Szenzorhálózat topológiájának tervezése és optimalizálása gráfelméleti módszerekkel

Mezőgazdasági vezeték nélküli szenzorhálózatok topológiájának optimalizálása gráfelméleti módszerekkel

Groff Tamás – GCY3H5

Bevezetés – IoT technológiák



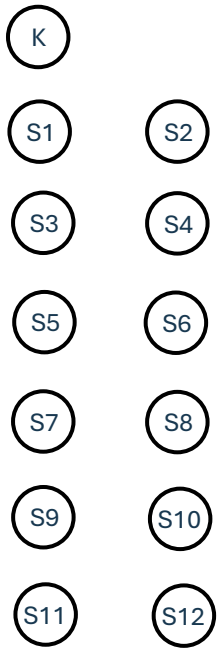
Bevezetés – Meglévő megoldások

- LoRaWAN
- NB–IoT
- dFarm



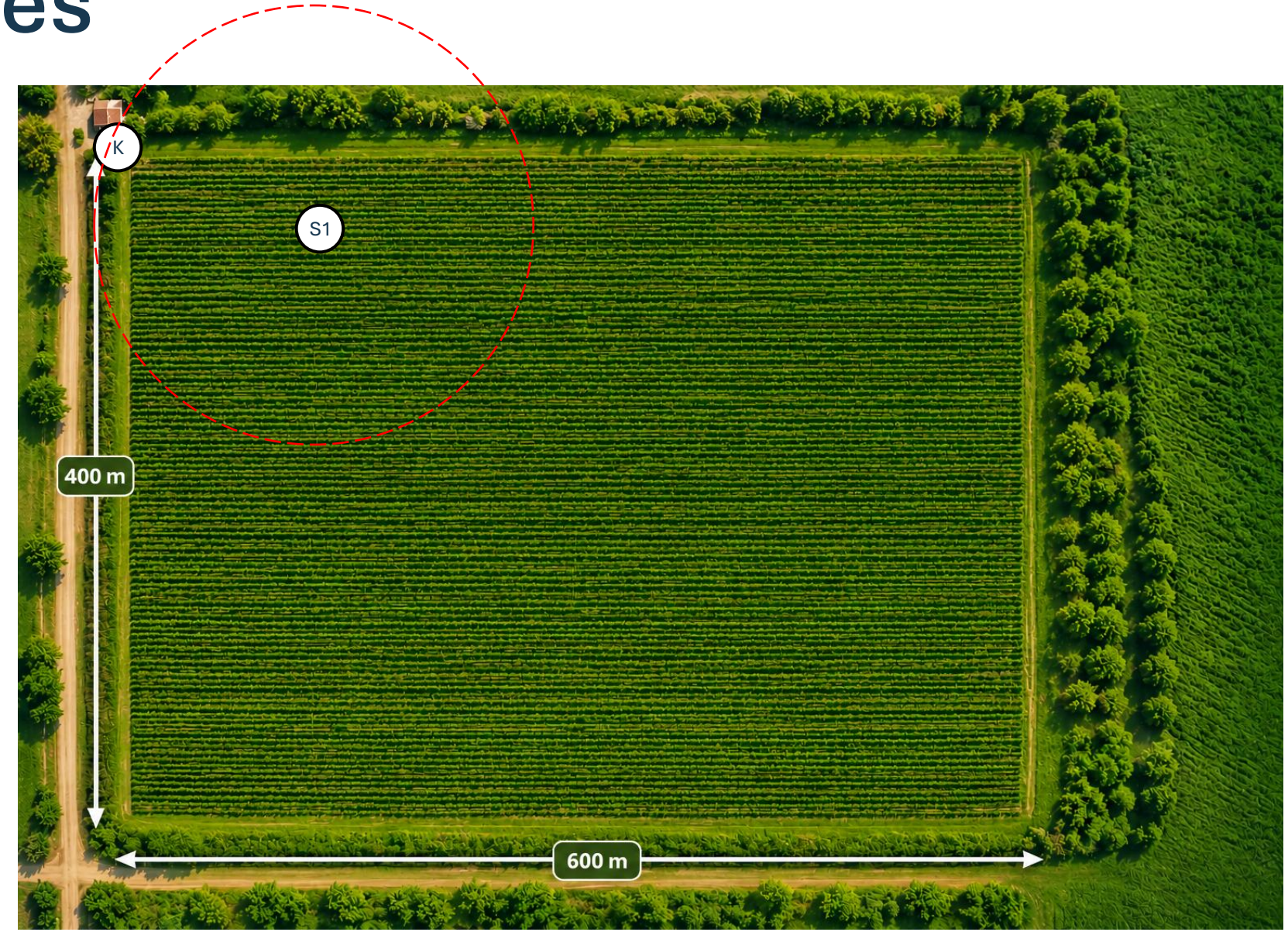


Tervezés

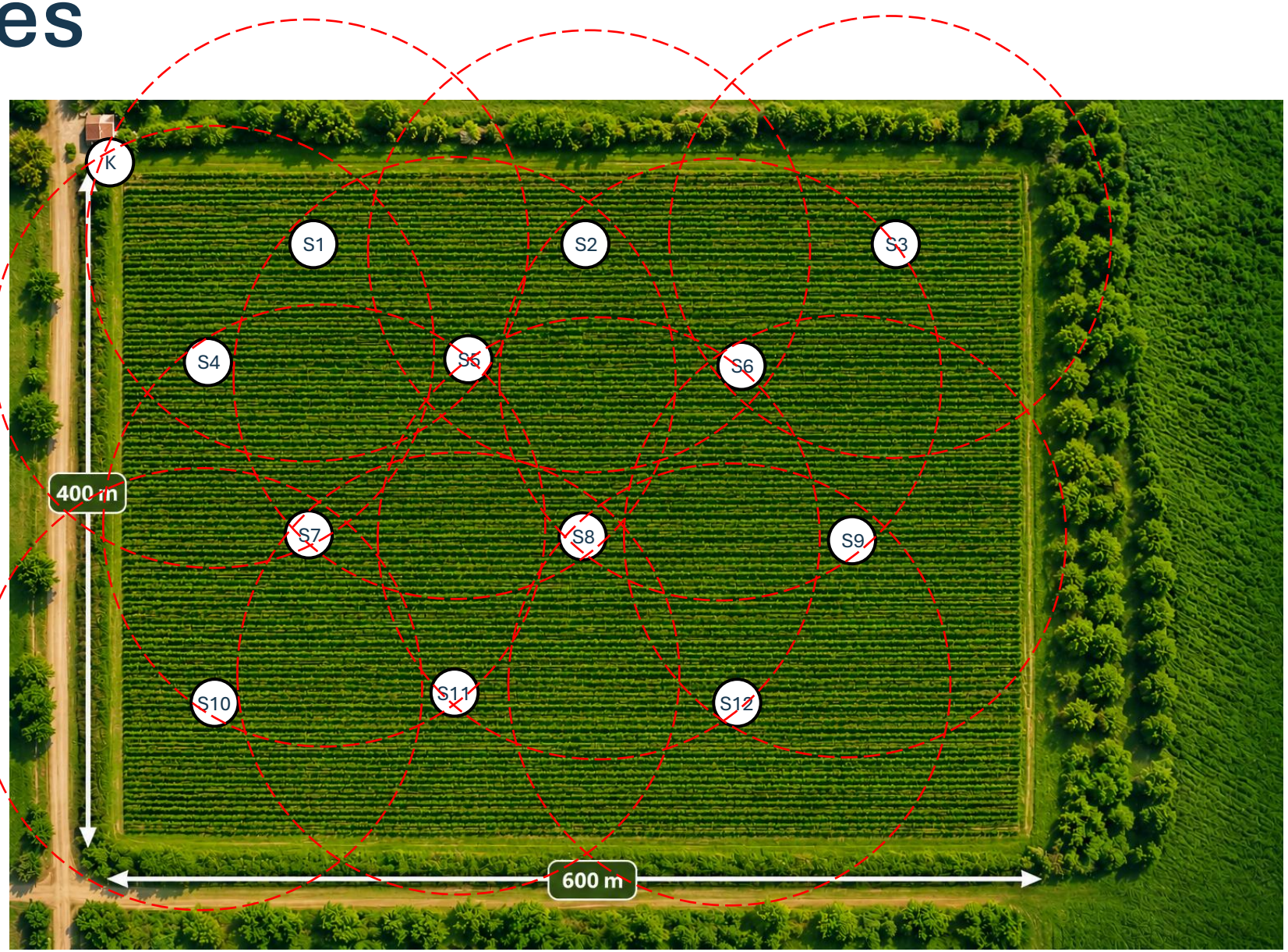




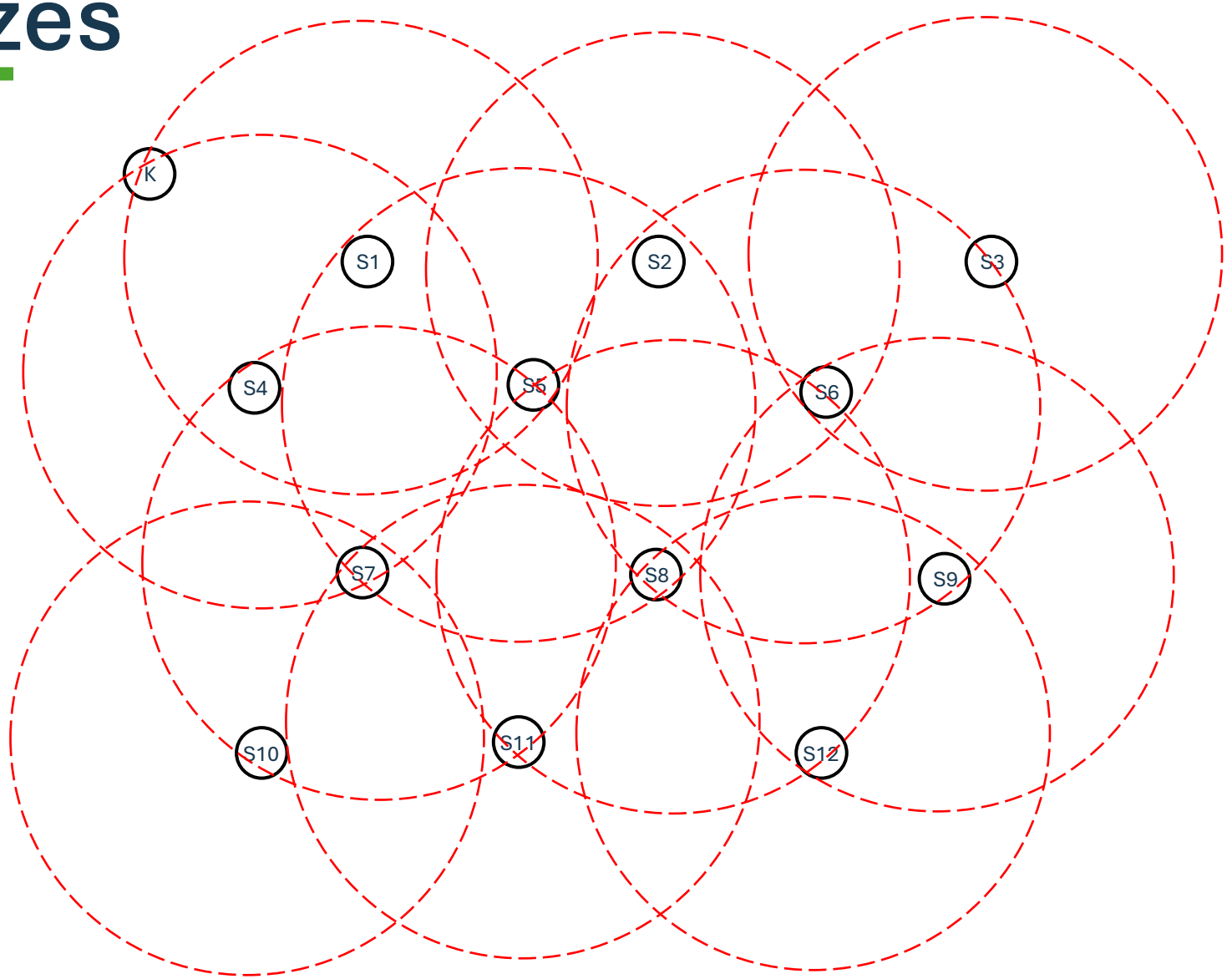
Tervezés



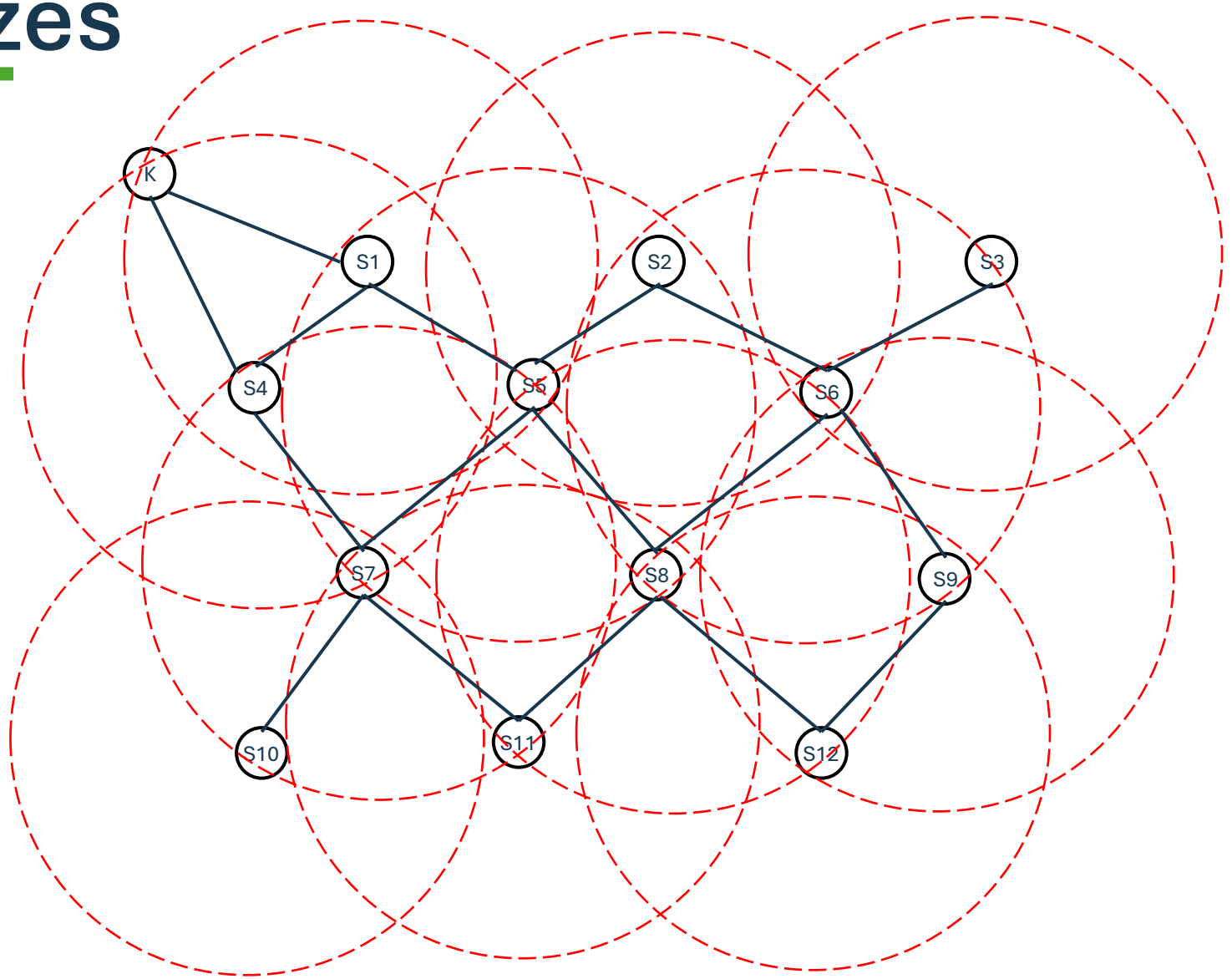
Tervezés



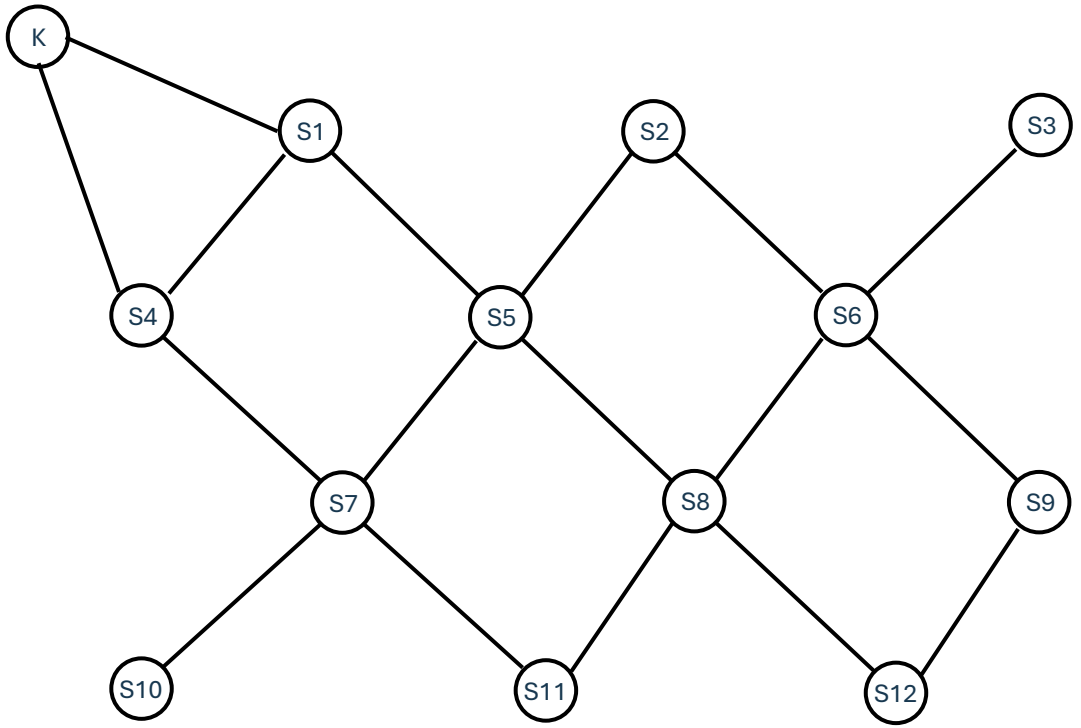
Tervezés



Tervezés



Tervezés

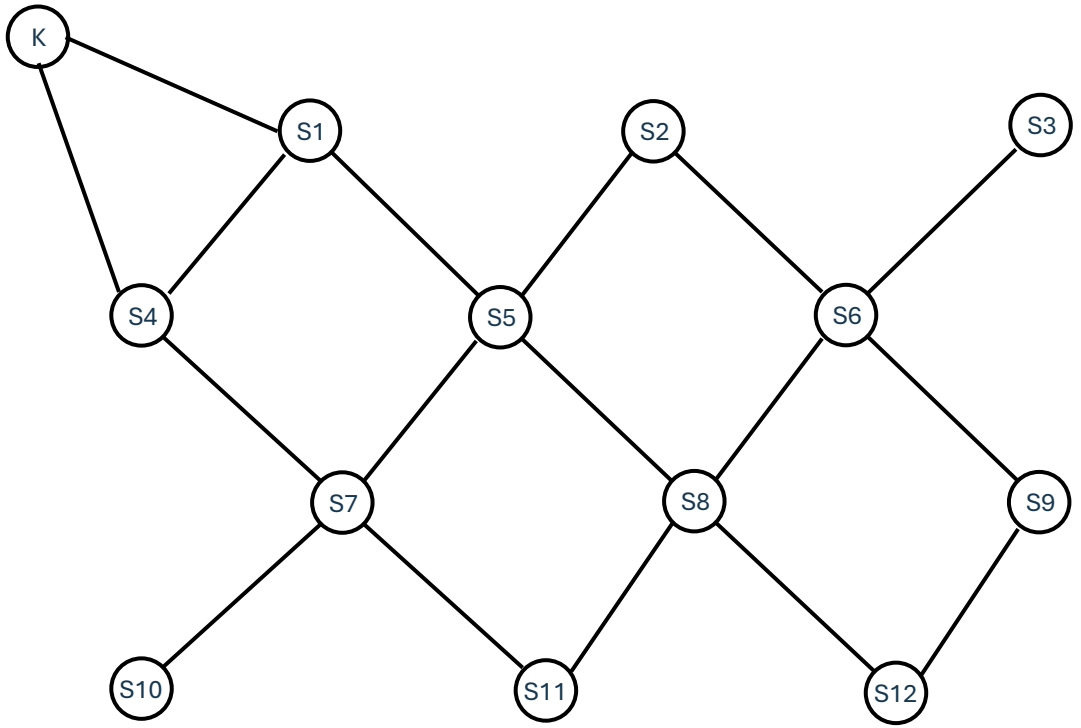


