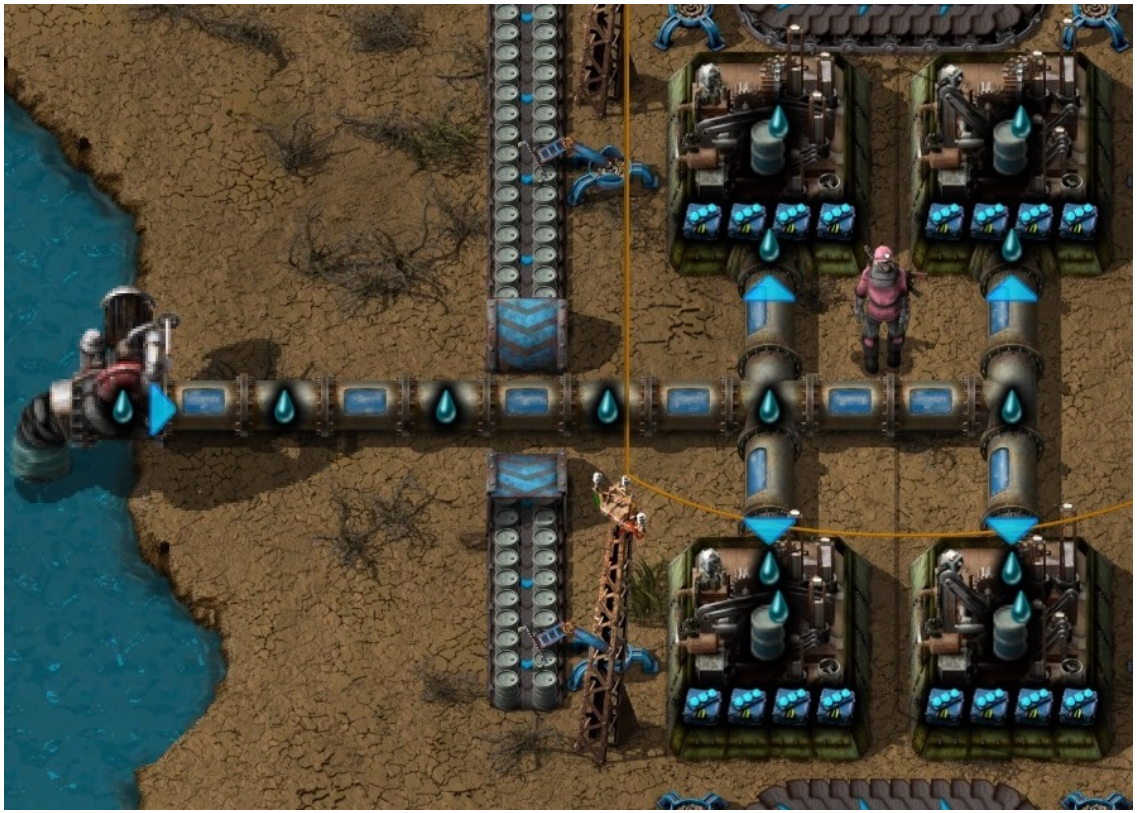
Applications of Ford-Fulkerson COMS20010 (Algorithms II)

John Lapinskas, University of Bristol

Circulations: Multiple sources and sinks

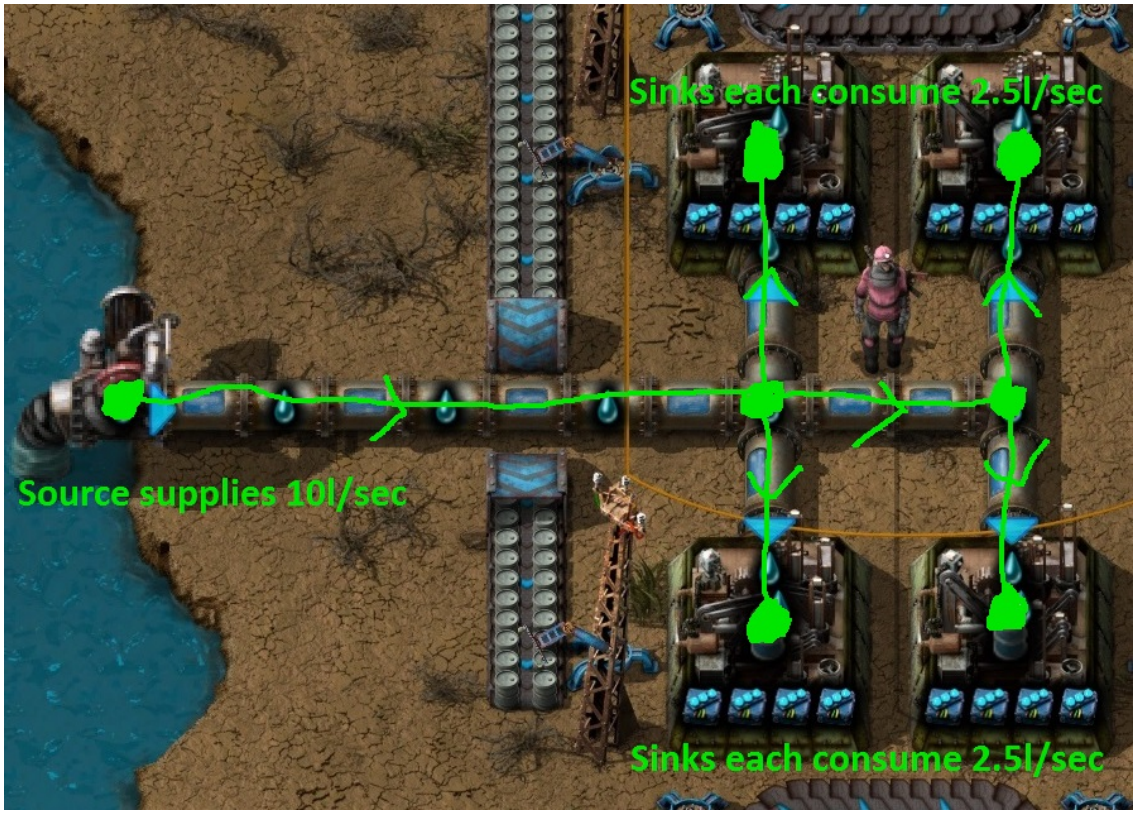
One of our simplifying assumptions was that a flow network only has one source and one sink. But in e.g. water networks, this is often not true.



We can model situations like these as **circulations**, in which we allow **every** node to be a source or sink.

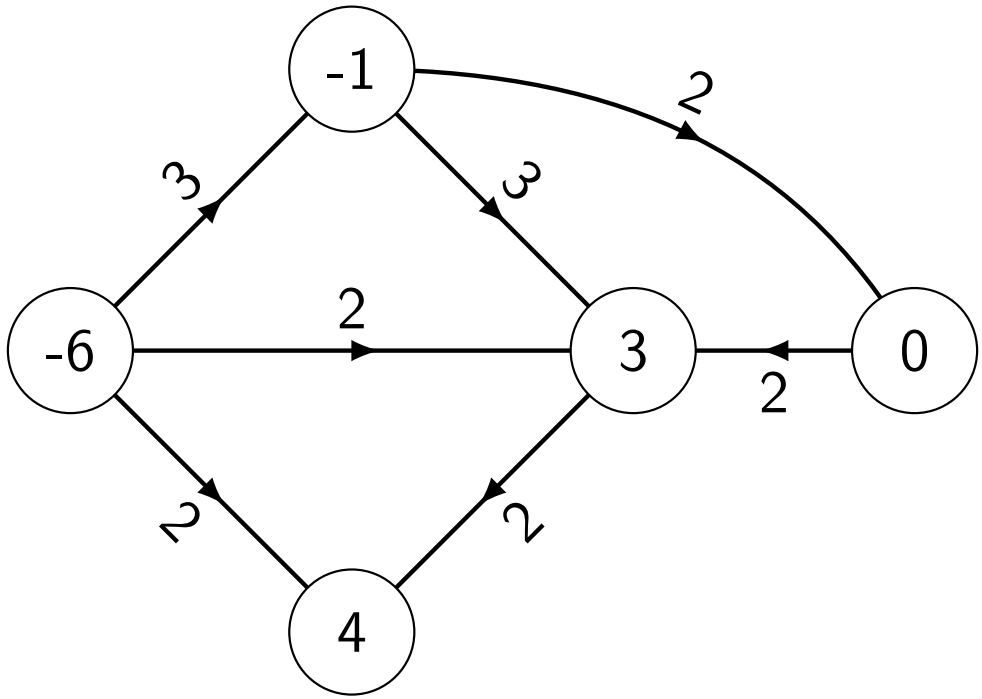
Circulations: Multiple sources and sinks

One of our simplifying assumptions was that a flow network only has one source and one sink. But in e.g. water networks, this is often not true.



