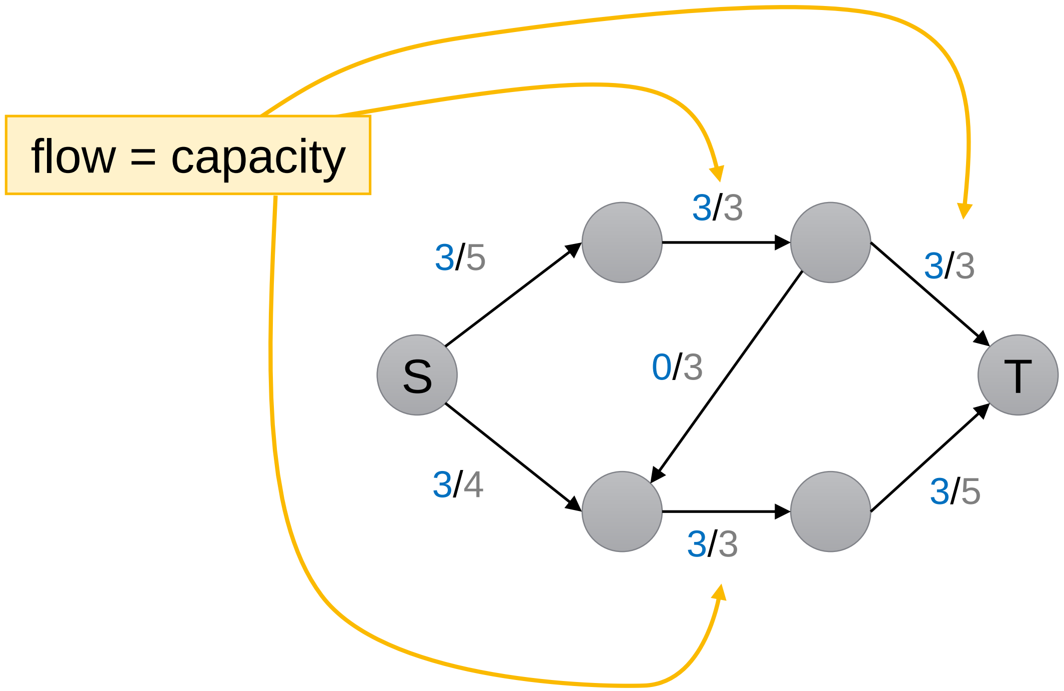


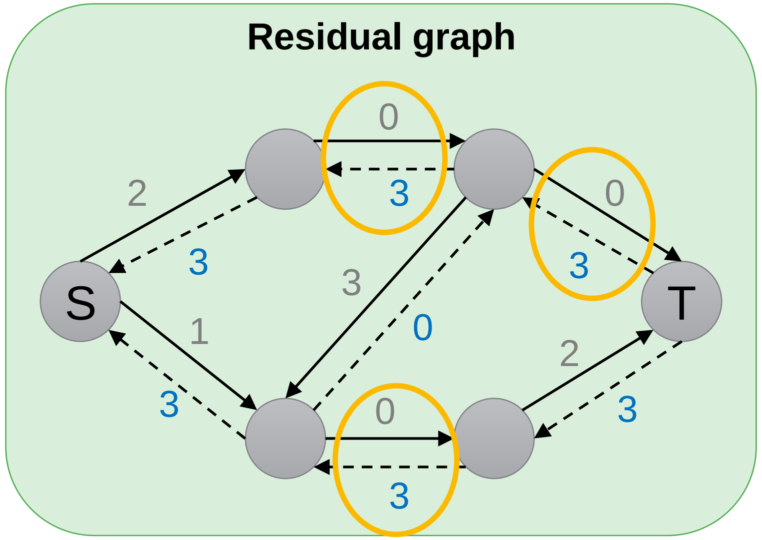




AUGMENTED PATH :: EXAMPLE STEP 7/7

No augmenting path from source S to sink T exists where flow could be increased (when flow = capacity).