Tervezés

A hálózatot egy G=(V,E) gráffal írjuk le, ahol:

- V: a szenzorok és a központi adatgyűjtő halmaza
- E: a lehetséges vezeték nélküli kommunikációs kapcsolatok halmaza





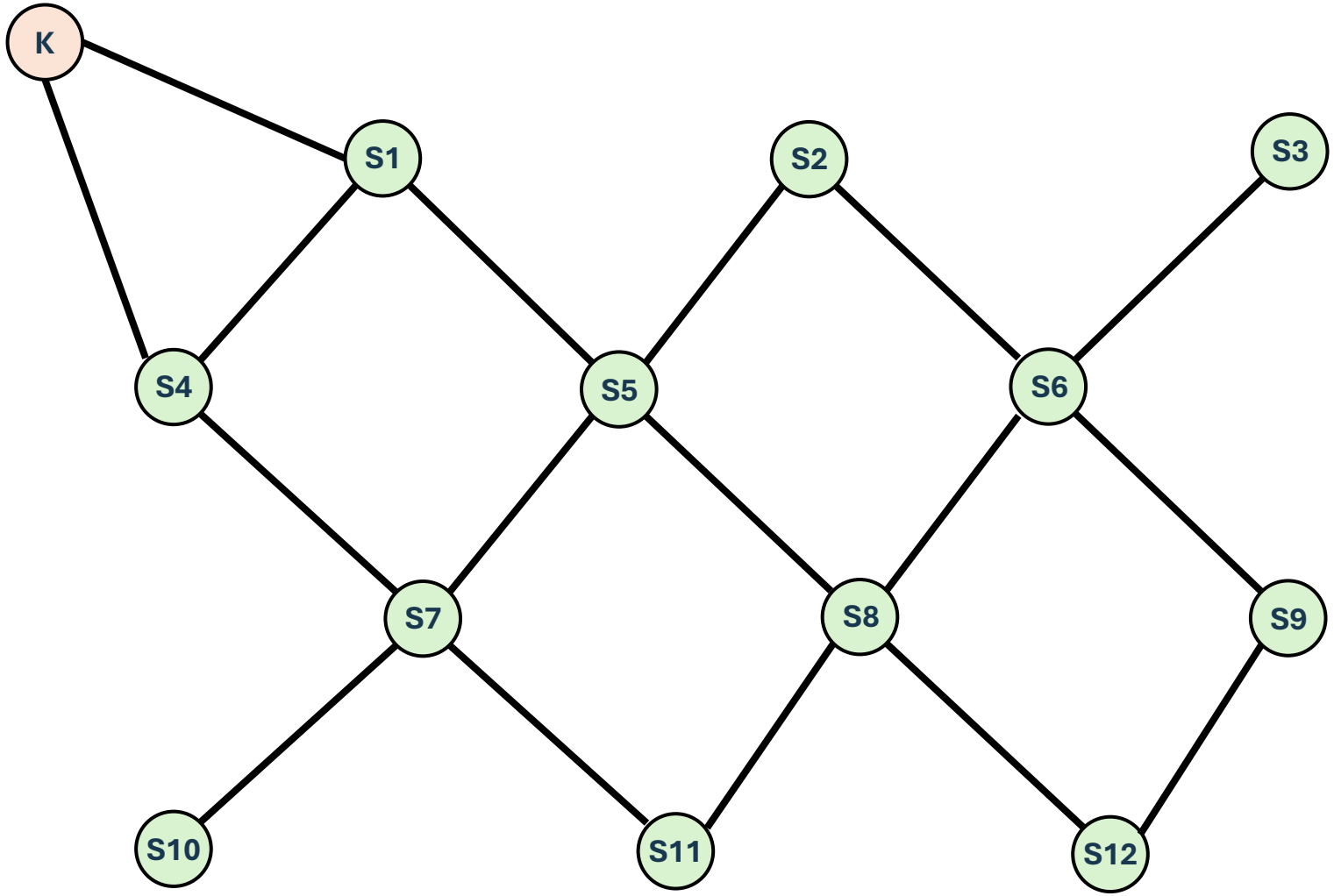
Következőkben

1. Összefüggőség, elérhetőség, útvonalválasztás
2. Hibatűrésre megoldások
3. Energiakezelés, optimalizálási lehetőségek

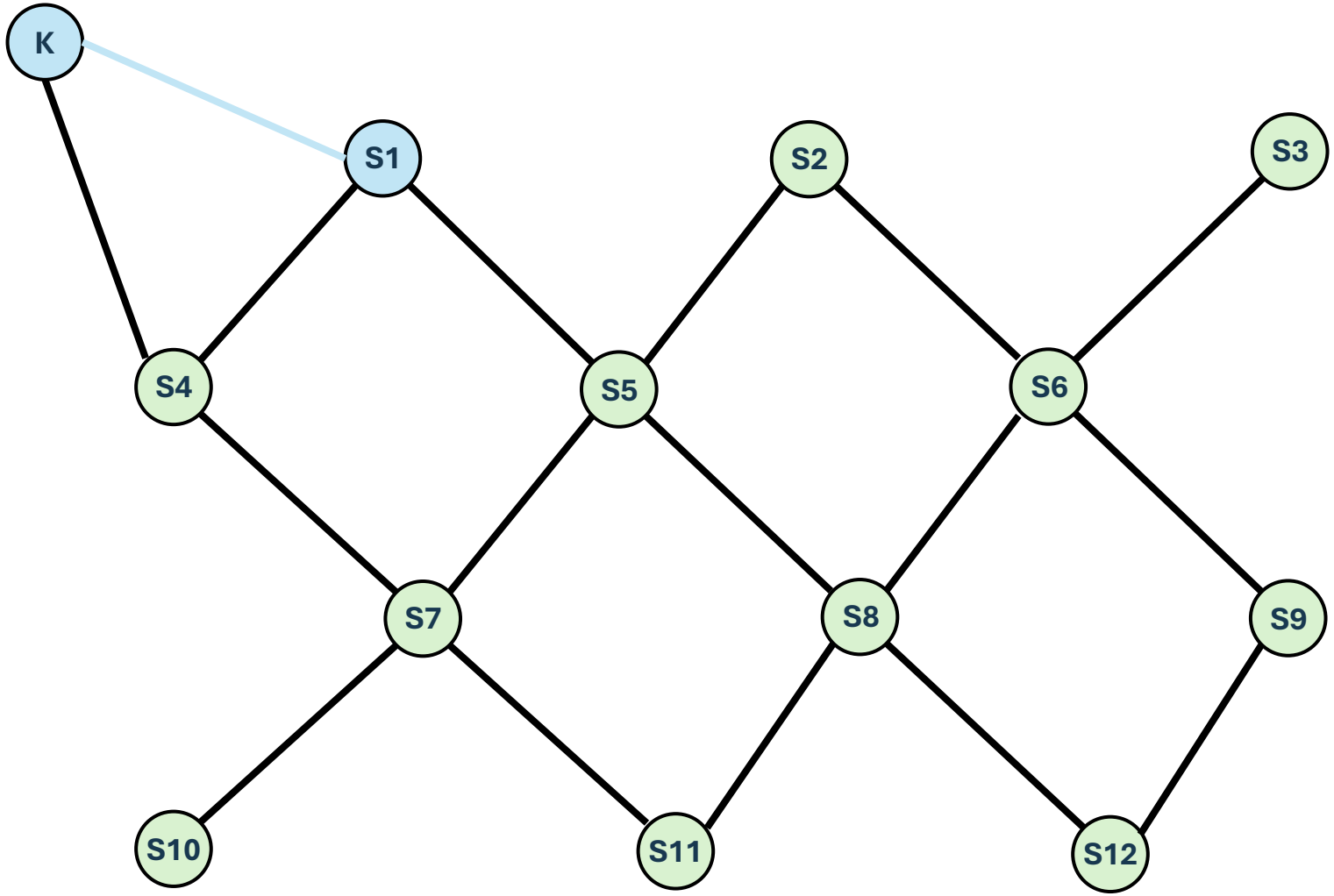
Szenzorhálózat felépítése BFS segítségével

1. Szenzorok kihelyezése és bekapcsolása
2. Szomszédok felderítése
3. Kapcsolati információk összegyűjtése
4. A központ felépíti a gráfot
5. A központ BFS-t futtat
6. Az útvonalak kijelölése és a szenzorok értesítése