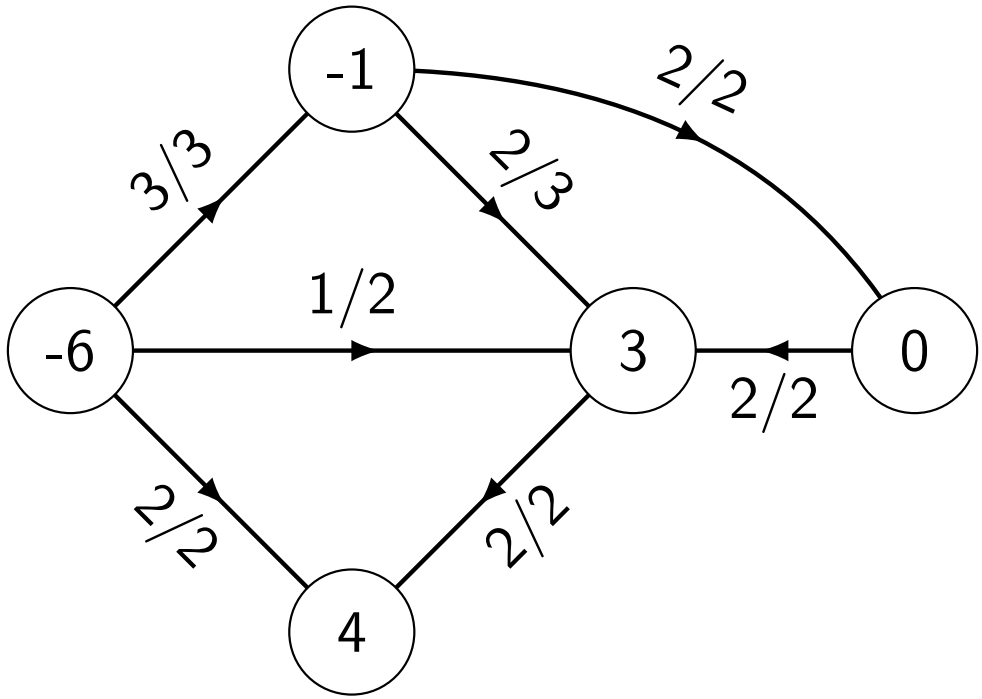
Instead of maximising flow, we specify how much each source supplies and how much each sink consumes, and try to satisfy these requirements.

A **circulation network** (G, c, D) is a directed graph $G = (V, E)$, a capacity function $c\colon E \to \mathbb{N}$, and a **demand** function $D\colon V \to \mathbb{Z}$.



A vertex v with demand $D(v) > 0$ is a **sink**.
A vertex v with demand $D(v) < 0$ is a **source**.

A **circulation network** (G, c, D) is a directed graph $G = (V, E)$, a capacity function $c\colon E \to \mathbb{N}$, and a **demand** function $D\colon V \to \mathbb{Z}$.



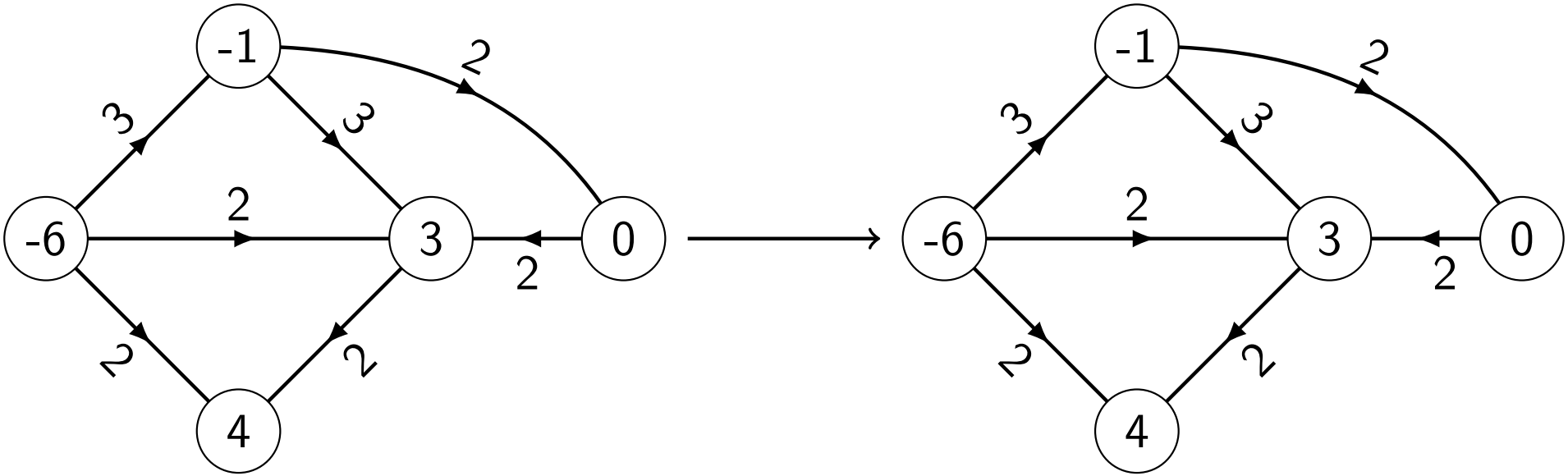
A vertex v with demand $D(v) > 0$ is a **sink**.
A vertex v with demand $D(v) < 0$ is a **source**.

A **circulation** is a function $f\colon E \to \mathbb{R}$ with $0 \leq f(e) \leq c(e)$ for all $e \in E$, and $f^-(v) - f^+(v) = D(v)$ (not zero!) for all $v \in V$. Note flow is conserved except at sources and sinks.

Our problem: Does a circulation exist? Here: yes!

Finding circulations

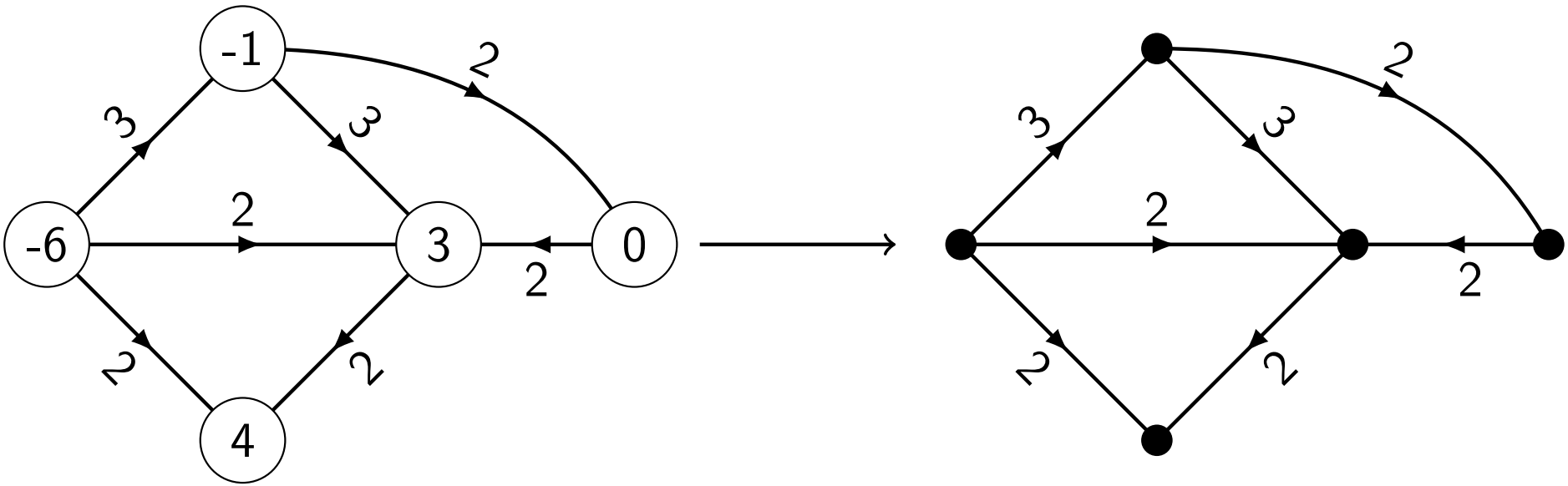
Again, we can reduce to finding a maximum flow in a flow network.



Write $S = \{\text{sources}\}$, $T = \{\text{sinks}\}$.
Then transform our input circulation network into a flow network:

Finding circulations

Again, we can reduce to finding a maximum flow in a flow network.

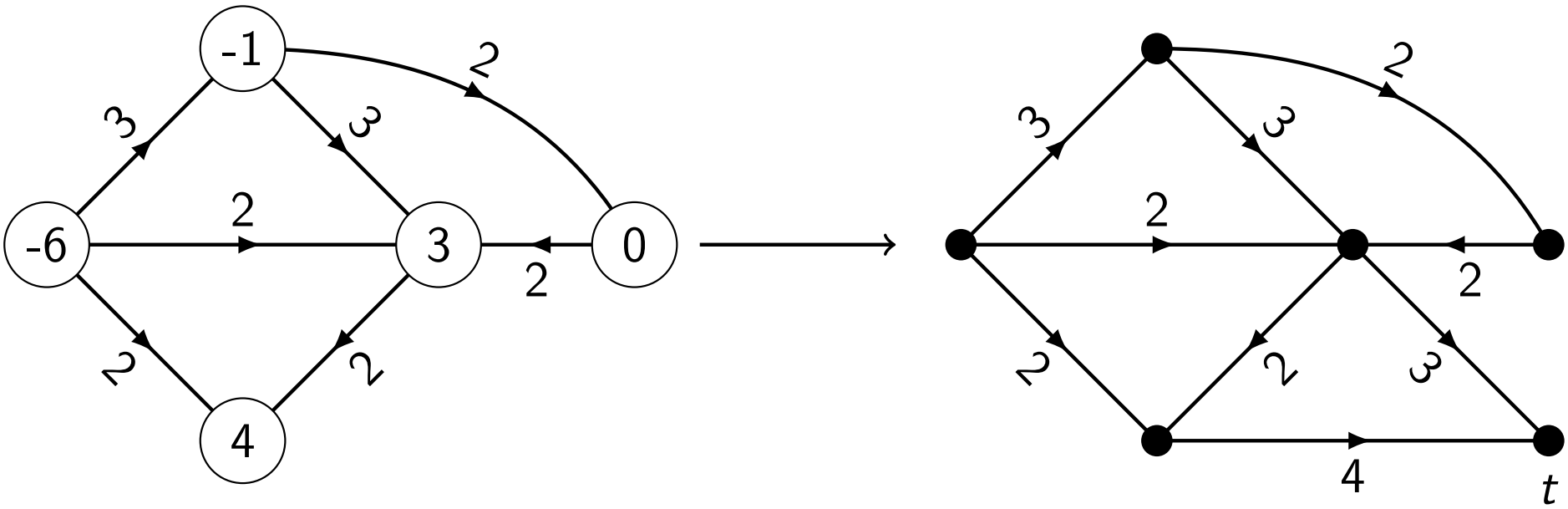


Write $S = \{\text{sources}\}$, $T = \{\text{sinks}\}$.
Then transform our input circulation network into a flow network:

- Remove the demand functions.