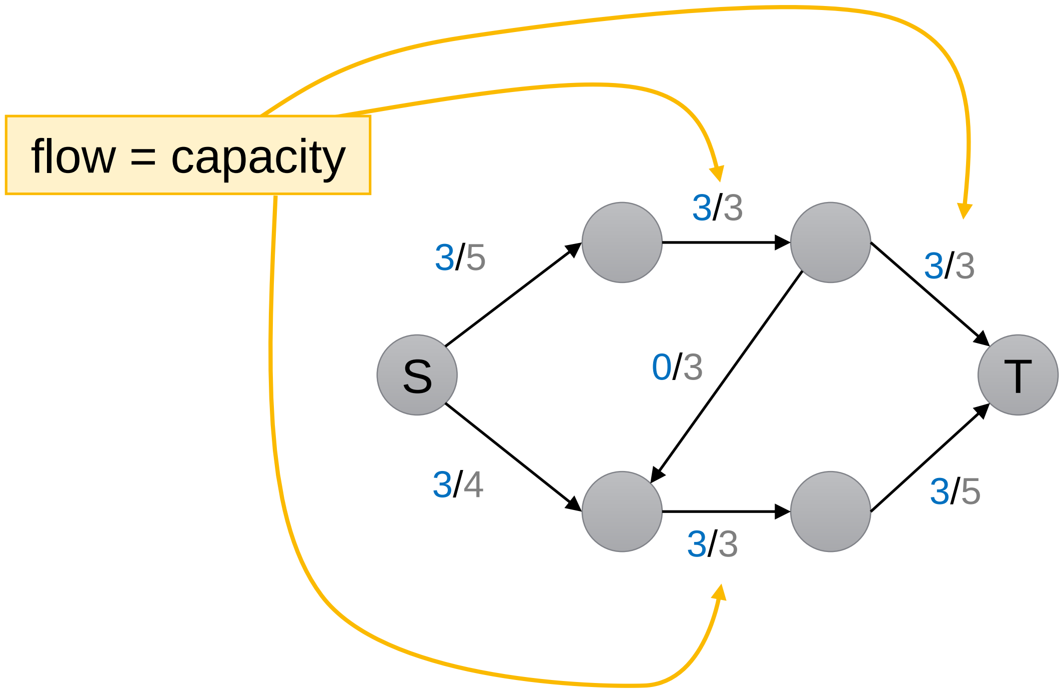


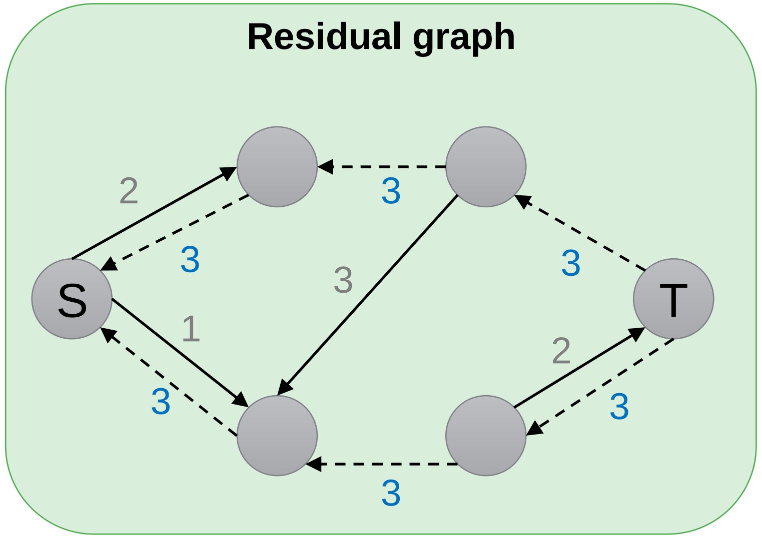


Max flow is: **|f| = 6**

AUGMENTED PATH :: EXAMPLE STEP 7/7

No augmenting path from source S to sink T exists where flow could be increased (when flow = capacity).





Max flow is: **|f| = 6**

Maximum Flow Algorithm

Part I: Setup

```
Algorithm: MaxFlow(N)
      Input: network N
      Output: network N_f with maximum flow
Start with null flow:
f(u,v) ← 0 ∀ (u,v) ∈ E;
Initialize residual network:
N_f ← N;
```

Part II: Loop

```
repeat
      search for directed path p in N_f from s to t
      if (path p found)
            D_f ← min {c_f (u,v), f(u,v) ∈ p};
            for (each (u,v) ∈ p) do
                  if (forward (u,v))
                        f(u,v) ← f(u,v) + D_f ;
                  if (backward (u,v))
                        f(u,v) ← f(u,v) - D_f ;
            update N_f ;
until (no augmenting path exists);
```

Runtime
$O(F \cdot (n+m))$

Improvement of Maximum Flow

Theorem [Edmonds & Karp 1972]

By using **BFS** the maximum flow can be determine in a **runtime** of

$$O((n+m) \cdot n \cdot m) = O(n^5)$$

Advantage:

Runtime is **independent from** the **value** of the **maximum flow**.

Graphs
Part: Flows

Algorithms and Data Structures 2, 340300
Lecture – 2023W
Univ.-Prof. Dr. Alois Ferscha, teaching@pervasive.jku.at

Institute of
Pervasive Computing

**JOHANNES KEPLER
UNIVERSITY LINZ**
Altenberger Straße 69
4040 Linz, Austria
jku.at