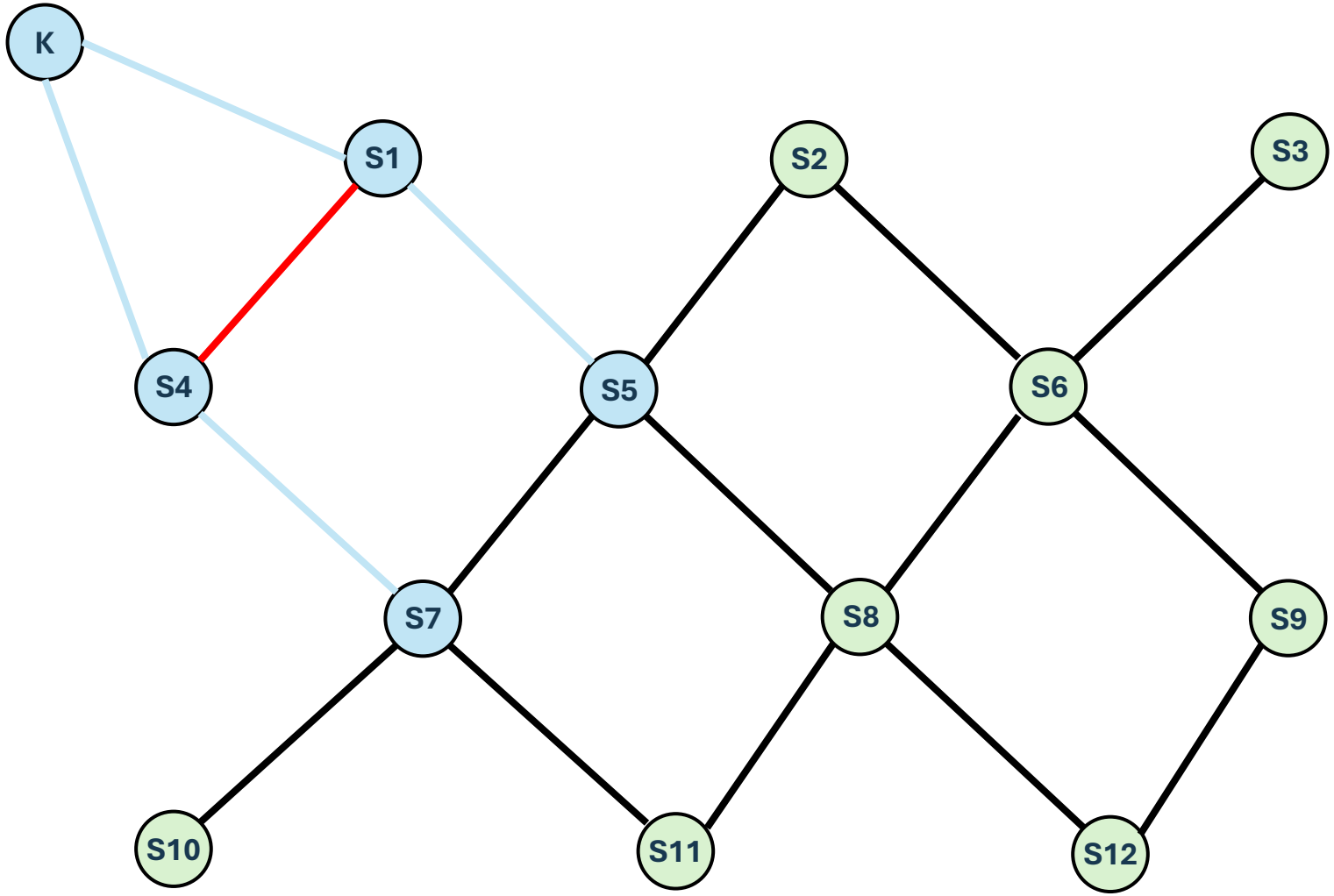
Központban BFS futtatása



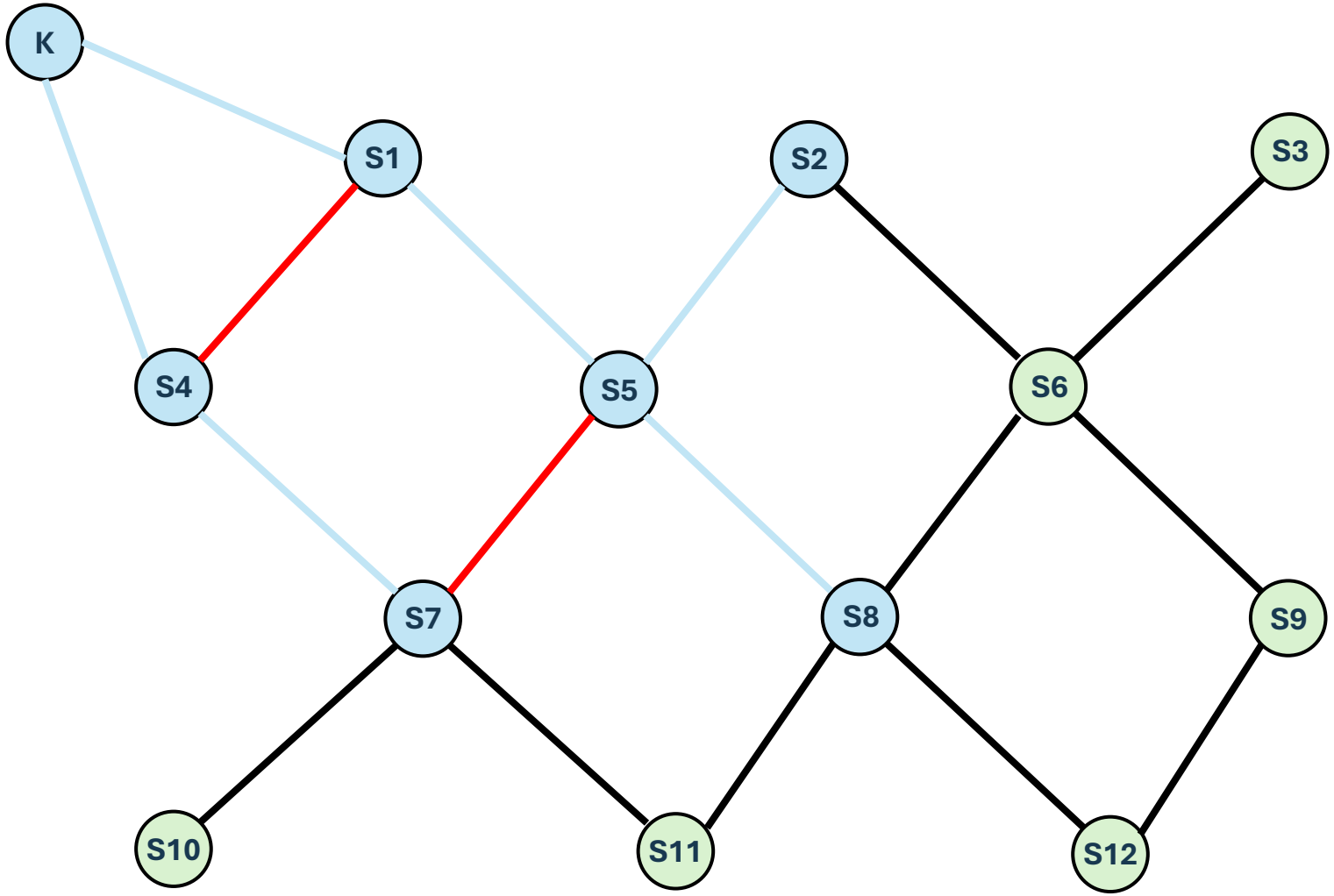
Központban BFS futtatása



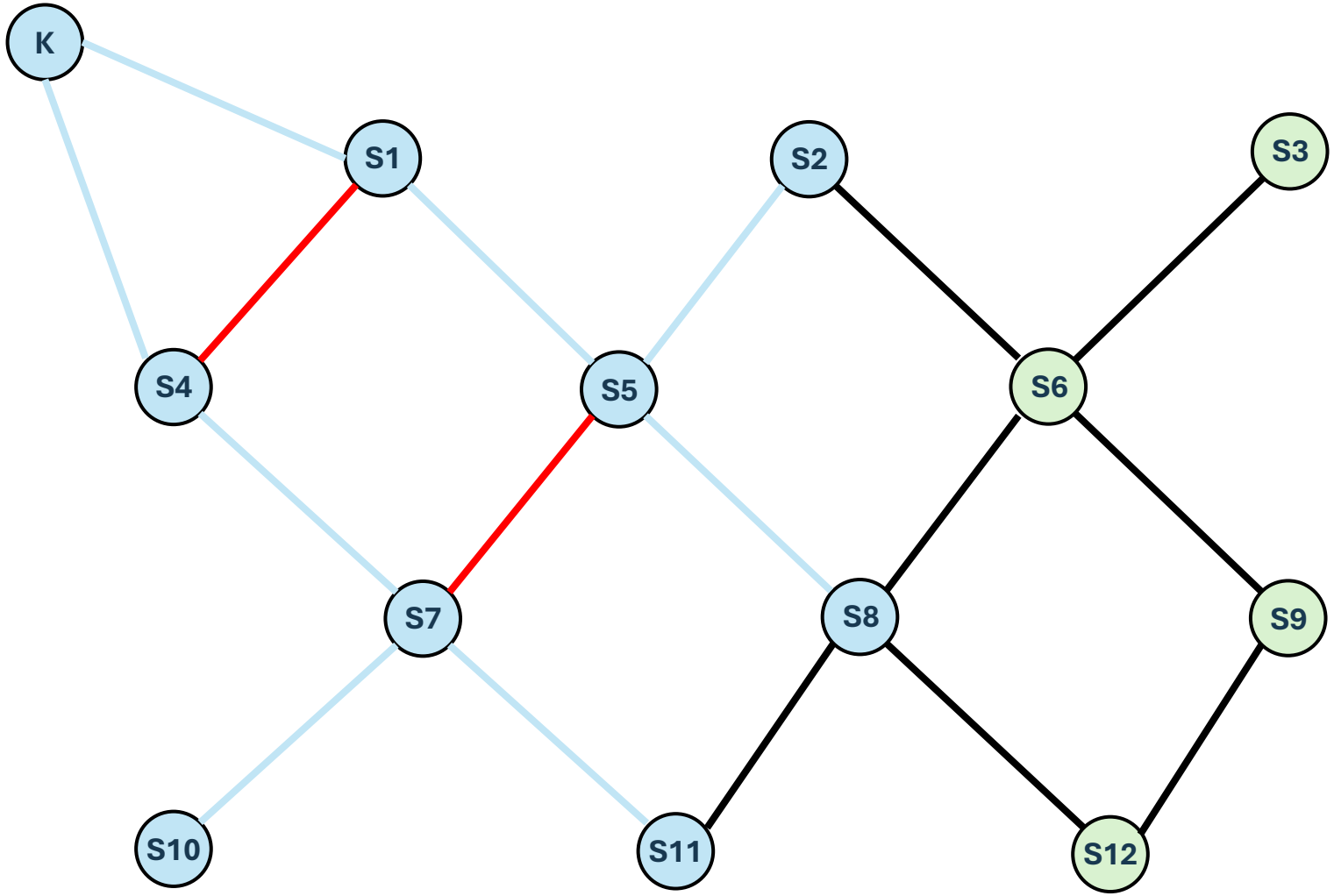
Központban BFS futtatása



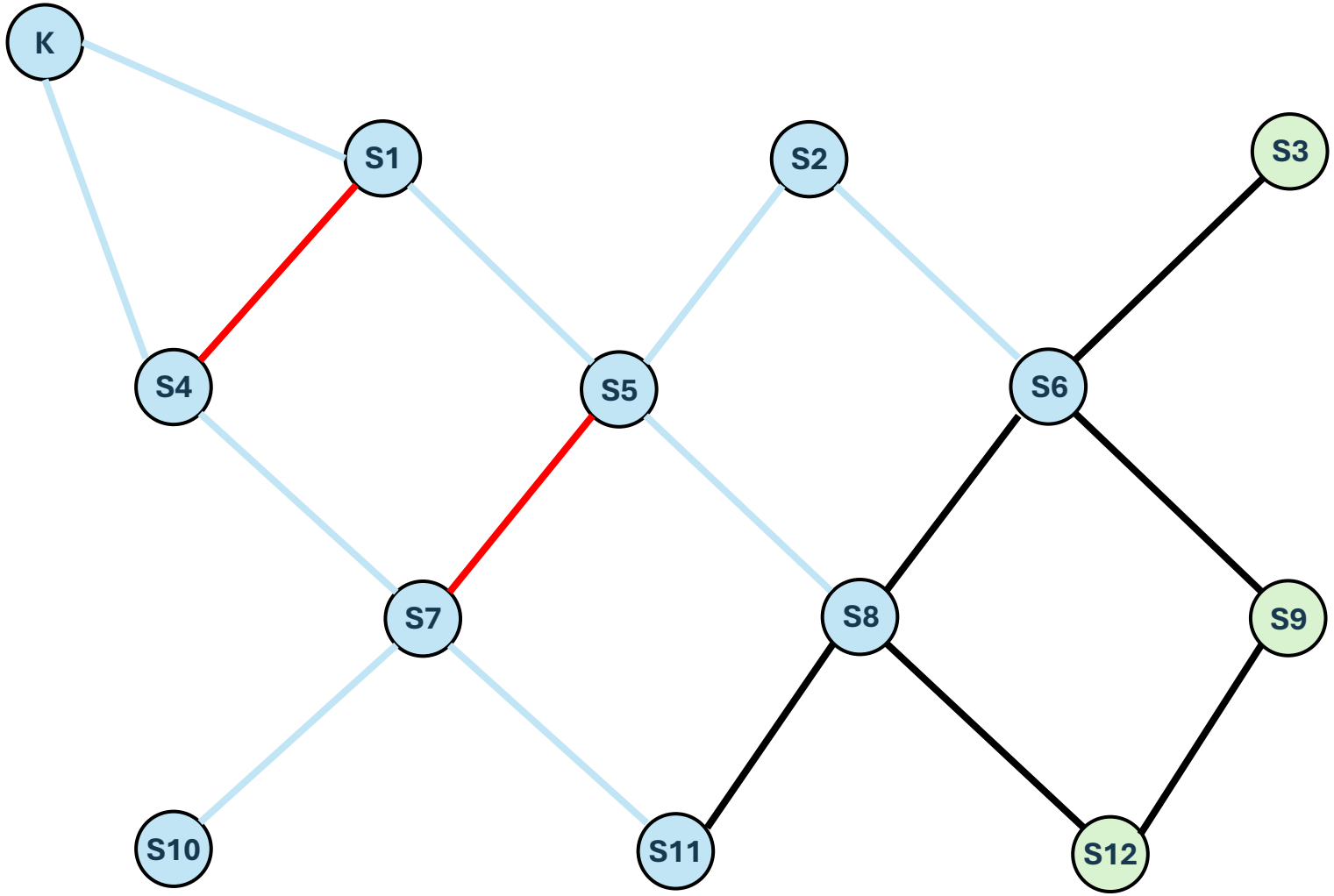
Központban BFS futtatása



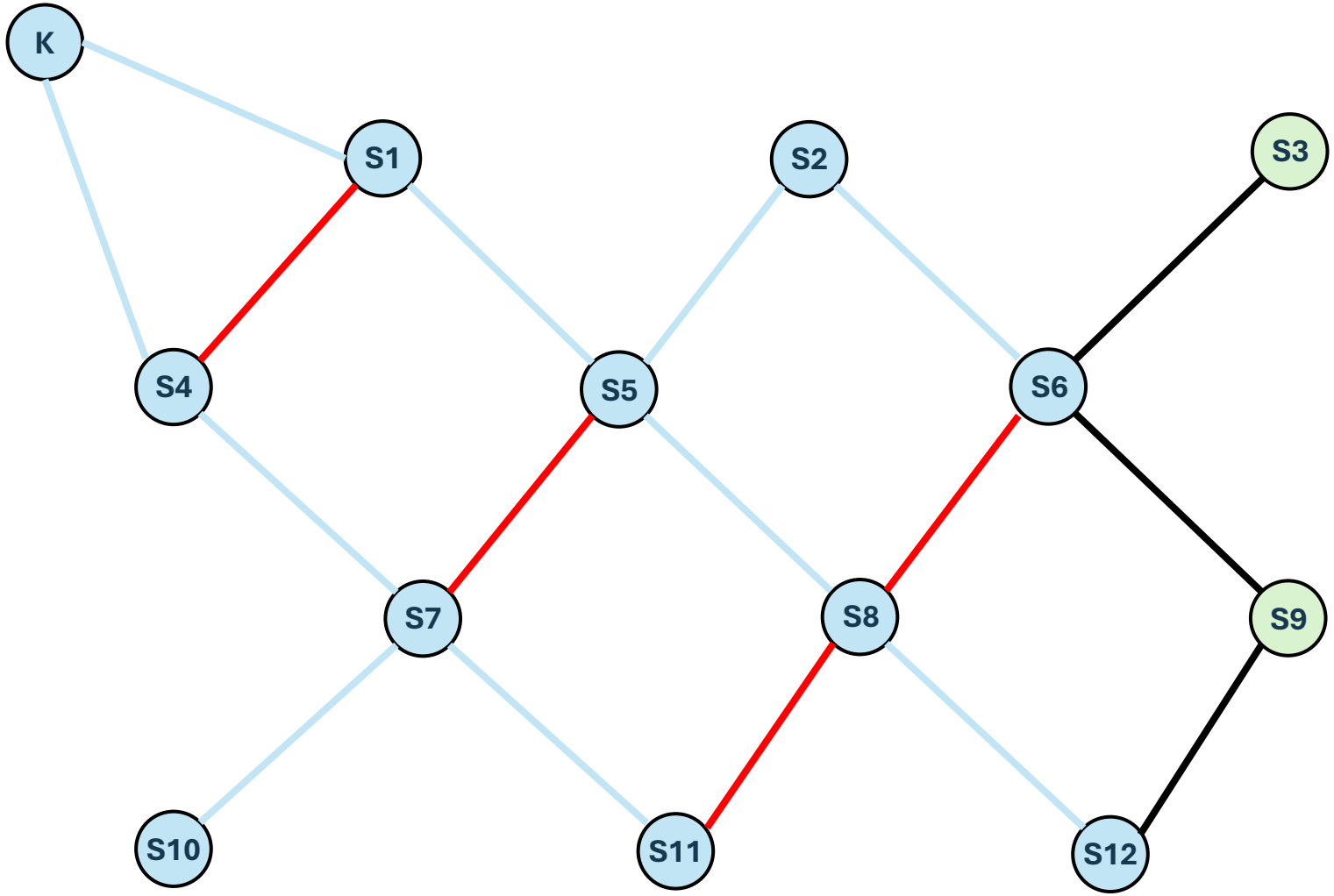
Központban BFS futtatása



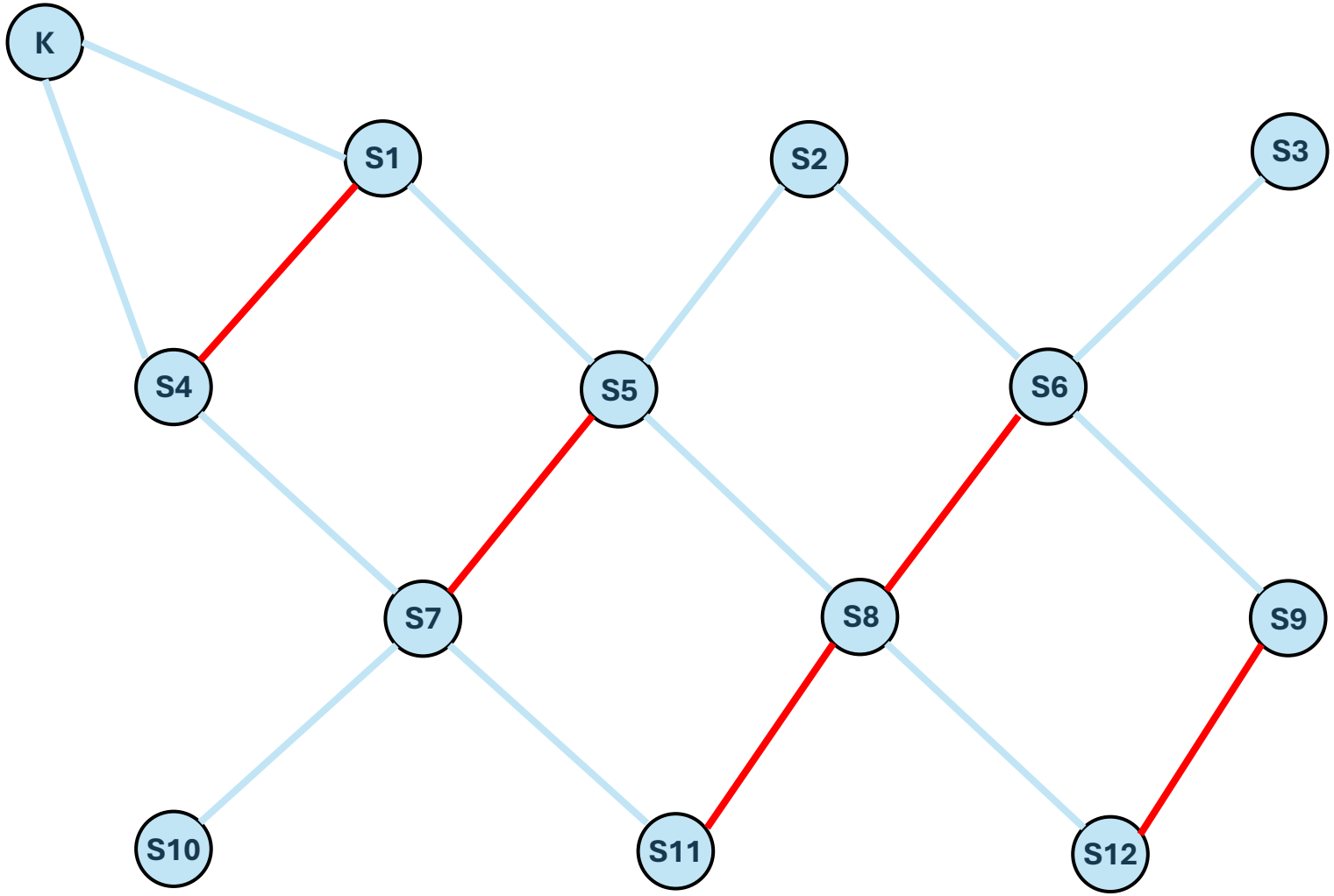
Központban BFS futtatása



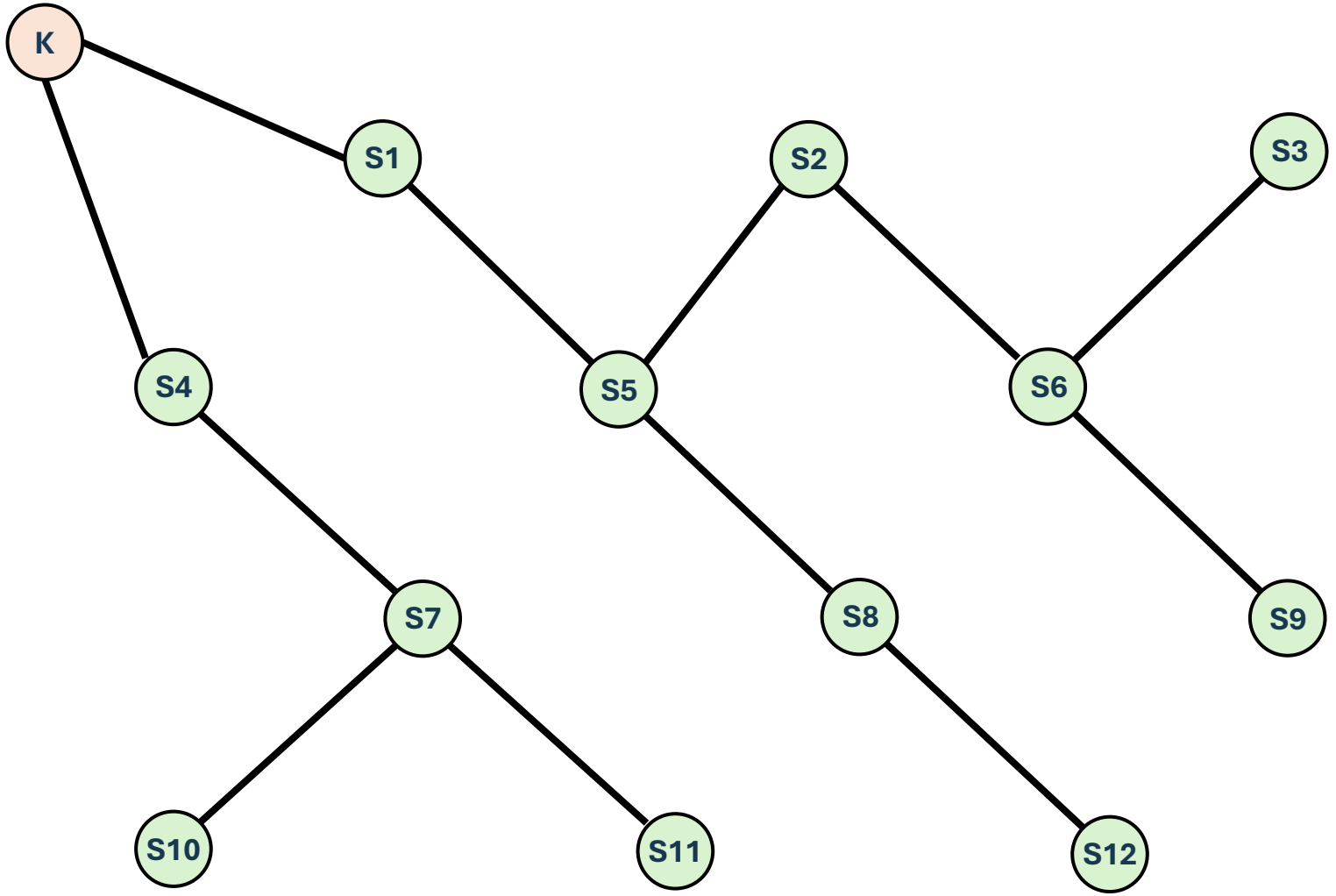
Központban BFS futtatása



Központban BFS futtatása



Központban BFS futtatása