Finding circulations

Again, we can reduce to finding a maximum flow in a flow network.

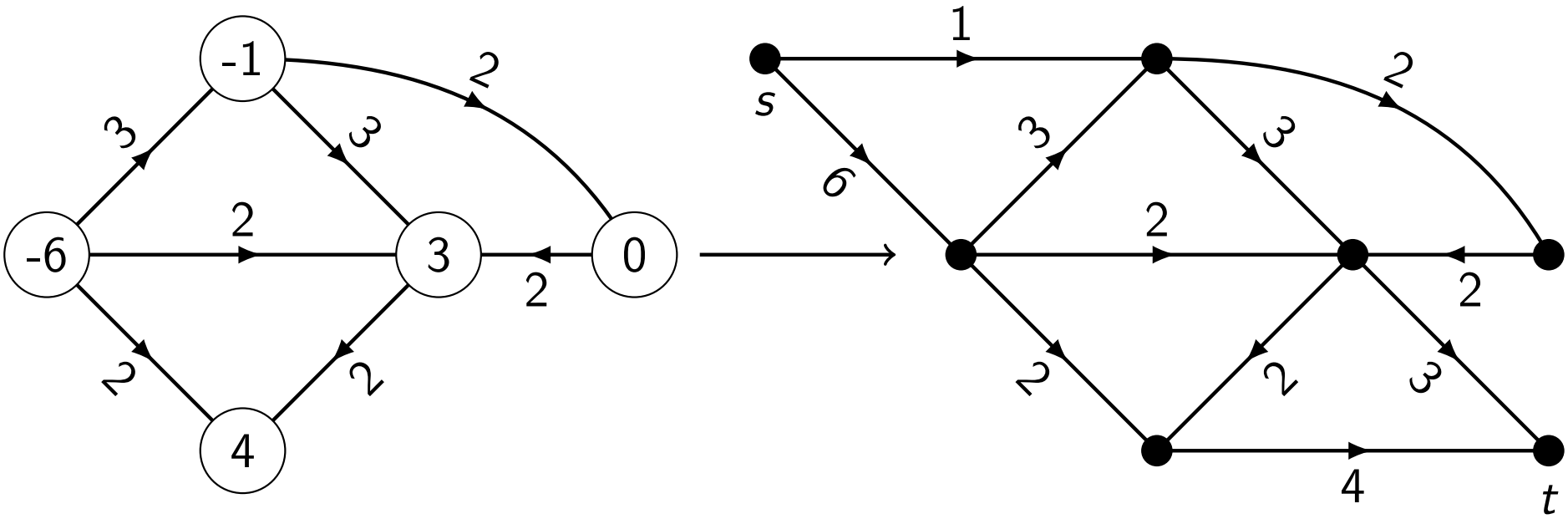


Write $S = \{\text{sources}\}$, $T = \{\text{sinks}\}$.
Then transform our input circulation network into a flow network:

- Remove the demand functions.
- Add a new vertex t and add every possible edge $T \to t$.
  Give each edge (v, t) capacity $D(v)$.

Finding circulations

Again, we can reduce to finding a maximum flow in a flow network.

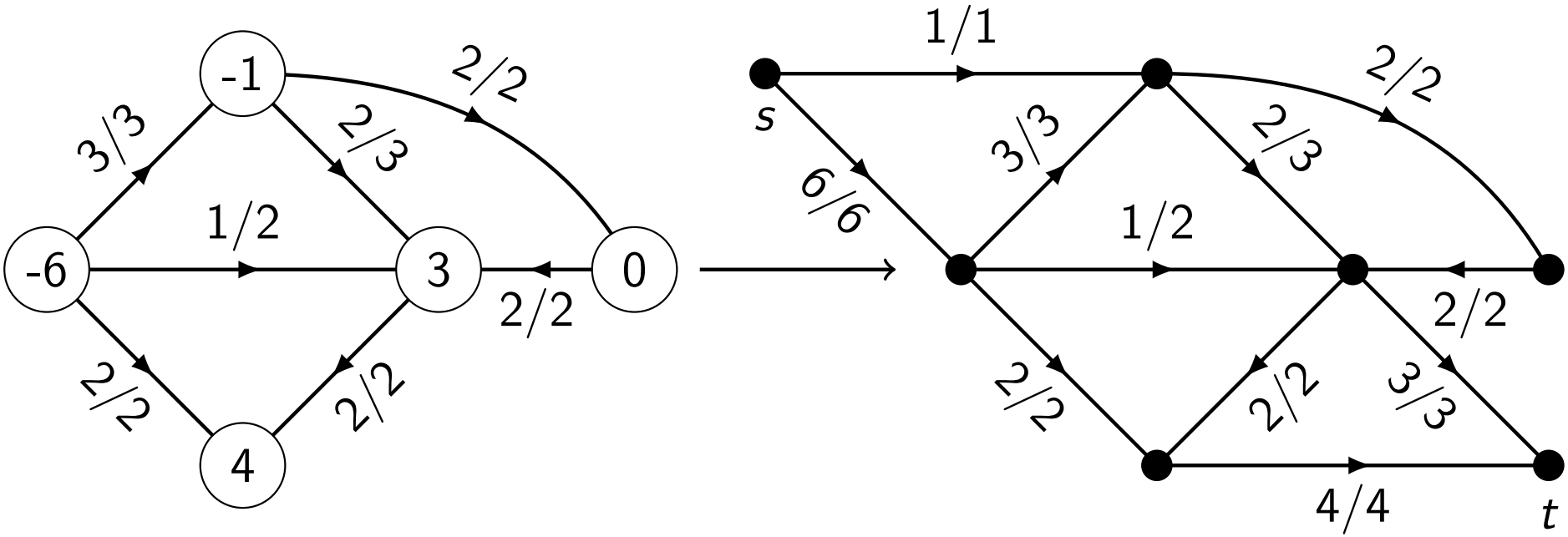


Write $S = \{\text{sources}\}$, $T = \{\text{sinks}\}$.
Then transform our input circulation network into a flow network:

- Add a new vertex t and add every possible edge $T \to t$. Give each edge (v, t) capacity $D(v)$.
- Add a new vertex s and add every possible edge $s \to S$. Give each edge (s, v) capacity $-D(v)$.

Finding circulations

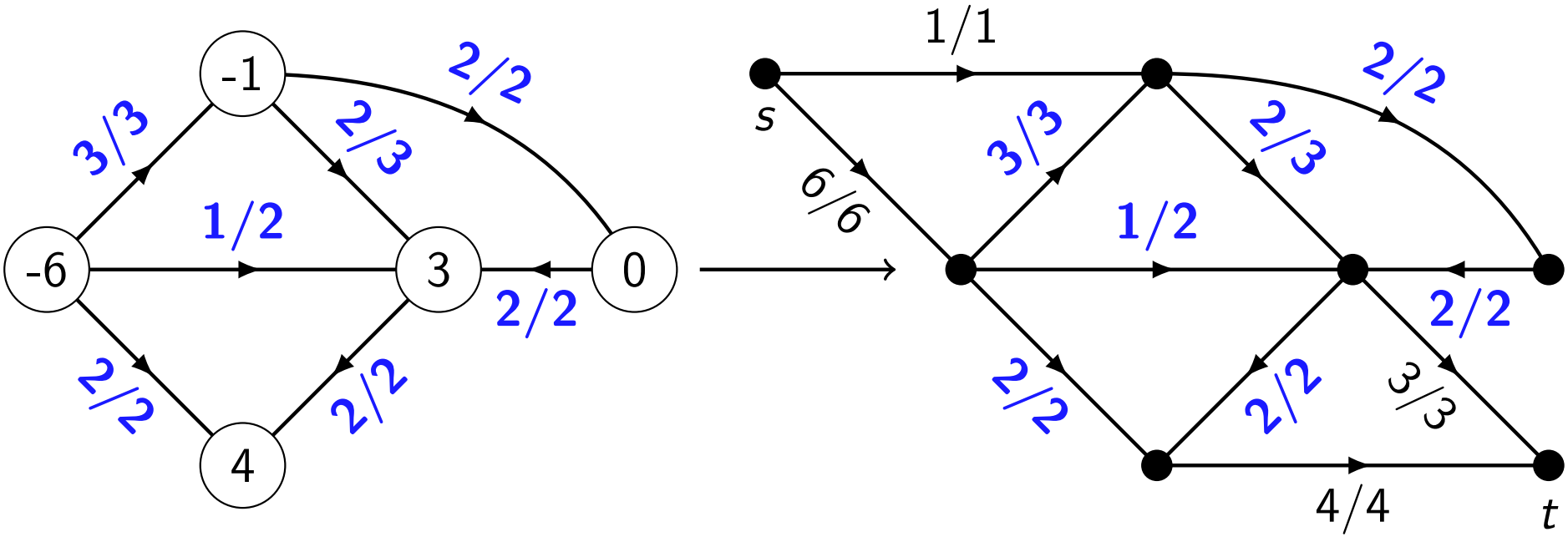
Again, we can reduce to finding a maximum flow in a flow network.



We claim every circulation on the left corresponds to a flow with value $\sum_{v \in T} D(v) = -\sum_{v \in S} D(v)$ on the right, and vice versa:

Finding circulations

Again, we can reduce to finding a maximum flow in a flow network.

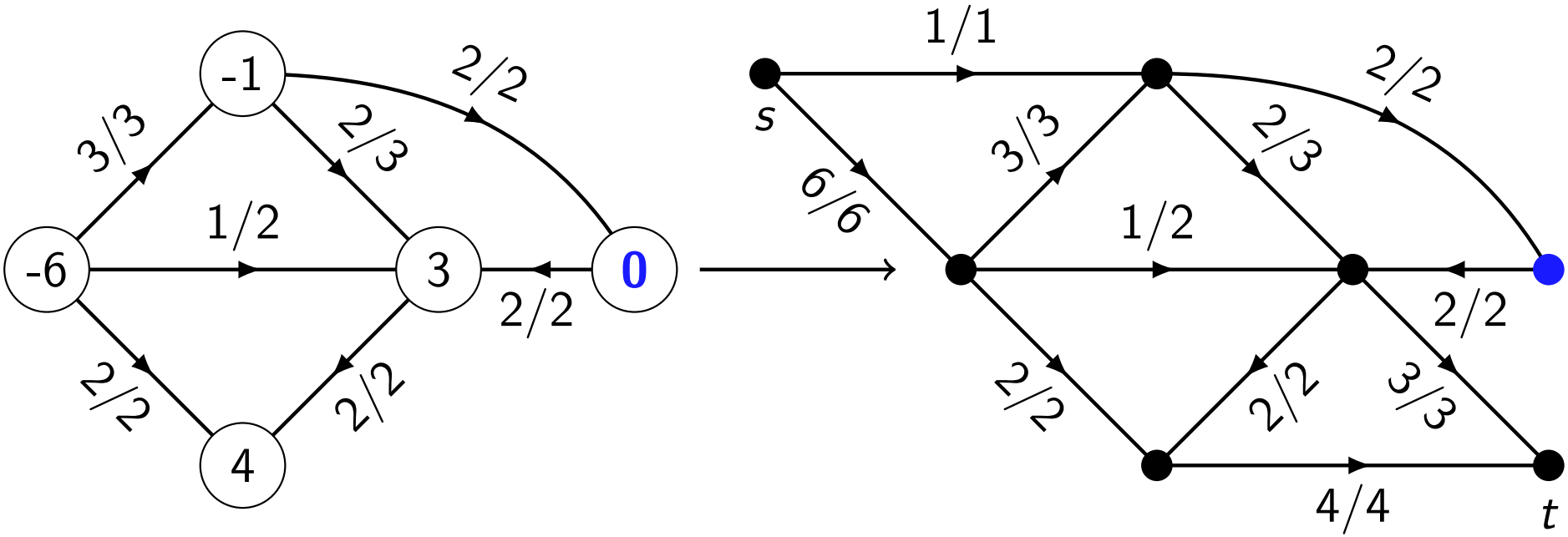


We claim every circulation on the left corresponds to a flow with value $\sum_{v \in T} D(v) = -\sum_{v \in S} D(v)$ on the right, and vice versa:

Capacity constraints on left $\Leftrightarrow$ Corresponding constraints on right

Finding circulations

Again, we can reduce to finding a maximum flow in a flow network.

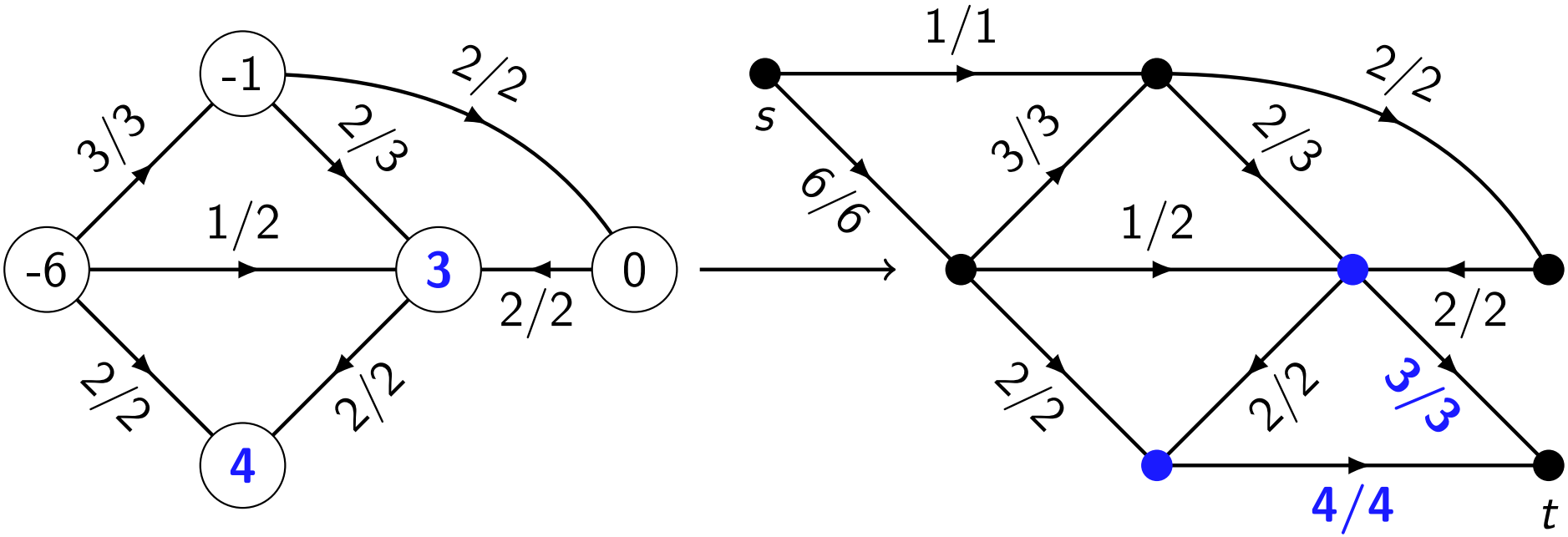


We claim every circulation on the left corresponds to a flow with value $\sum_{v \in T} D(v) = -\sum_{v \in S} D(v)$ on the right, and vice versa:

Demand satisfied outside $S \cup T \Leftrightarrow$ Flow conserved outside $S \cup T$

Finding circulations

Again, we can reduce to finding a maximum flow in a flow network.



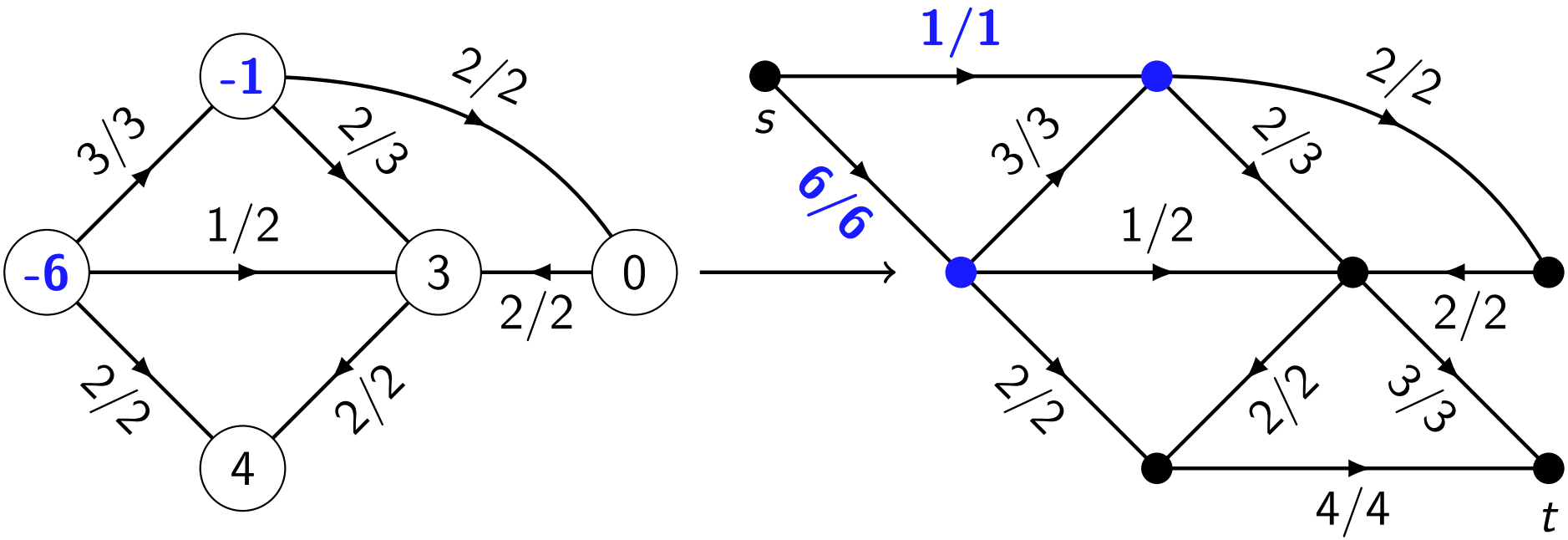
We claim every circulation on the left corresponds to a flow with value $\sum_{v \in T} D(v) = -\sum_{v \in S} D(v)$ on the right, and vice versa:

$$\text{Demand satisfied in } T \Leftrightarrow \begin{array}{l}\text{Flow conserved in } T \text{ and} \\ \text{flow value is } \sum_{v \in T} D(v)\end{array}$$

(Remember flow value can be taken at any cut; see Lemma 2 last lecture.)