Egy szenzor kiesése

Hiba észlelése

Hogyan derül ki, hogy egy szenzor nem válaszol?

- Állapot üzenetek
- Elmaradó állapotfrissítés

Hálózat újratervezése

Mely kapcsolatok és útvonalak maradtak használhatók?

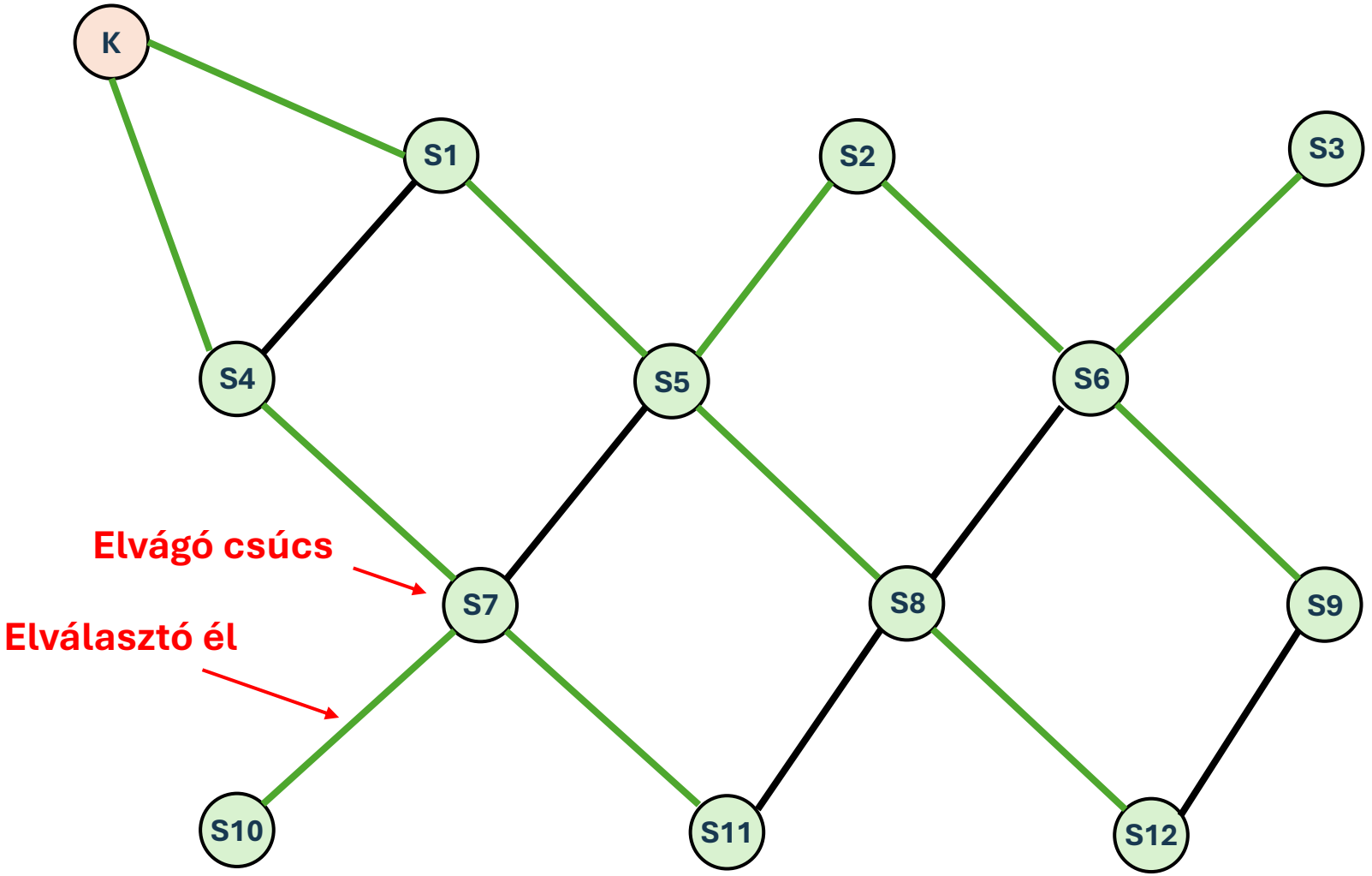
1. A központ értesül a kiesésről, és frissíti a gráfot
2. Újra lefuttatjuk a BFS-t a K csúcsból kiindulva
3. Megkapjuk az új kommunikációs fát

Kommunikáció helyreállítása

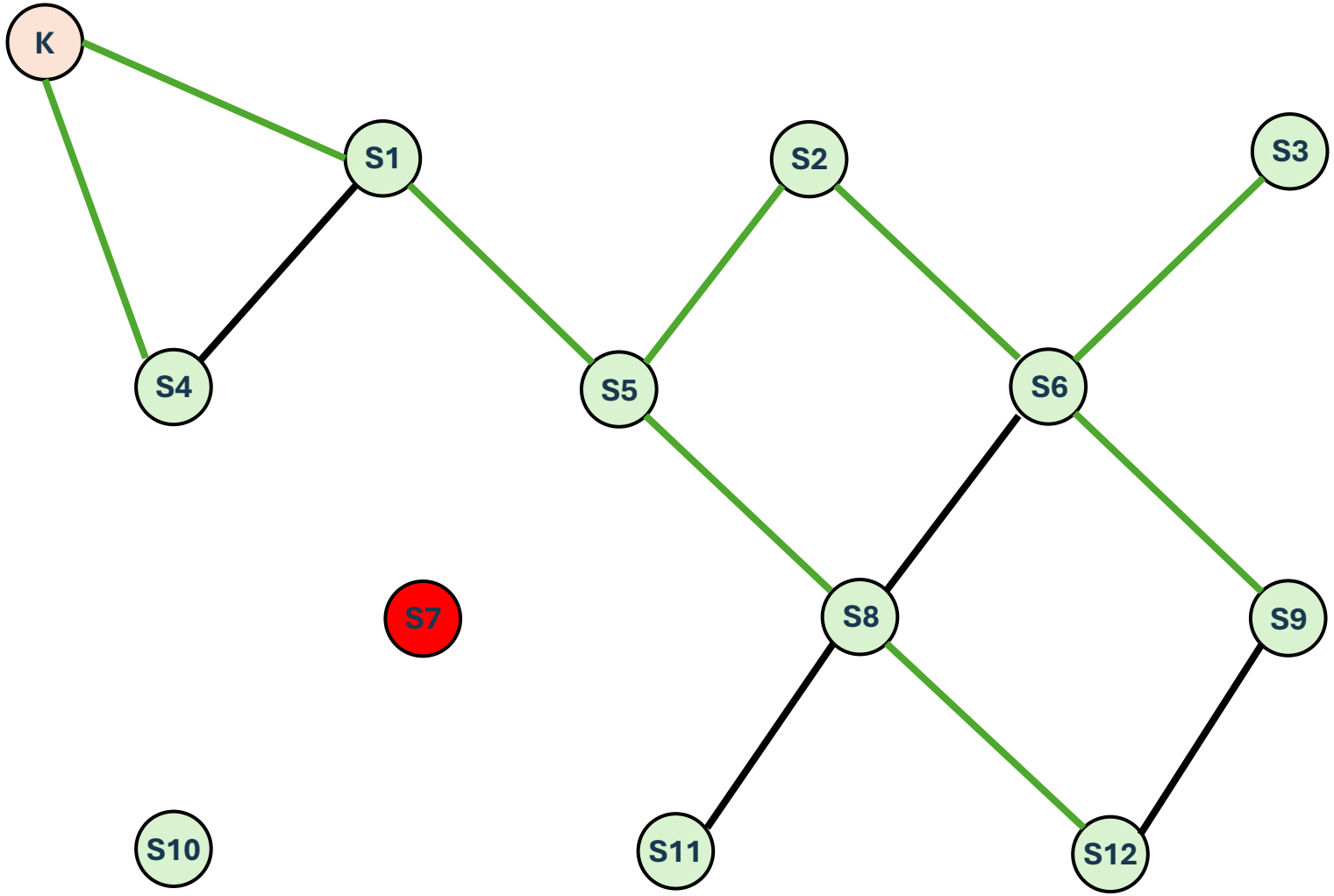
Hogyan állítjuk át a szenzorokat az új útvonalakra?