Finding circulations

Again, we can reduce to finding a maximum flow in a flow network.

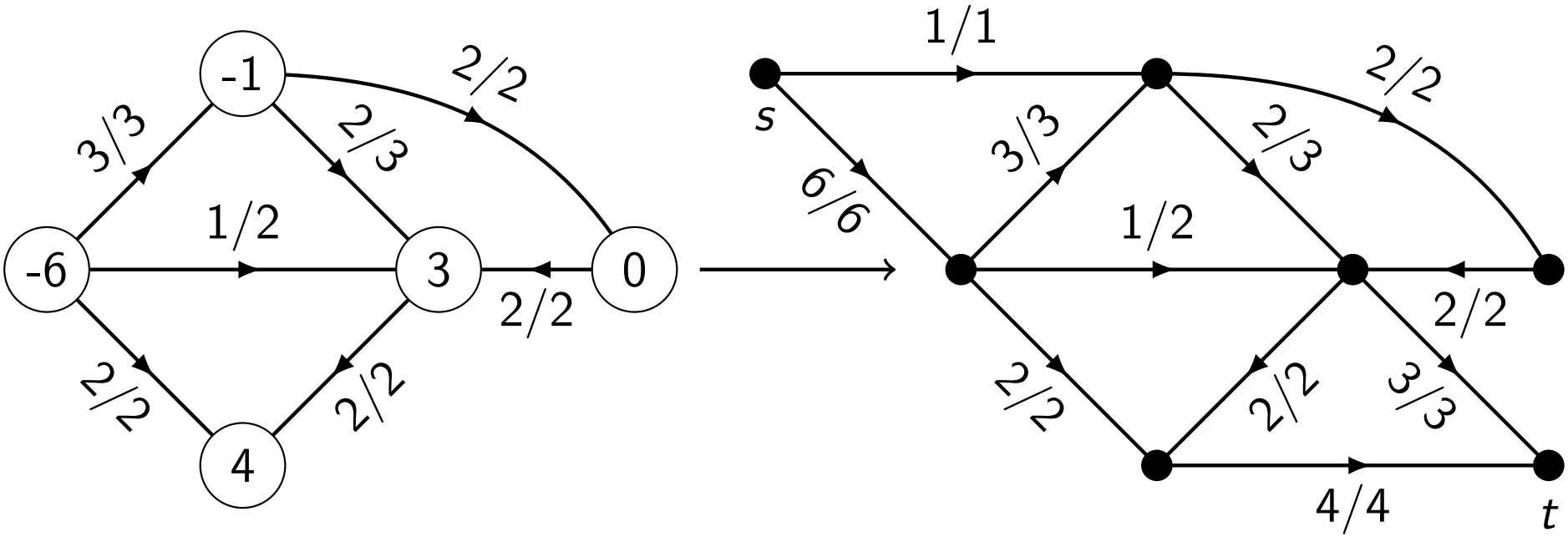


We claim every circulation on the left corresponds to a flow with value $\sum_{v \in T} D(v) = -\sum_{v \in S} D(v)$ on the right, and vice versa:

$$\text{Demand satisfied in } S \Leftrightarrow \begin{array}{c}\text{Flow conserved in } S \text{ and} \\ \text{flow value is } -\sum_{v \in S} D(v)\end{array}$$

Finding circulations

Again, we can reduce to finding a maximum flow in a flow network.

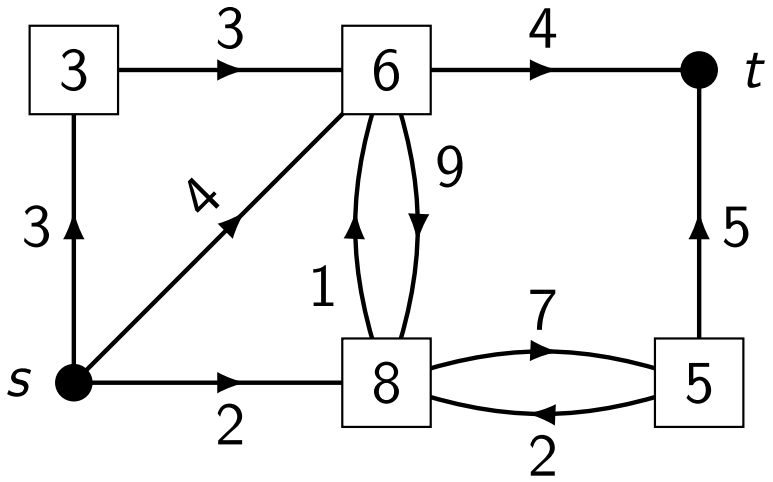


We claim every circulation on the left corresponds to a flow with value $\sum_{v \in T} D(v) = -\sum_{v \in S} D(v)$ on the right, and vice versa. □

The algorithm: Construct a flow network as above and run Edmonds-Karp. If it returns a flow with value $\sum_{v \in T} D(v) = -\sum_{v \in S} D(v)$, compute and return the corresponding circulation; otherwise, no circulation exists.

Adding vertex capacities

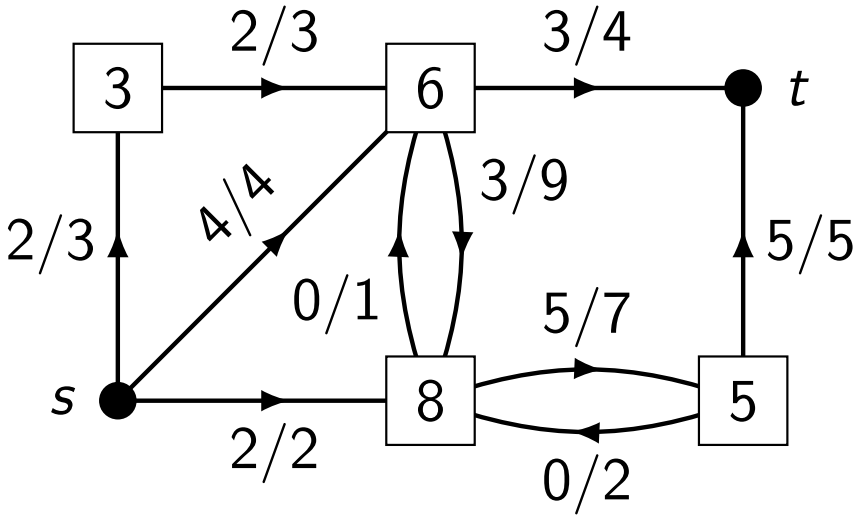
We've already put limits on the amount of flow that can pass through each edge of the network. What if we want to do the same for vertices? E.g. a pumping station might only be able to handle a limited amount of water.



A **vertex flow network** (G, c_E, c_V, s, t) is a flow network (G, c_E, s, t) combined with a **vertex capacity** function $c_V : V \setminus \{s, t\} \to \mathbb{N}$.

Adding vertex capacities

We've already put limits on the amount of flow that can pass through each edge of the network. What if we want to do the same for vertices? E.g. a pumping station might only be able to handle a limited amount of water.

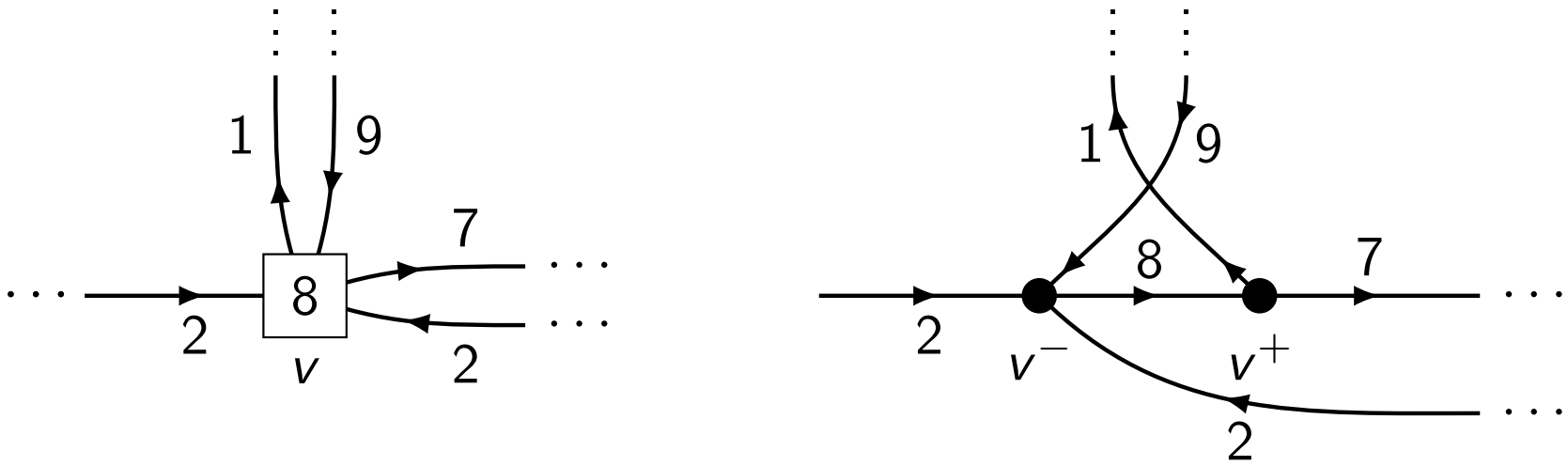


A **vertex flow network** (G, c_E, c_V, s, t) is a flow network (G, c_E, s, t) combined with a **vertex capacity** function $c_V : V \setminus \{s, t\} \to \mathbb{N}$.