- Útvonalváltó üzenetek
- Egymás utáni végrehajtás, a hálózatnak továbbra is összefüggőnek kell maradnia

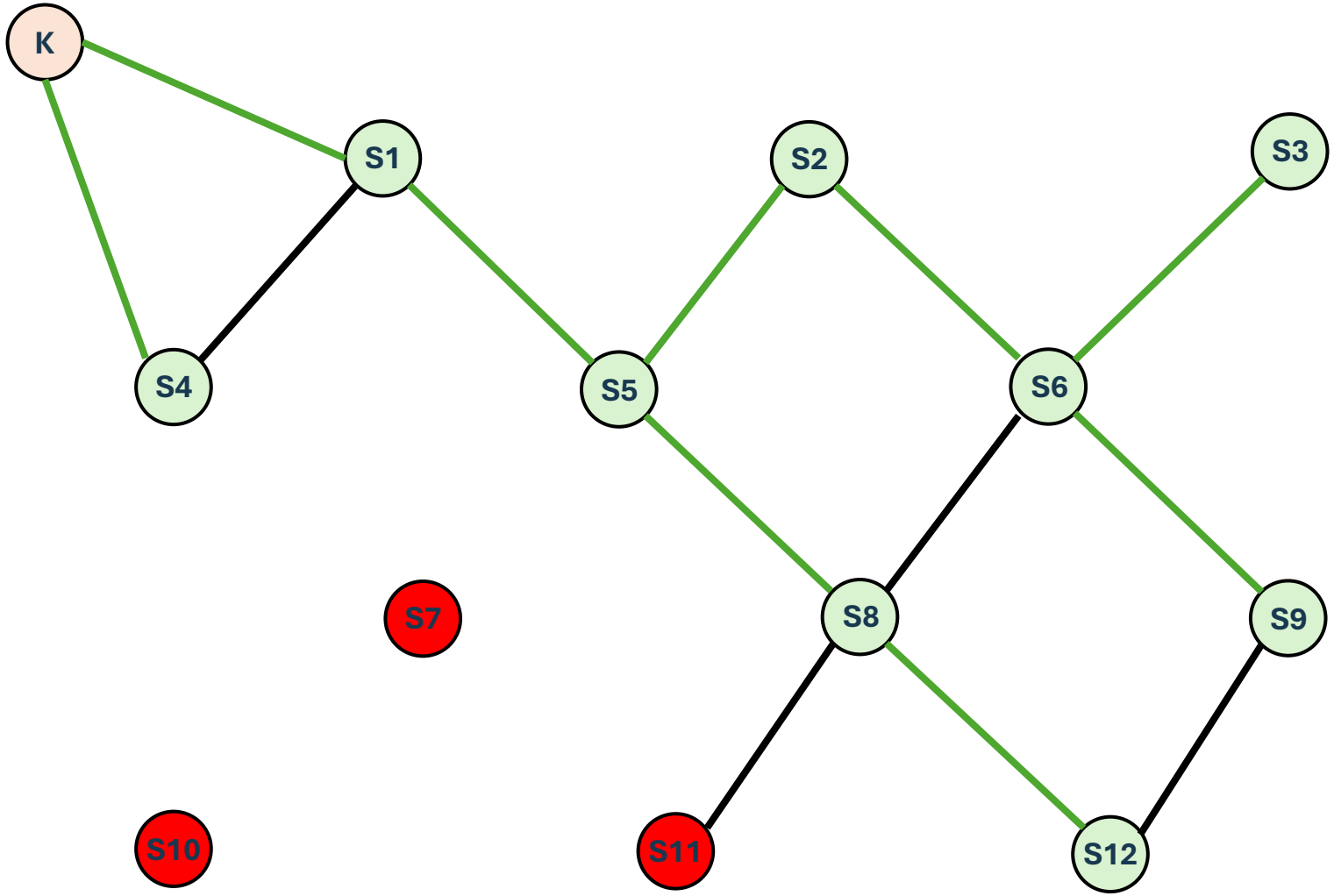
Szenzor kiesés példa 1.



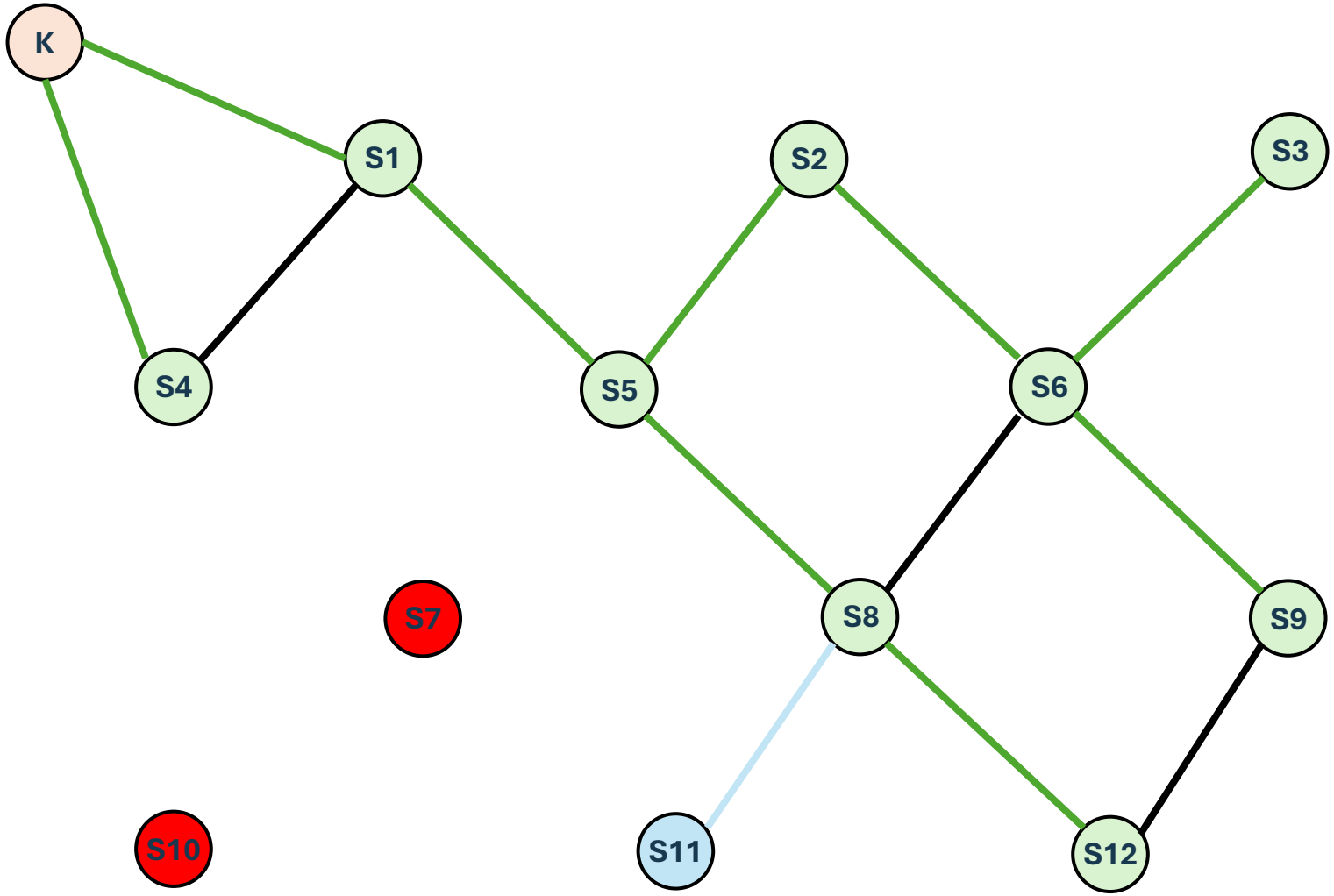
Szenzor kiesés példa 1.



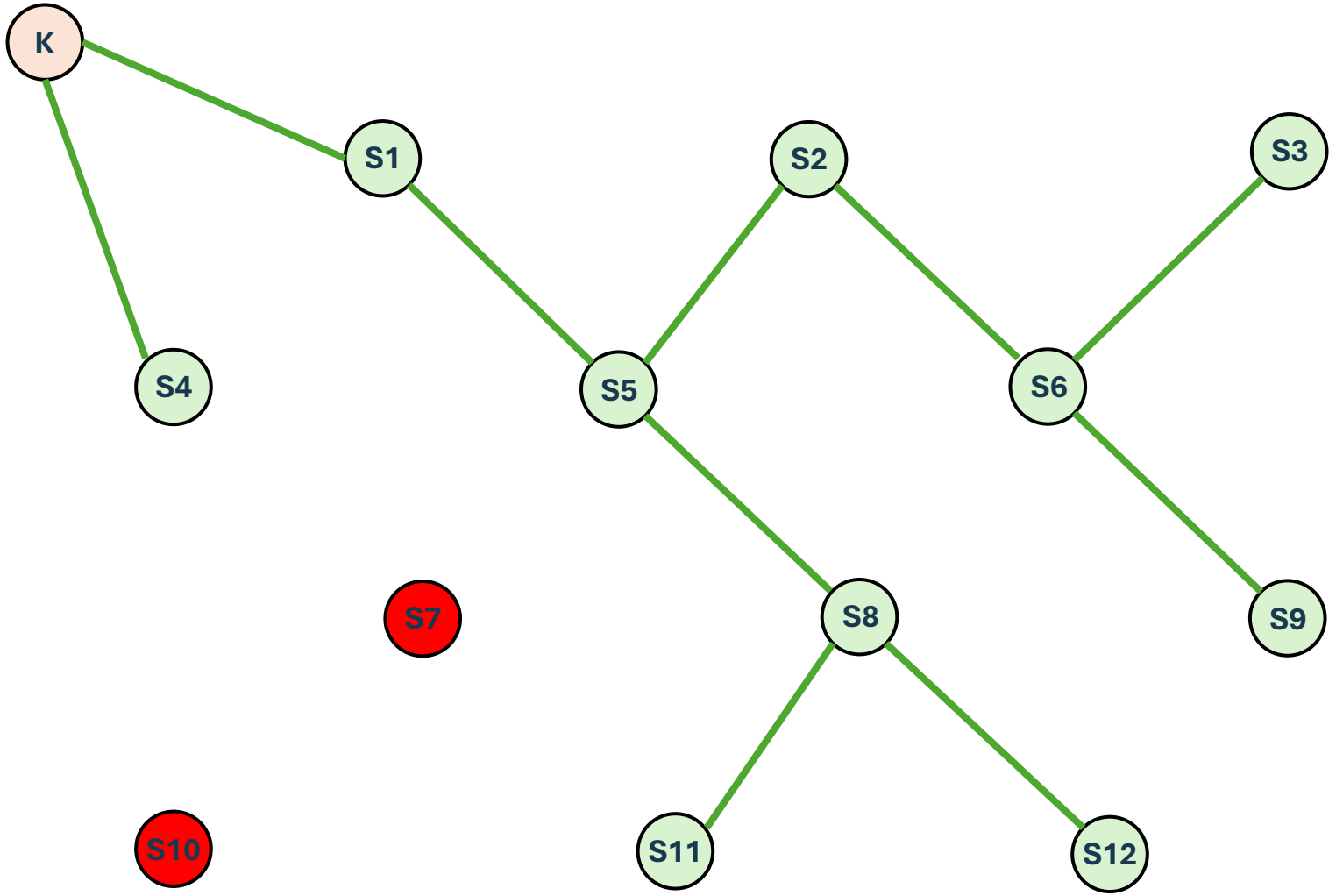
Szenzor kiesés példa 1.



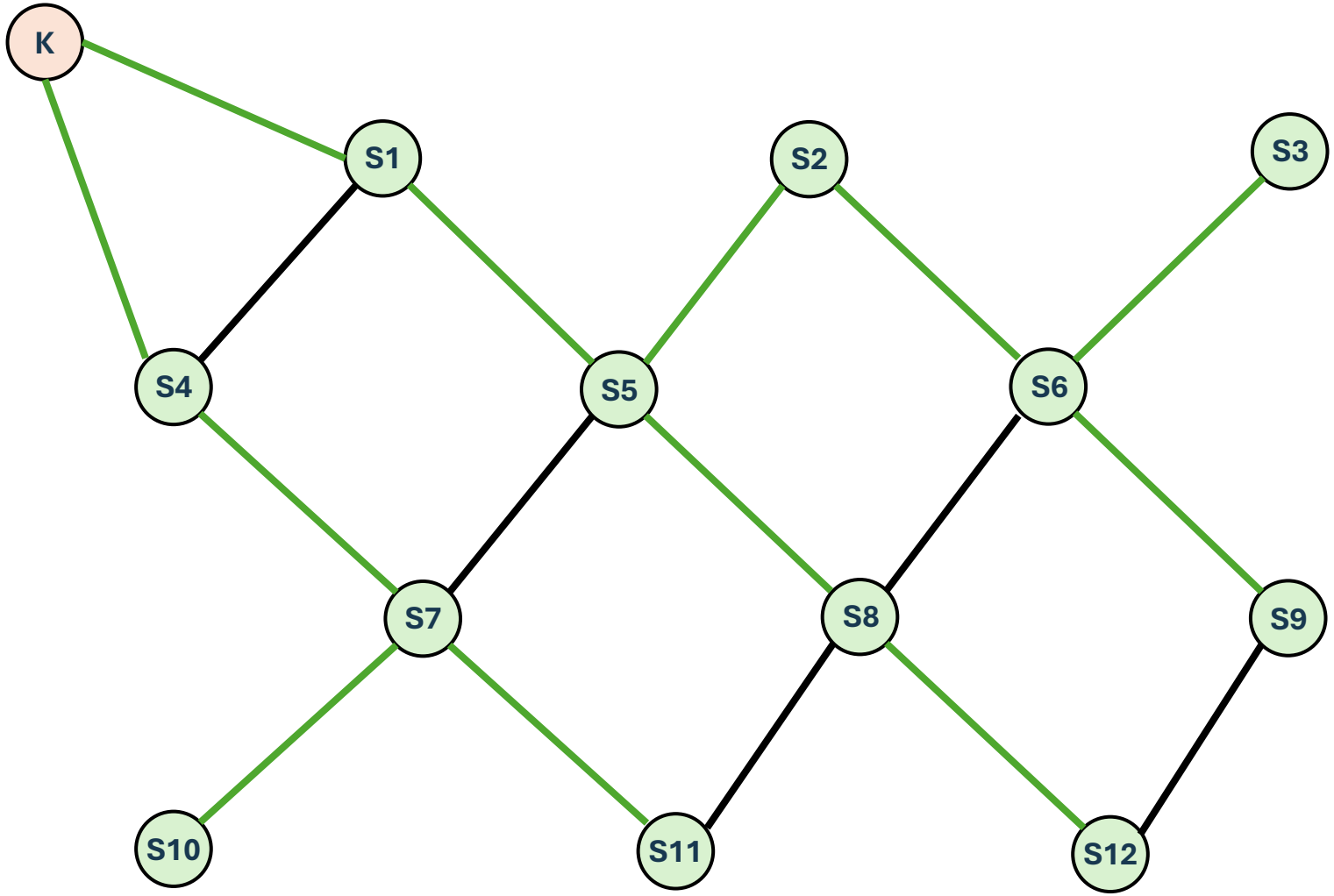
Szenzor kiesés példa 1.



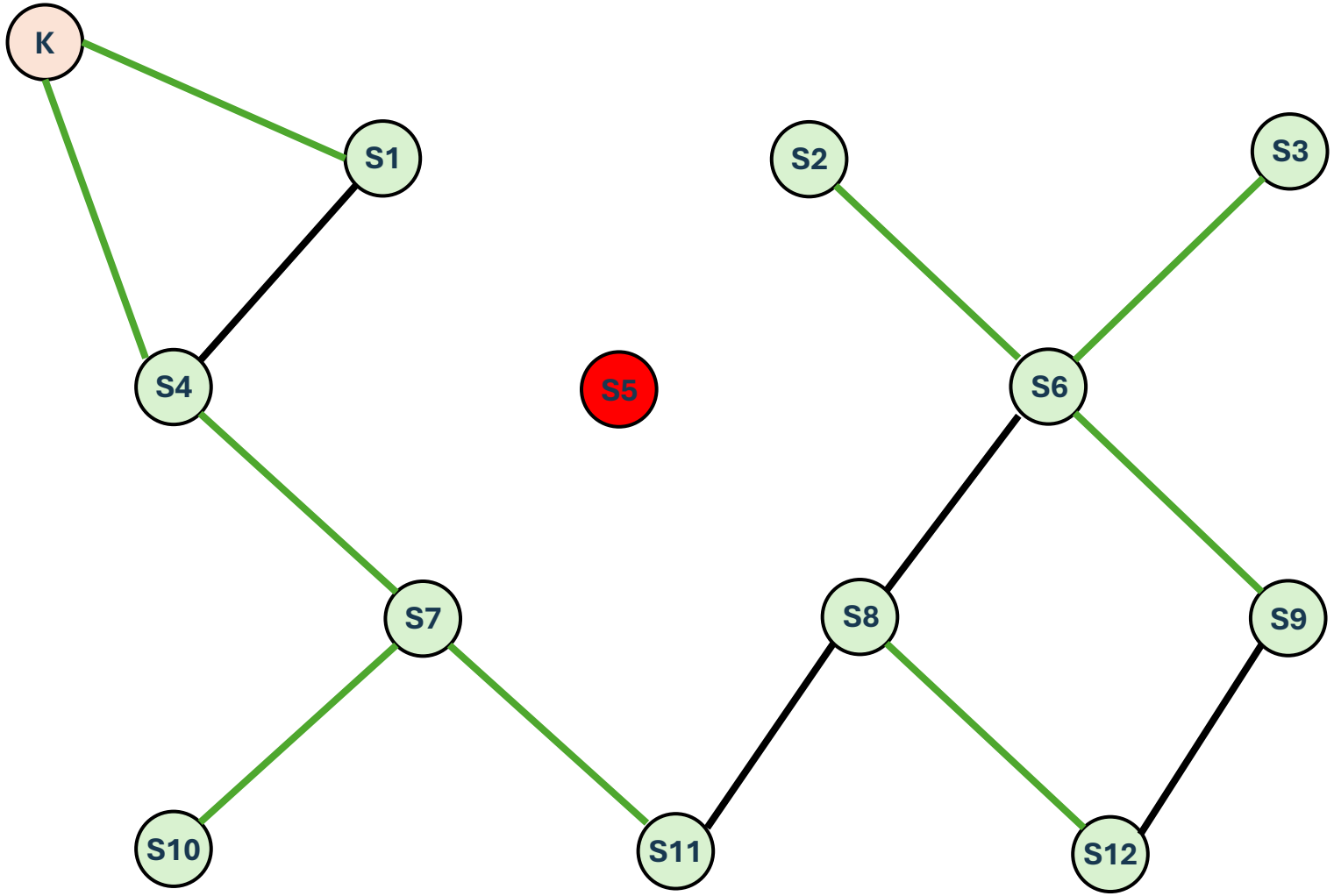
Szenzor kiesés példa 1.