A function $f : E \to \mathbb{R}$ is a **flow** if it is a flow in (G, c_E, s, t) and also, for all $v \in V(G)$, $0 \leq f^+(v) \leq c_V(v)$. (Note the first condition implies $f^+(v) = f^-(v)$.)

Again, we can reduce this to a flow network!

We use a design pattern called a **vertex gadget**: we simulate each vertex of the vertex flow network "independently" in the flow network.

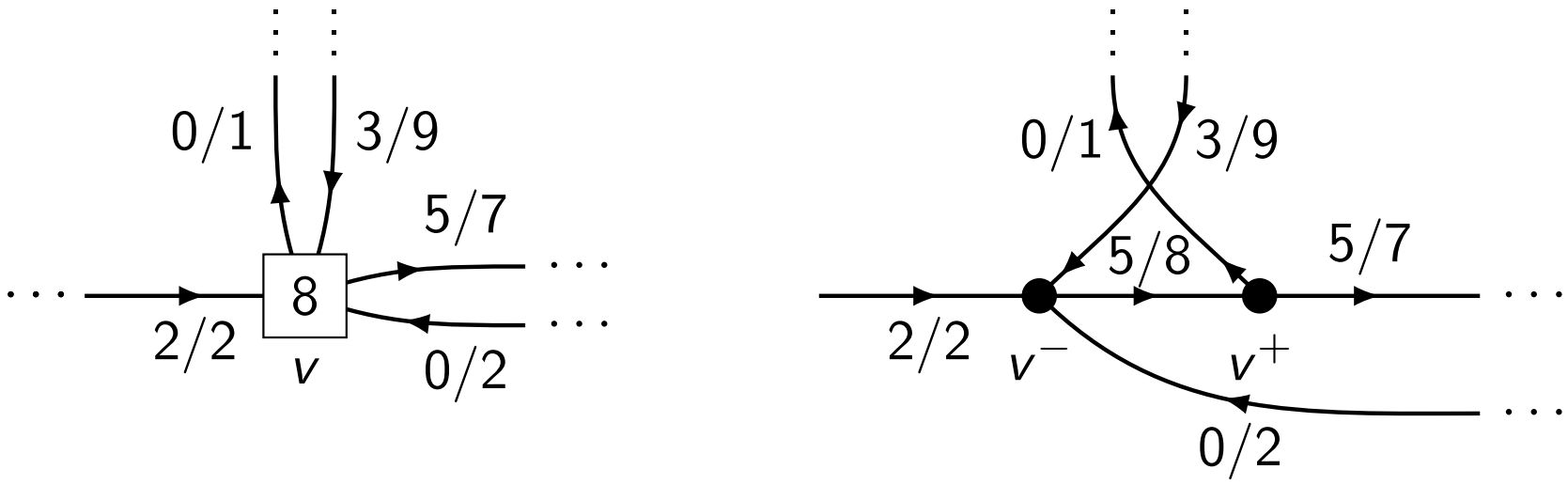


For each vertex v in the original vertex flow network:

- Replace v by new vertices v^+ and v^-, joined by an edge $v^- \to v^+$ with capacity $c_V(v)$;
- For each edge (u, v), add an edge (u^+, v^-) with capacity $c_E(u, v)$;
- For each edge (v, w), add an edge (v^+, w^-) with capacity $c_E(v, w)$.

Again, we can reduce this to a flow network!

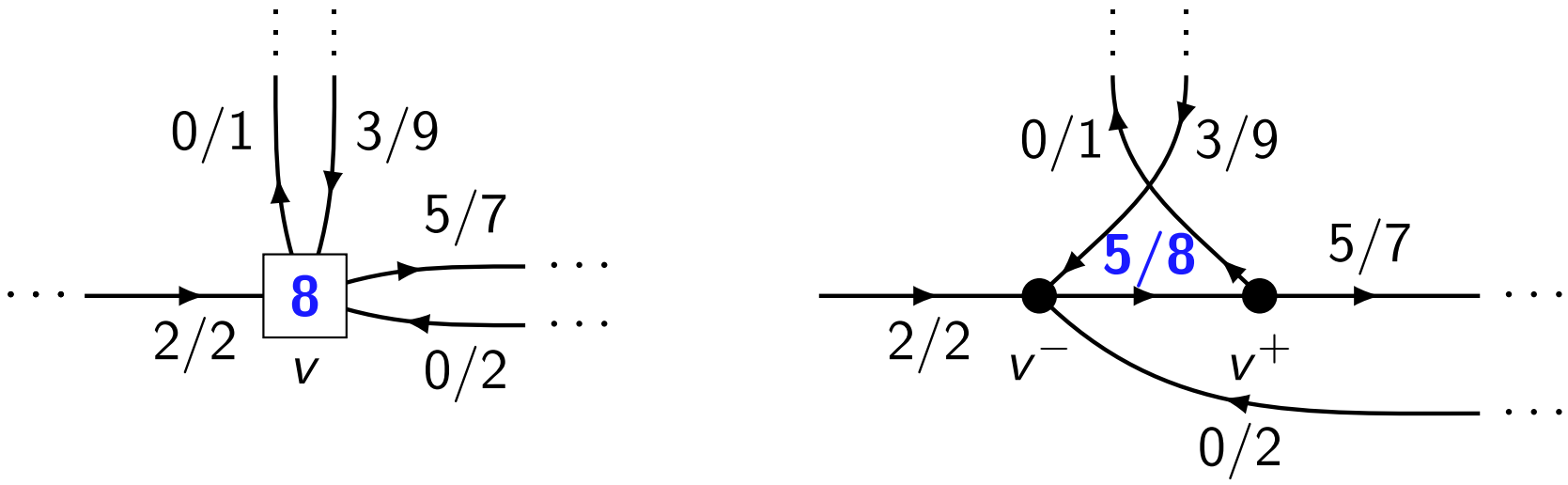
We use a design pattern called a **vertex gadget**: we simulate each vertex of the vertex flow network "independently" in the flow network.



Then flows on the left correspond to flows on the right as above:

Again, we can reduce this to a flow network!

We use a design pattern called a **vertex gadget**: we simulate each vertex of the vertex flow network "independently" in the flow network.

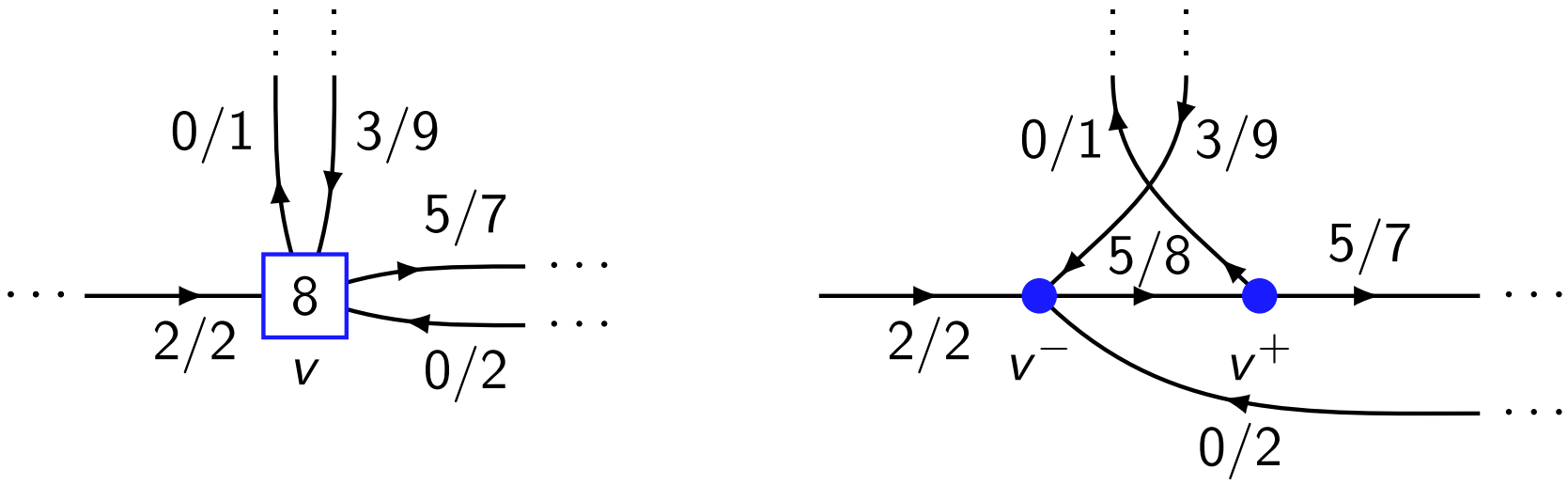


Then flows on the left correspond to flows on the right as above:

Vertex capacity constraint for $v \Leftrightarrow$ Edge capacity constraint for (v^-, v^+)

Again, we can reduce this to a flow network!

We use a design pattern called a **vertex gadget**: we simulate each vertex of the vertex flow network "independently" in the flow network.

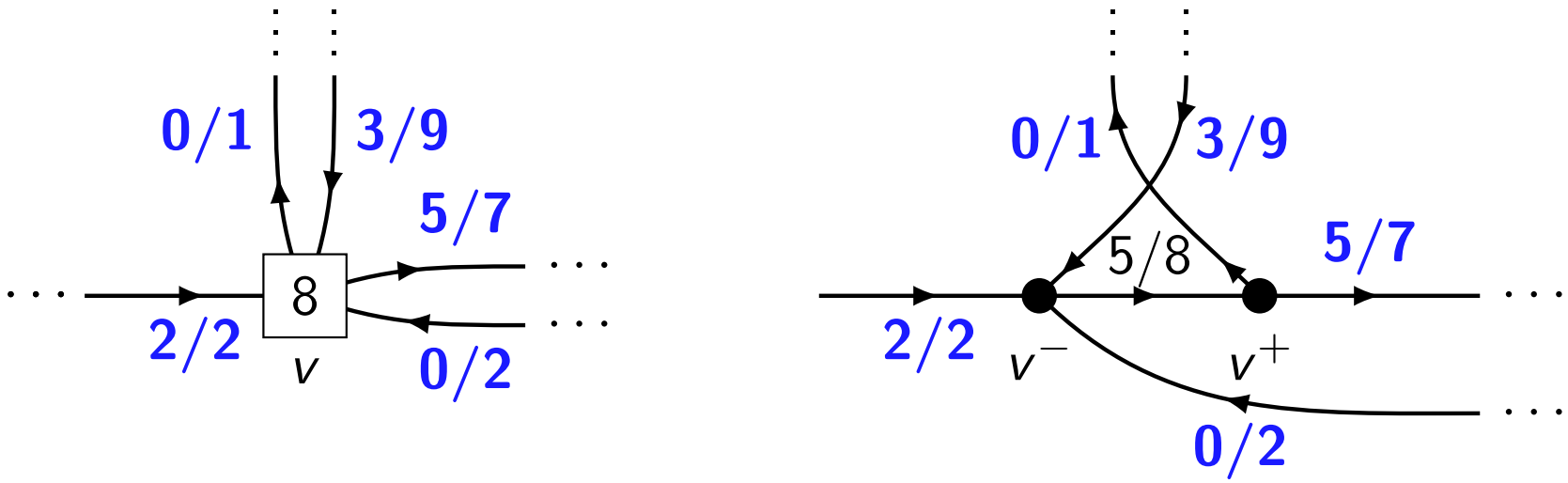


Then flows on the left correspond to flows on the right as above:

$$\text{Flow conservation at } v \Leftrightarrow \text{Flow conservation at } v^-, v^+$$

Again, we can reduce this to a flow network!

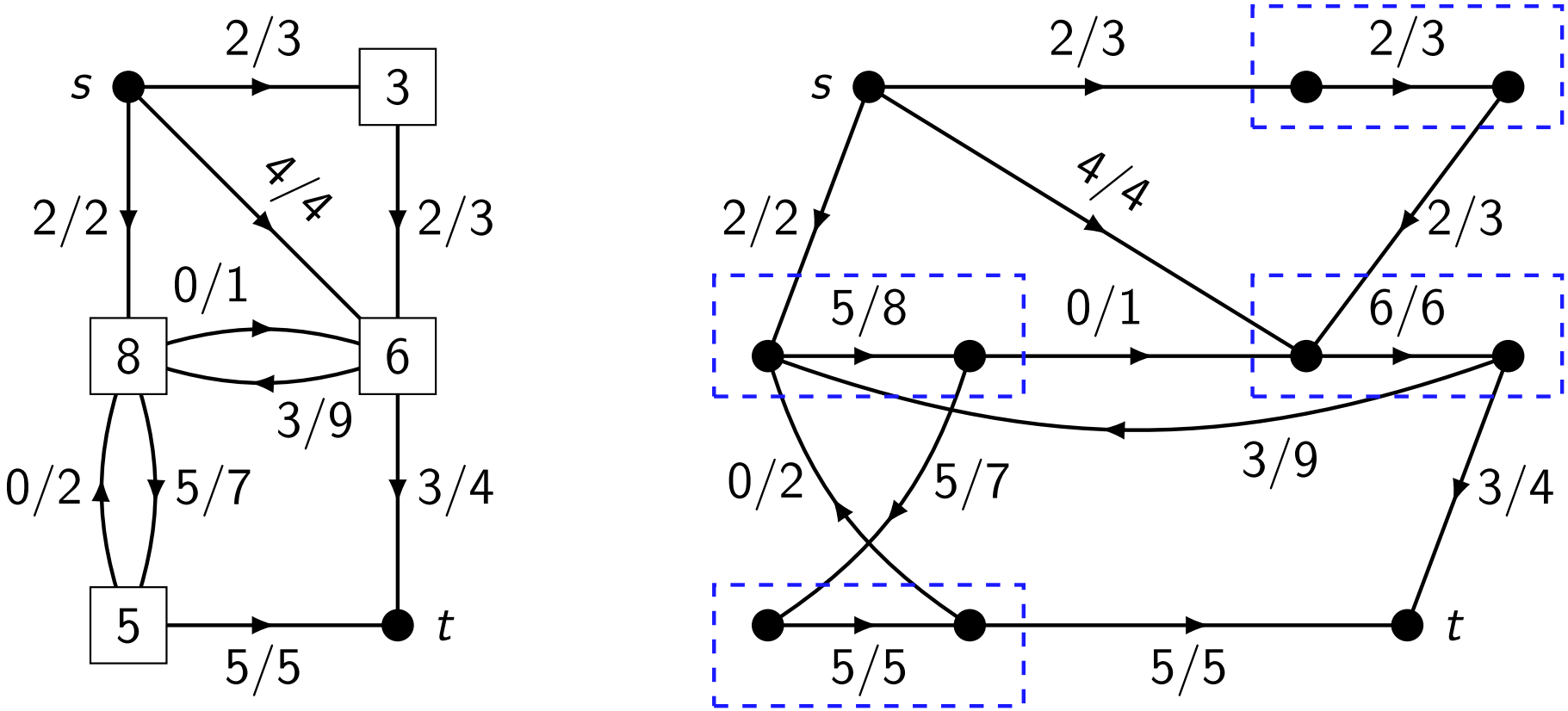
We use a design pattern called a **vertex gadget**: we simulate each vertex of the vertex flow network "independently" in the flow network.



Then flows on the left correspond to flows on the right as above:

Edge capacity constraints $\Leftrightarrow$ Edge capacity constraints other than (v^-, v^+).

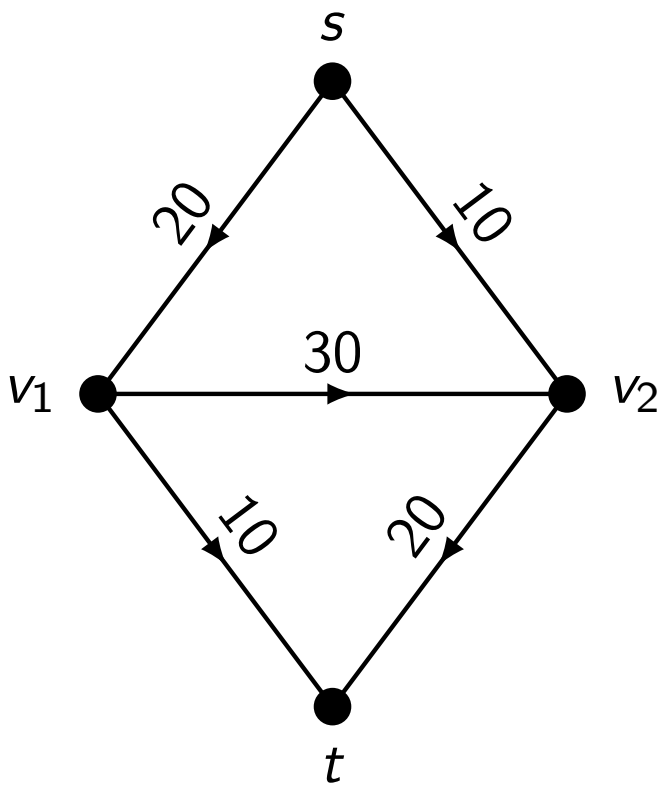
The full construction looks like this (with boxes to point out the gadgets):



As with circulations, to find a maximum flow we build the flow network on the right, run Edmonds-Karp, and then take the corresponding flow in the vertex flow network on the left.