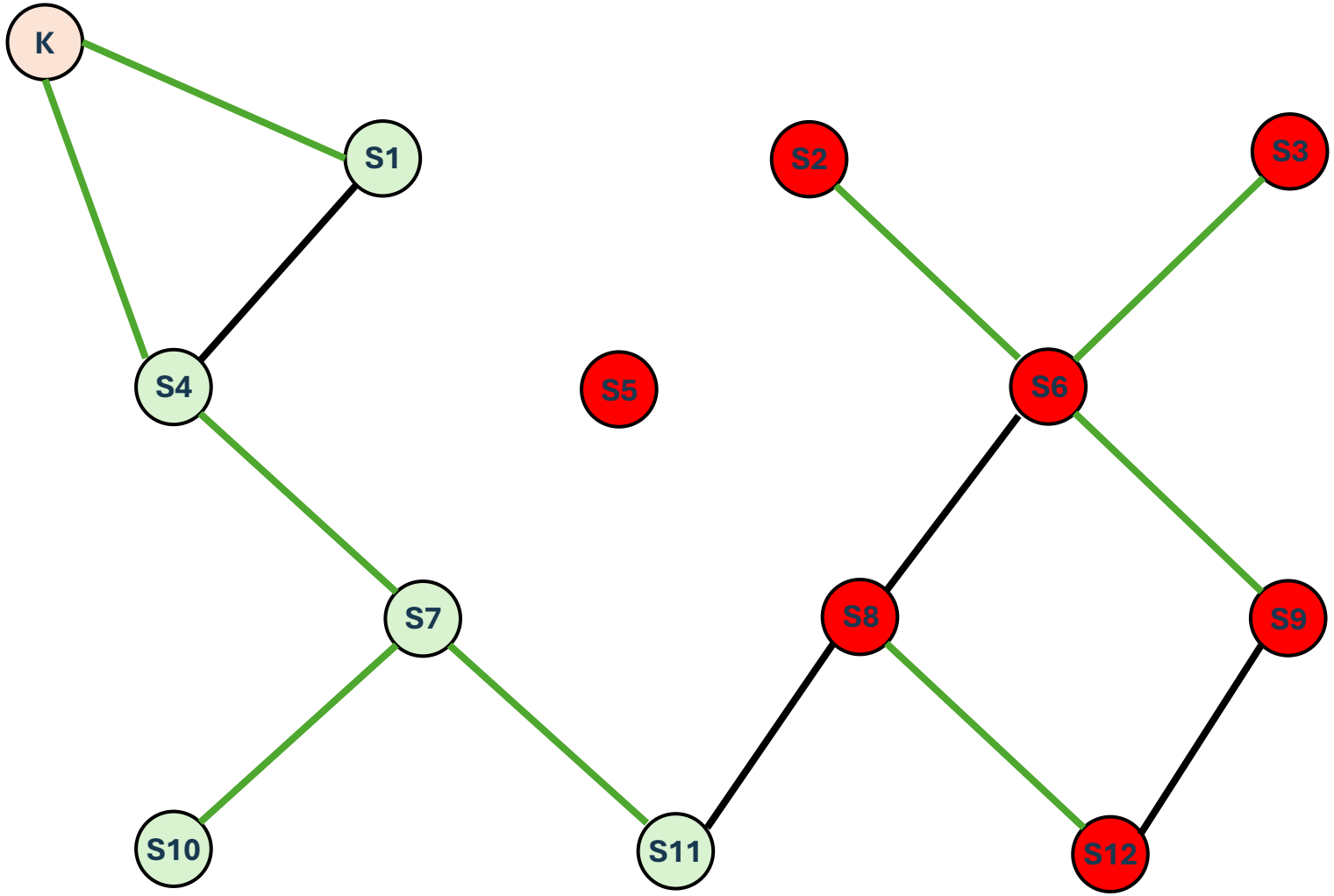
Szenzor kiesés példa 2.



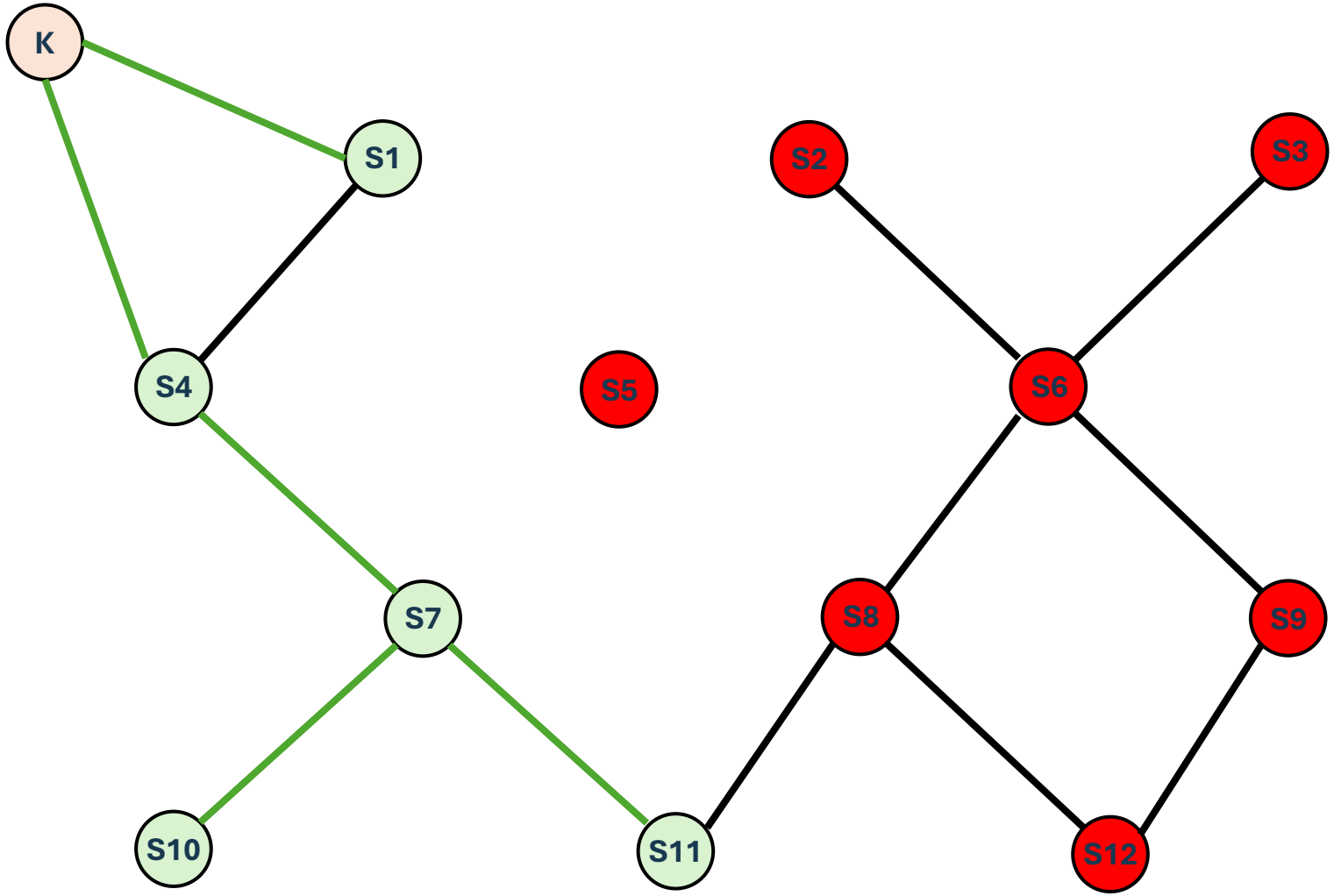
Szenzor kiesés példa 2.



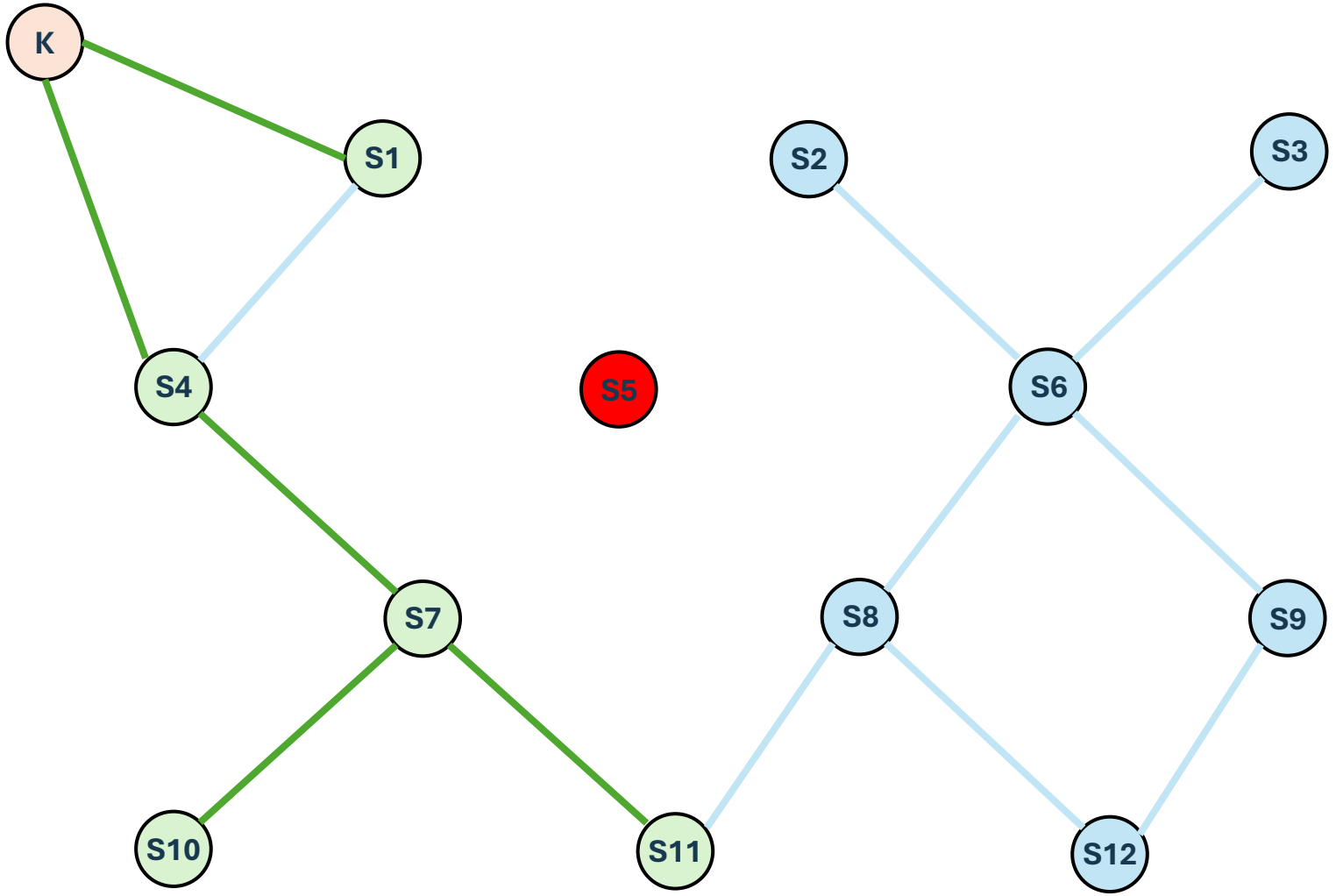
Szenzor kiesés példa 2.



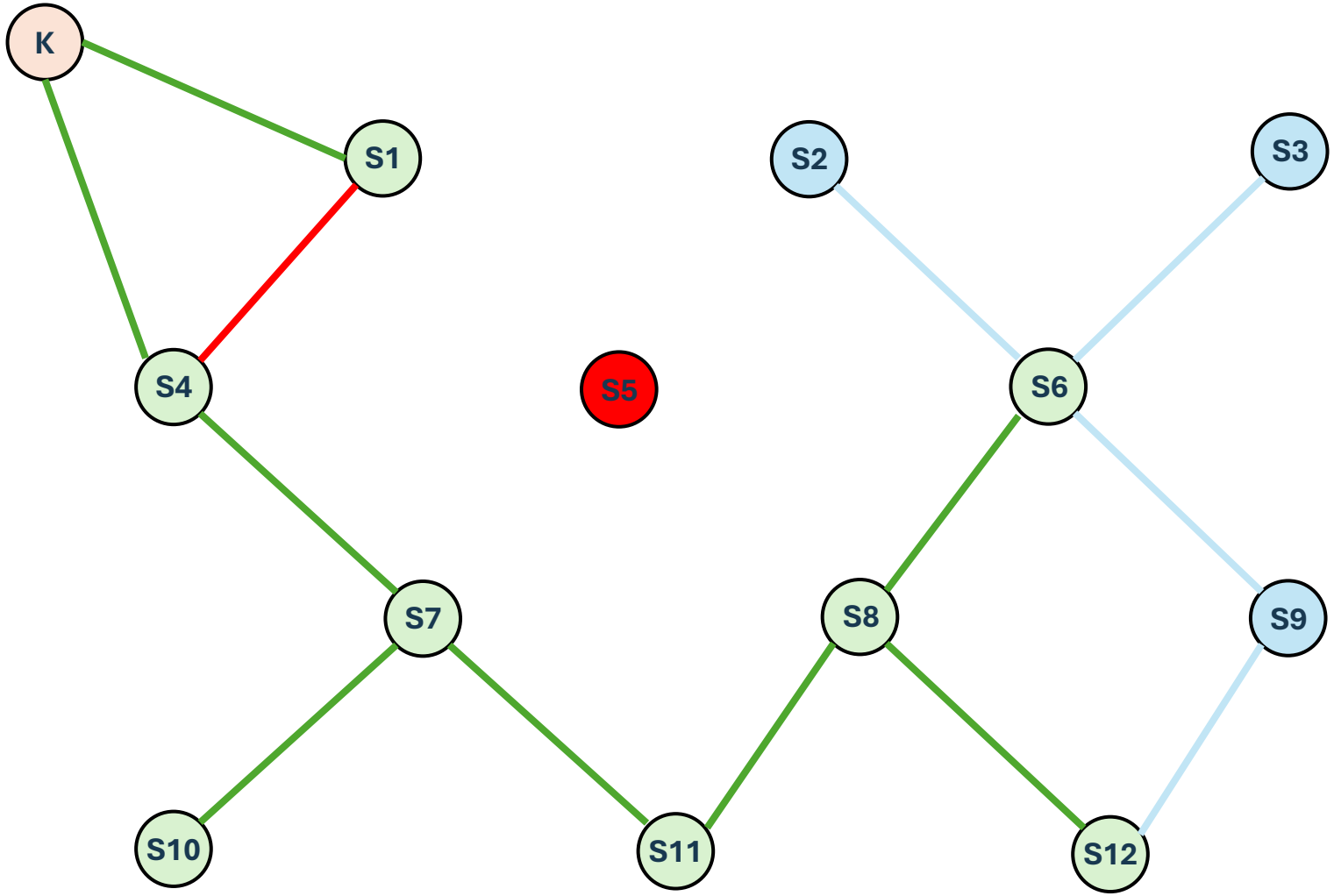
Szenzor kiesés példa 2.