Connection to linear programming

We can also look at flow problems as linear programming problems whose variables are the flow at each edge:

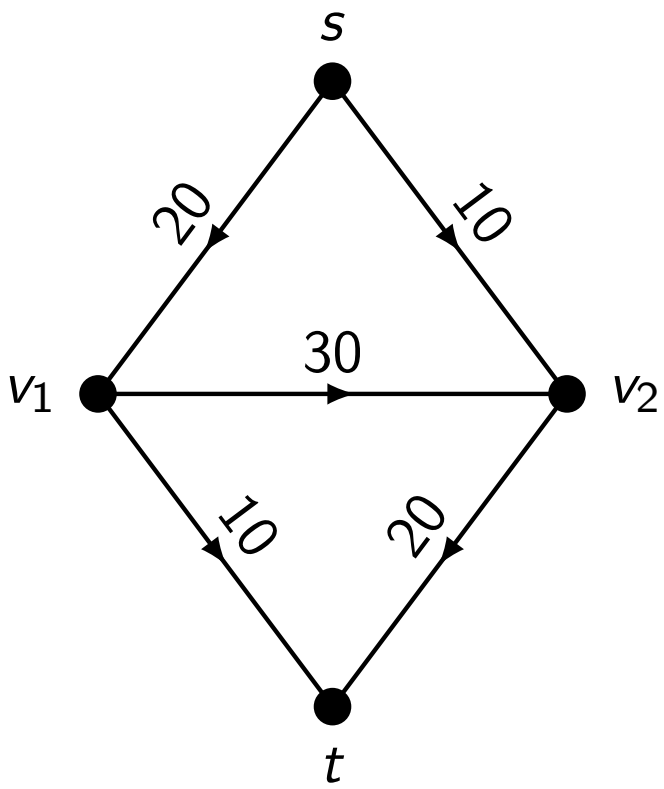


$$\boldsymbol{f(s, v_1) + f(s, v_2)} \to \max \text{ subject to}$$
$$f(s, v_1) = f(v_1, v_2) + f(v_1, t);$$
$$f(s, v_2) + f(v_1, v_2) = f(v_2, t);$$
$$0 \leq f(s, v_1) \leq 20;$$
$$0 \leq f(s, v_2) \leq 10;$$
$$0 \leq f(v_1, v_2) \leq 30;$$
$$0 \leq f(v_1, t) \leq 10;$$
$$0 \leq f(v_2, t) \leq 20.$$

The objective function is the value of the flow.

Connection to linear programming

We can also look at flow problems as linear programming problems whose variables are the flow at each edge:

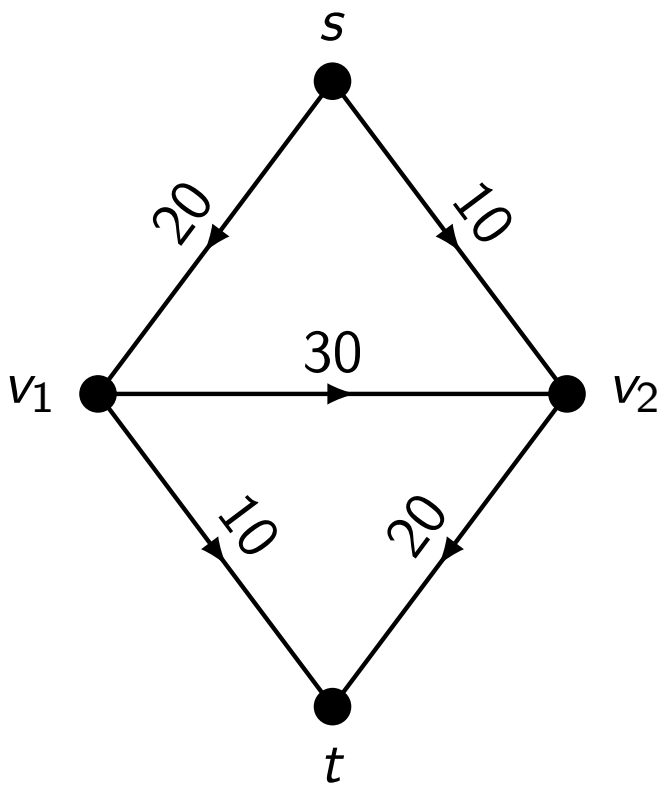


$$f(s, v_1) + f(s, v_2) \to \max \text{ subject to}$$
$$f(s, v_1) = f(v_1, v_2) + f(v_1, t);$$
$$f(s, v_2) + f(v_1, v_2) = f(v_2, t);$$
$$\mathbf{0 \le f(s, v_1) \le 20};$$
$$\mathbf{0 \le f(s, v_2) \le 10};$$
$$\mathbf{0 \le f(v_1, v_2) \le 30};$$
$$\mathbf{0 \le f(v_1, t) \le 10};$$
$$\mathbf{0 \le f(v_2, t) \le 20}.$$

These constraints are satisfied precisely when $0 \le f(e) \le c(e)$ for all edges e.

Connection to linear programming

We can also look at flow problems as linear programming problems whose variables are the flow at each edge:



$$f(s, v_1) + f(s, v_2) \to \max \text{ subject to}$$

$$\boldsymbol{f(s, v_1) = f(v_1, v_2) + f(v_1, t)};$$

$$\boldsymbol{f(s, v_2) + f(v_1, v_2) = f(v_2, t)};$$

$$0 \le f(s, v_1) \le 20;$$

$$0 \le f(s, v_2) \le 10;$$

$$0 \le f(v_1, v_2) \le 30;$$

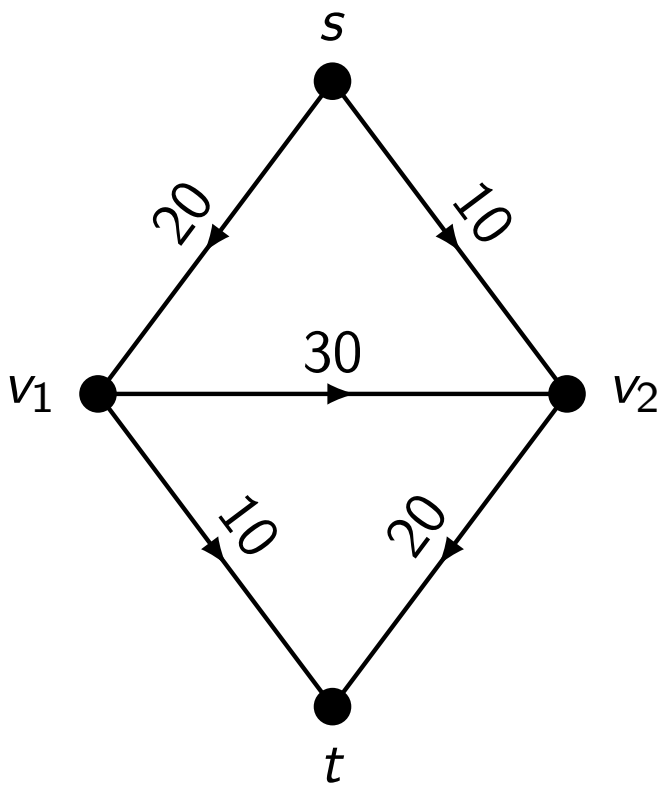
$$0 \le f(v_1, t) \le 10;$$

$$0 \le f(v_2, t) \le 20.$$

These constraints are satisfied precisely when flow is conserved at v_1 and v_2.

Connection to linear programming

We can also look at flow problems as linear programming problems whose variables are the flow at each edge:

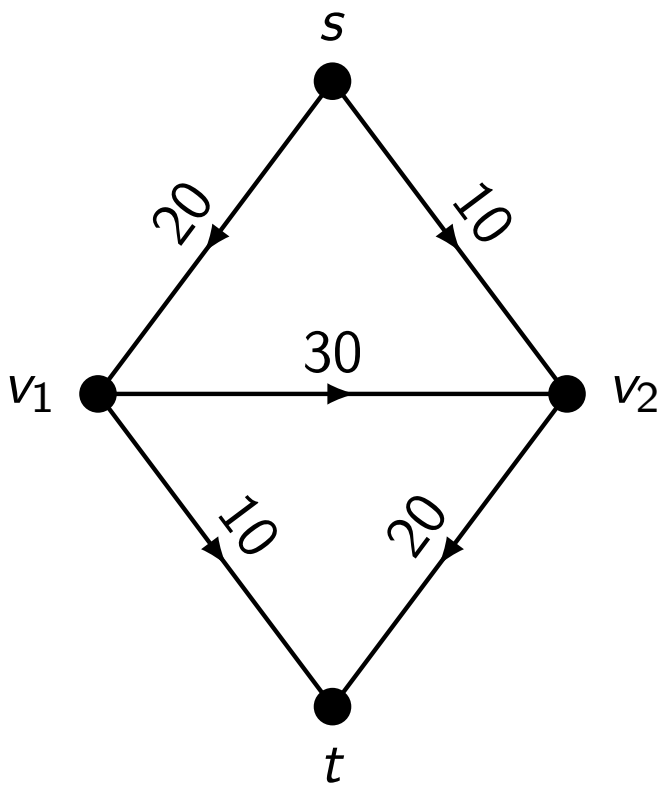


$$
\begin{aligned}
f(s, v_1) + f(s, v_2) \to \max \text{ subject to} \\
f(s, v_1) = f(v_1, v_2) + f(v_1, t); \\
f(s, v_2) + f(v_1, v_2) = f(v_2, t); \\
0 \le f(s, v_1) \le 20; \\
0 \le f(s, v_2) \le 10; \\
0 \le f(v_1, v_2) \le 30; \\
0 \le f(v_1, t) \le 10; \\
0 \le f(v_2, t) \le 20.
\end{aligned}
$$

So maximum flows on the left correspond exactly to optimal LP solutions on the right, and we could use LP algorithms to solve flow problems!

Connection to linear programming

We can also look at flow problems as linear programming problems whose variables are the flow at each edge:



$$
\begin{aligned}
f(s, v_1) + f(s, v_2) &\to \max \text{ subject to} \\
f(s, v_1) &= f(v_1, v_2) + f(v_1, t); \\
f(s, v_2) + f(v_1, v_2) &= f(v_2, t); \\
0 \le f(s, v_1) &\le 20; \\
0 \le f(s, v_2) &\le 10; \\
0 \le f(v_1, v_2) &\le 30; \\
0 \le f(v_1, t) &\le 10; \\
0 \le f(v_2, t) &\le 20.
\end{aligned}
$$

Sadly, this is a nice connection but not a good idea in practice — usually flow-specific algorithms are much faster...