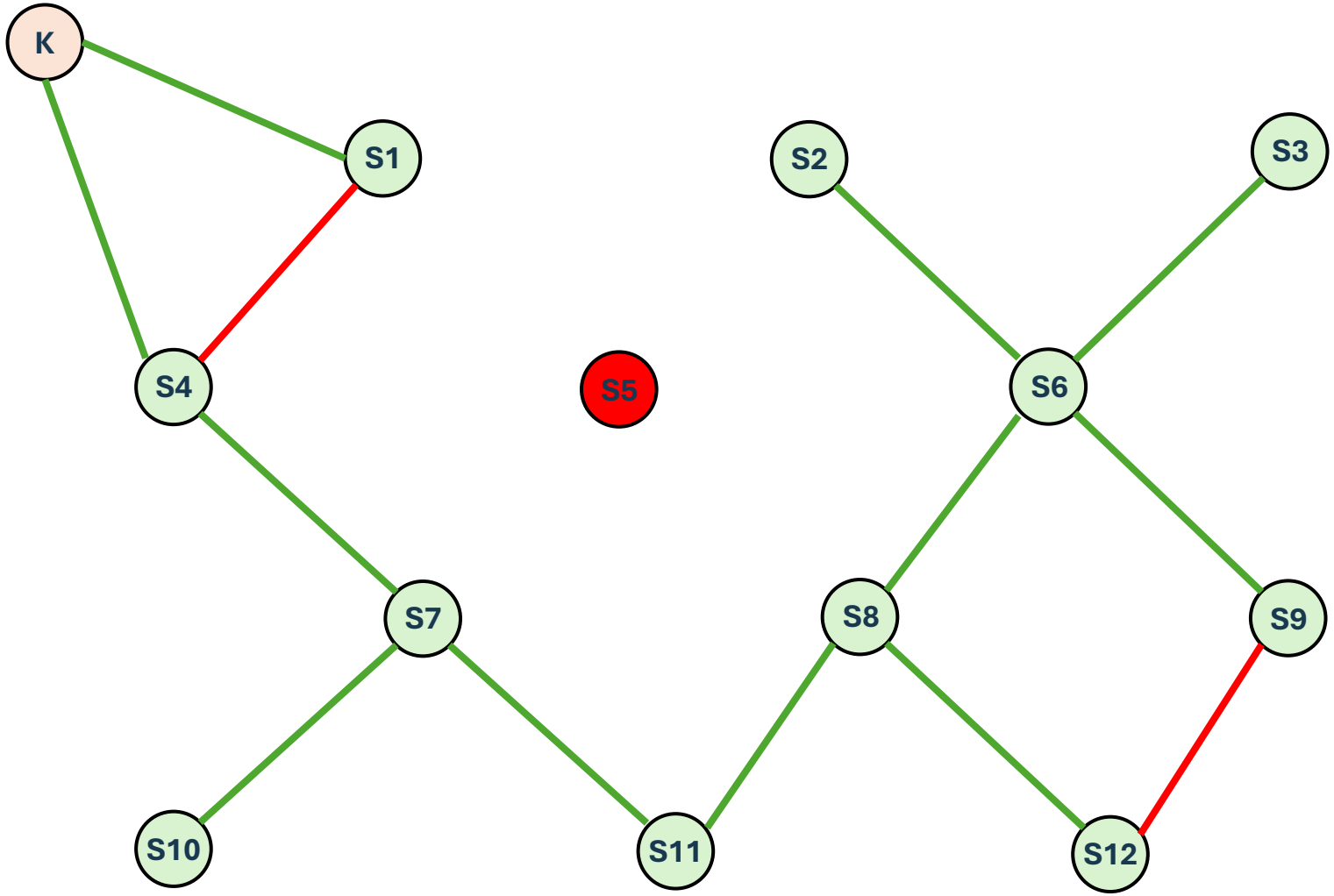
Szenzor kiesés példa 2.



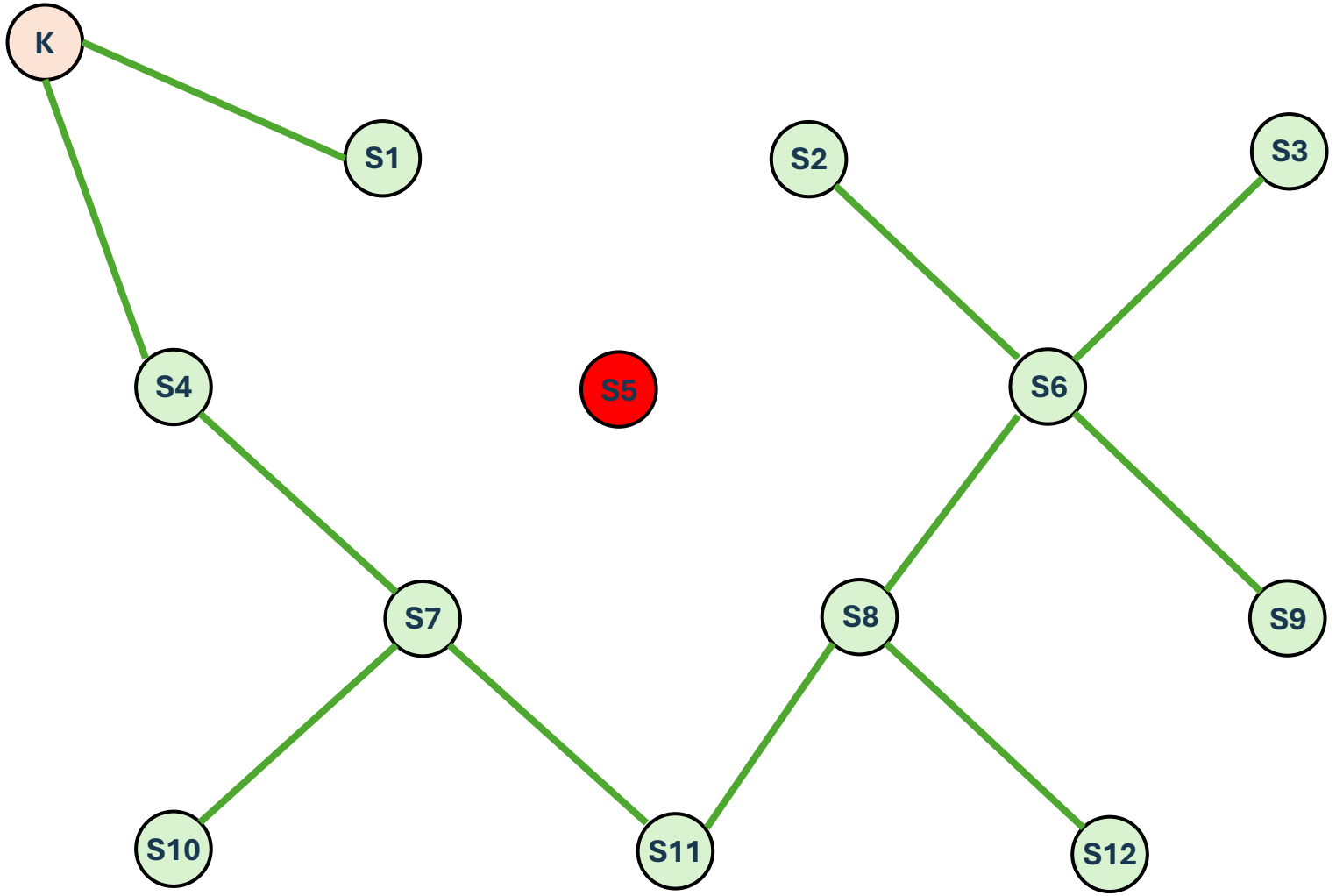
Szenzor kiesés példa 2.



Szenzor kiesés példa 2.



Szenzor kiesés példa 2.



Energiakezelés

A szenzorok napelemes készülékek, így az energiafelhasználást is figyelembe kell venni.

- Akkumulátor szint
- Energiatermelő hatékonyság
- Mérési és adattovábbítási költségek

Cél: Úgy meghatározni a kommunikációs útvonalakat, hogy a hálózat minél tovább, megbízhatóan működőképes maradjon, miközben a szenzorok energiafogyasztását és energiatermelését is figyelembe vesszük.

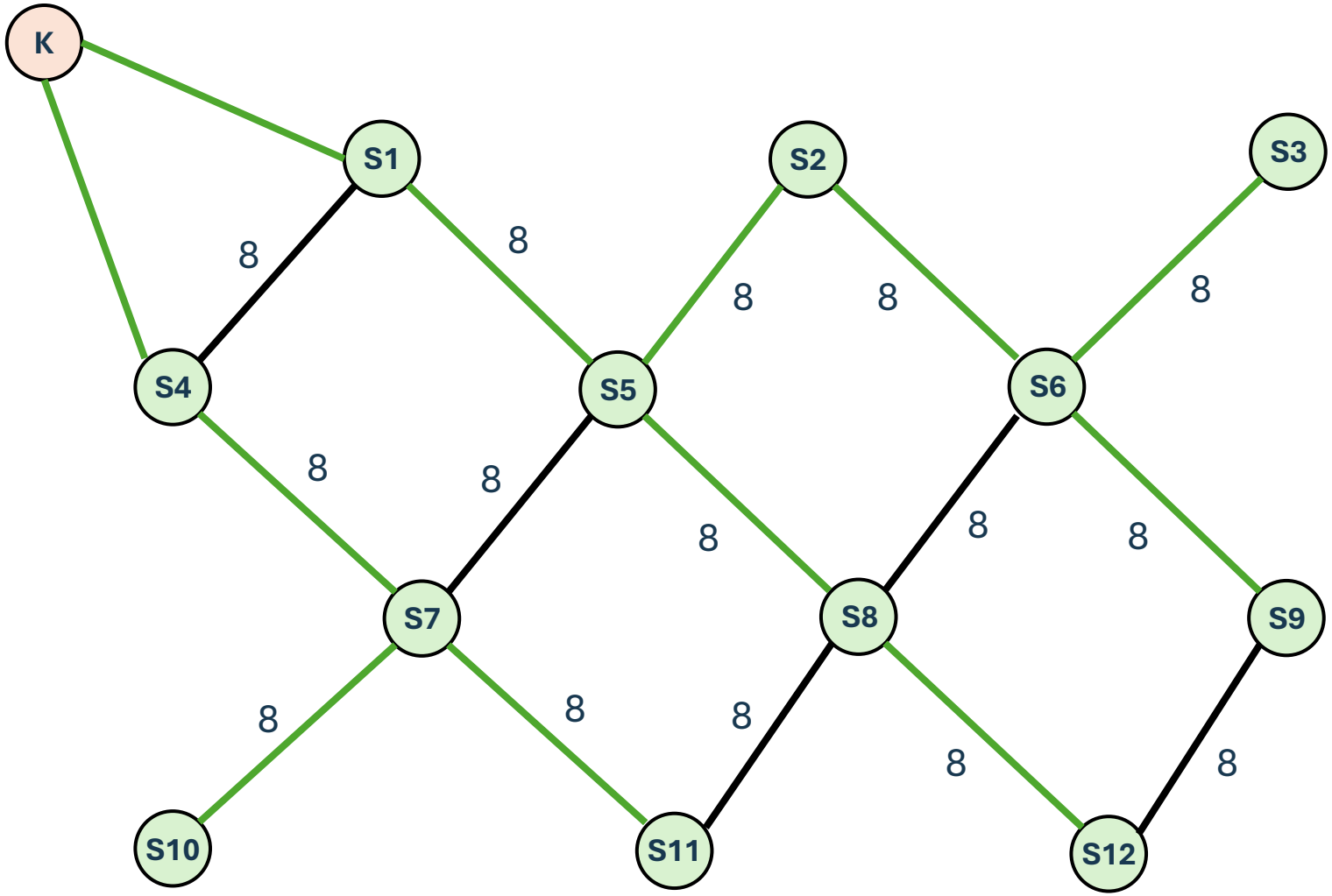
Energiakezelés

Most feltételezhetjük, hogy minden szenzor akkumulátor kapacitása egységes, minden szenzor ugyanazokat a méréseket végzi, aminek ugyanannyi az energiaköltsége. Továbbá az egyes kapcsolat felépítő, és útvonal váltó üzenetek száma elhanyagolható energiafelhasználású.

Először megnézzük egyedül a továbbítási költséggel vizsgálva. Utána a következő lépésben figyelembe vesszük az akkumulátor szinteket, majd a napelemes energiatermeléssel is foglalkozunk.

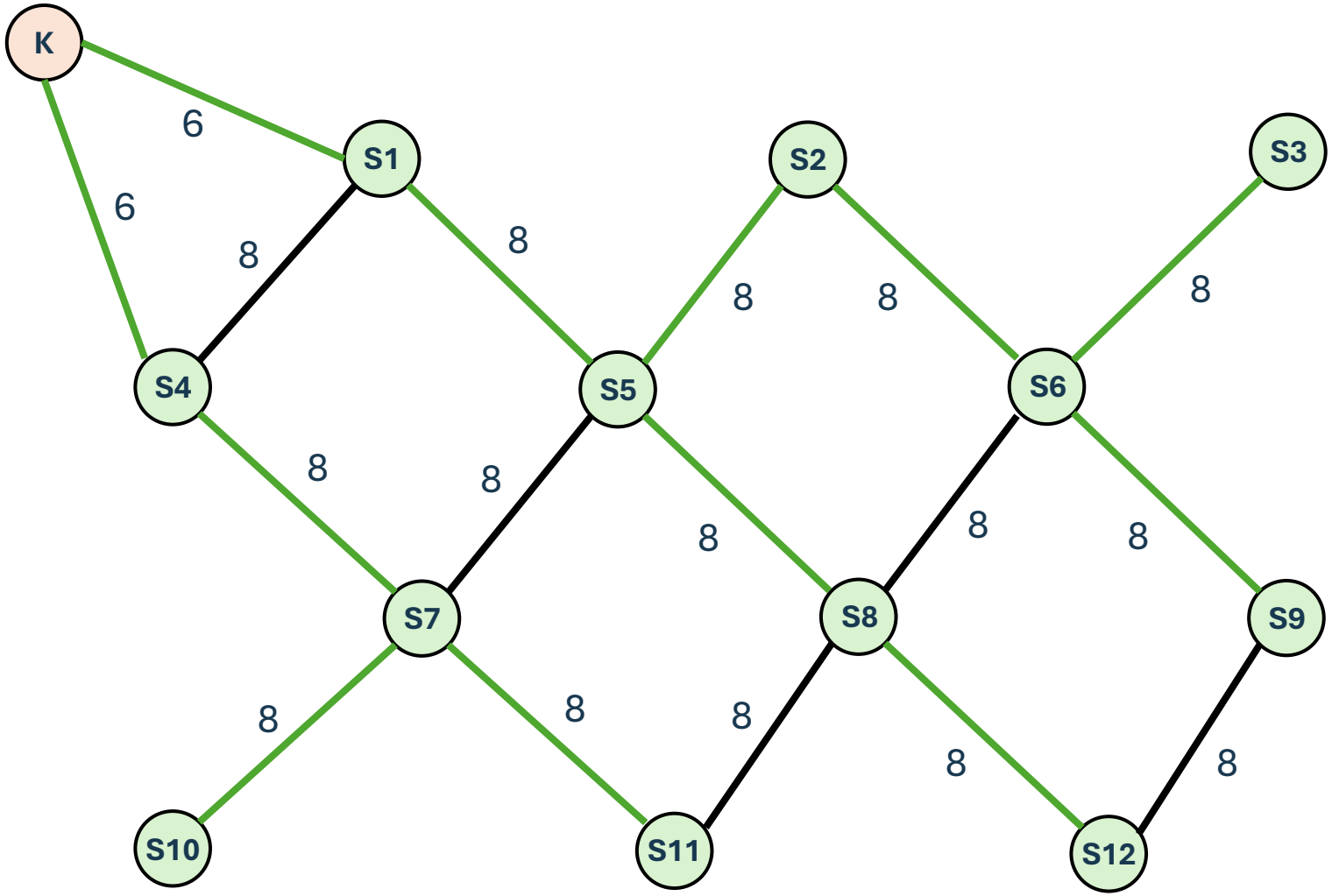
Költségfüggvény 1. verzió

- Egy adattcsomag elküldése (TX): 6
- Egy adattcsomag fogadása (RX): 2
- Egy adattcsomag továbbítása (TX+RX): 8



Költségfüggvény 1. verzió

- Egy adattcsomag elküldése (TX): 6
- Egy adattcsomag fogadása (RX): 2
- Egy adattcsomag továbbítása (TX+RX): 8
- A központban a fogadást elhanyagoljuk: 0



Akkumulátorállapot figyelembevétele

- Minden szenzorhoz hozzárendelünk egy akkumulátor töltöttségi értéket.
- A szenzorok saját akkumulátoruk állapotát képesek mérni vagy becsülni. Ezt az információt a környezeti méréseikkel együtt elküldhetik a központnak.
- A merülő szenzorokon keresztül kevesebb továbbítást kell végeznünk, ehhez az útvonalakat nagyobb költséggel vesszük figyelembe.

Cél: nem kizárólag a legkisebb energiafogyasztásra törekszünk, hanem a merülő szenzorok kímélésére is

Költségfüggvény 2. verzió

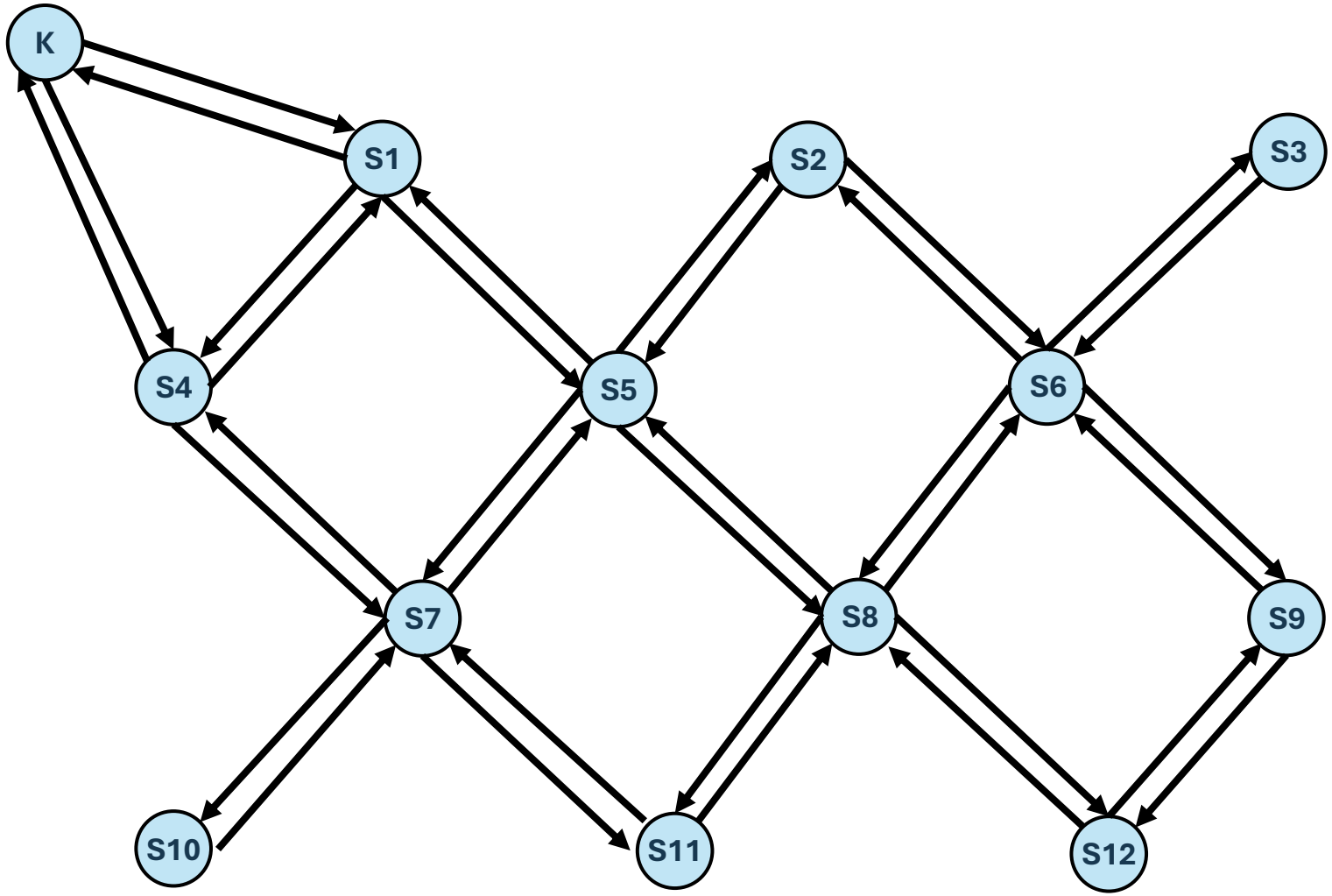
- Korábbról: TX=6, RX=2, $\mathrm{RX}_{\mathrm{központ}} = 0$
- $\mathrm{b_j}$: j szenzor töltöttségi százaléka
- λ: töltöttségi együttható, legyen most 0.2

$$\omega_{ij} = \mathrm{TX_i} + \mathrm{RX_j} + \lambda\left(100 - \mathrm{b_j}\right)$$

- A központ hálózati áramról megy, 100%-os akkumulátorral rendelkezőnek tekintjük.
- A vevő töltöttségét nézzük, irányított élek kellenek. A rádió kapcsolat oda-vissza működik, így minden élet le tudunk cserélni 2 irányított élre.

Dijkstra-algoritmus

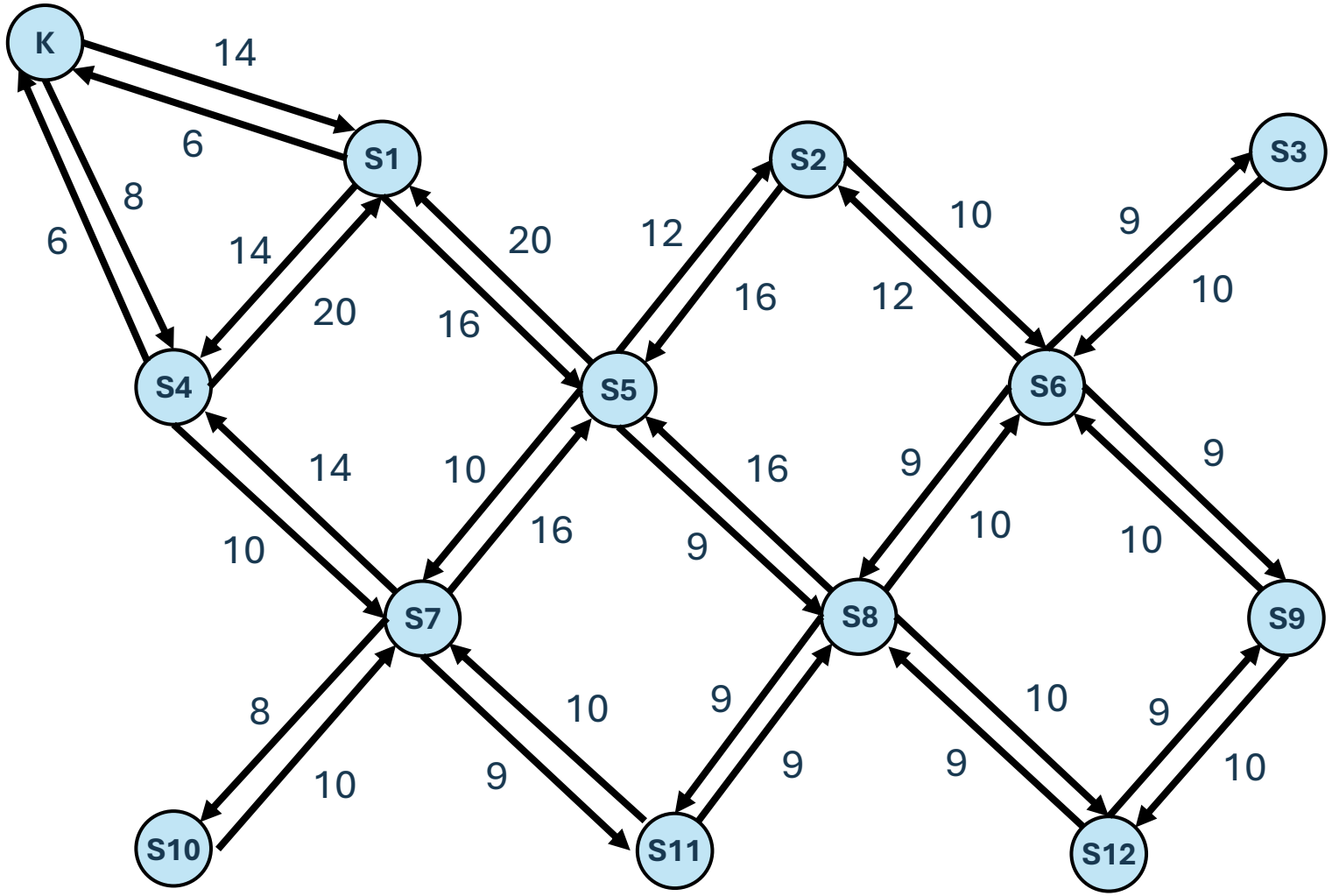
$$\omega_{ij} = TX_i + RX_j + \lambda(100 - b_j)$$



Dijkstra-algoritmus

$$\omega_{ij} = TX_i + RX_j + \lambda(100 - b_j)$$

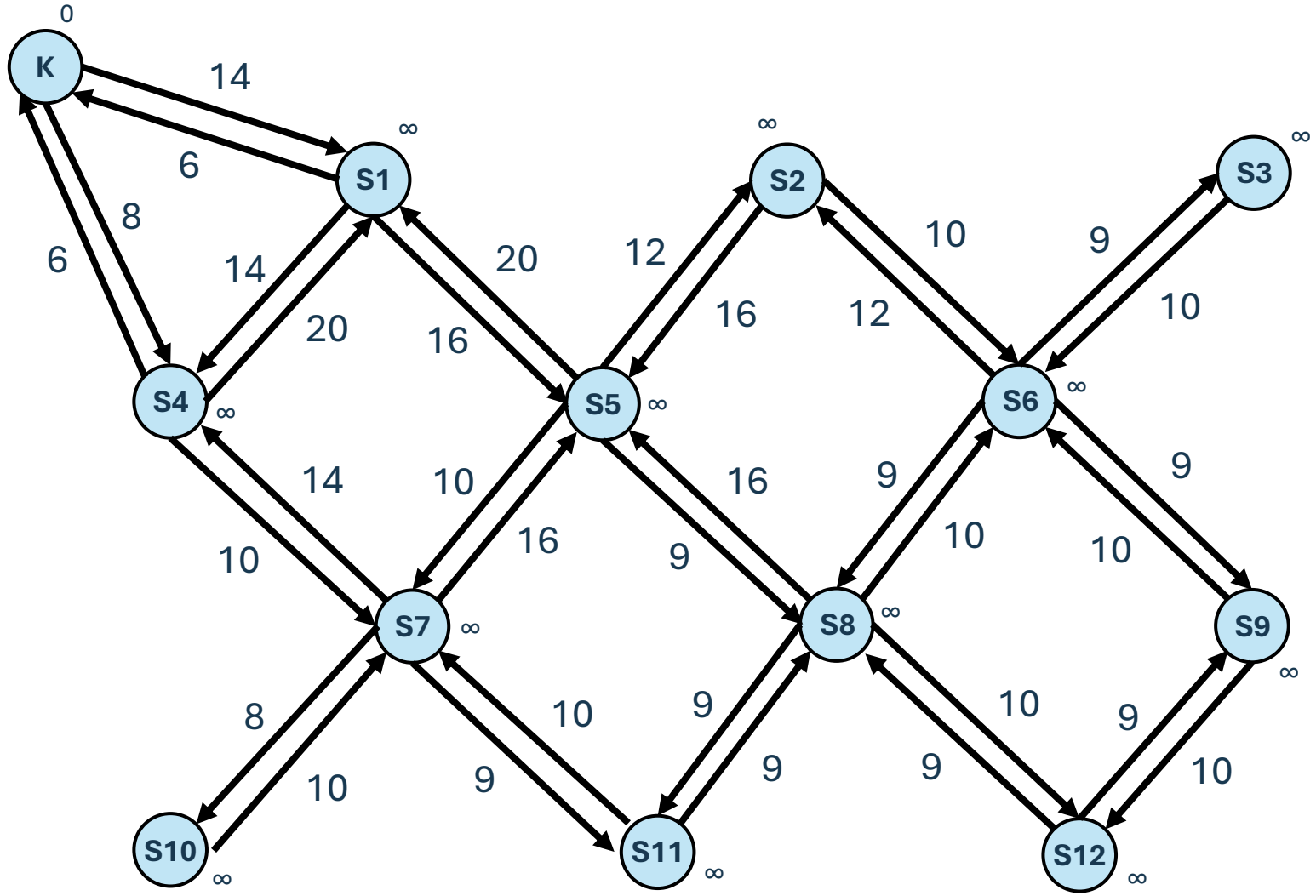
S1: 40%, S2: 80%, S3: 95%,
S4: 70%, S5: 60%, S6: 90%,
S7: 90%, S8: 95%, S9: 95%,
S10: 100%, S11: 95%,
S12: 90%, K: 100%



Dijkstra-algoritmus

$$\omega_{ij} = TX_i + RX_j + \lambda(100 - b_j)$$

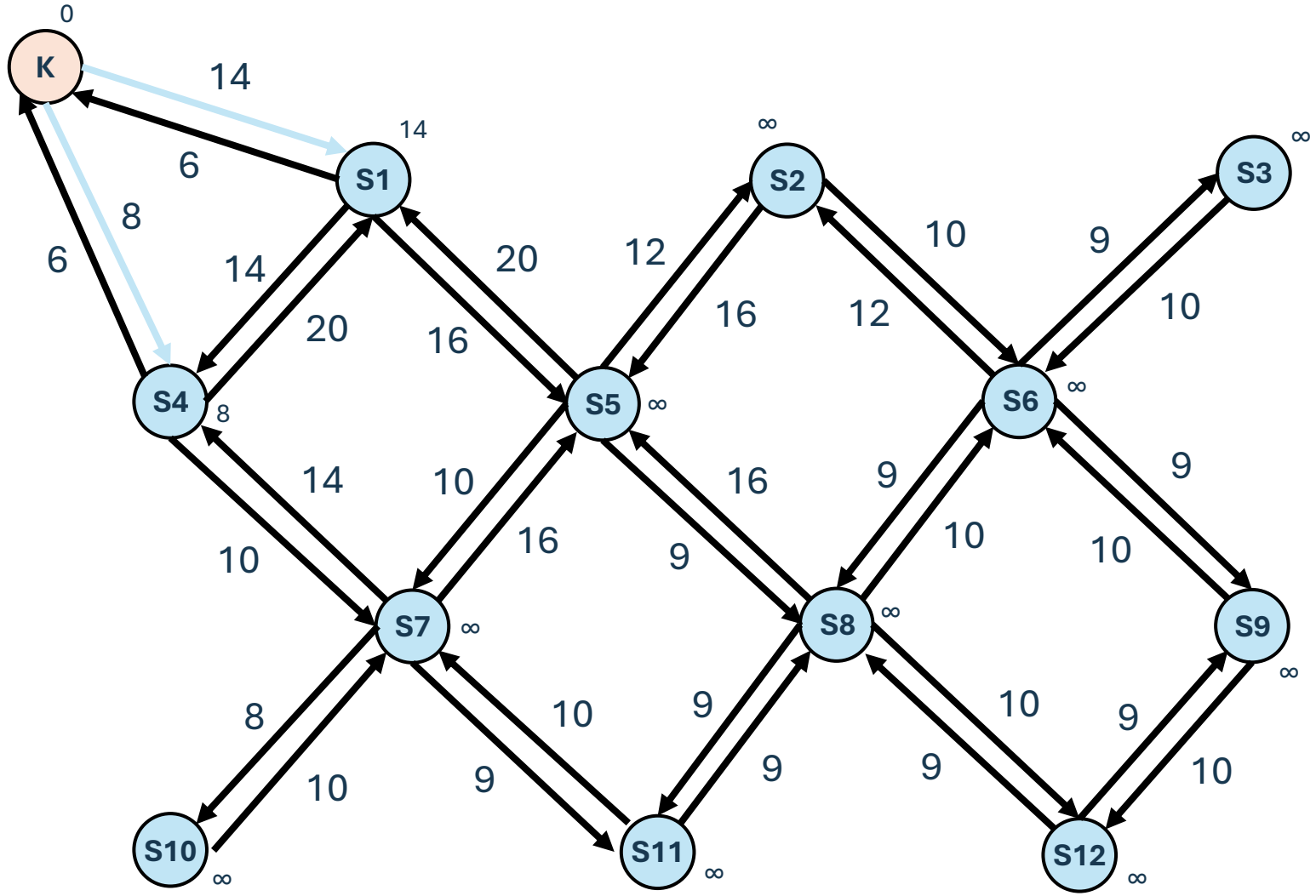
S1: 40%, S2: 80%, S3: 95%,
S4: 70%, S5: 60%, S6: 90%,
S7: 90%, S8: 95%, S9: 95%,
S10: 100%, S11: 95%,
S12: 90%, K: 100%



Dijkstra-algoritmus

$$\omega_{ij} = TX_i + RX_j + \lambda(100 - b_j)$$

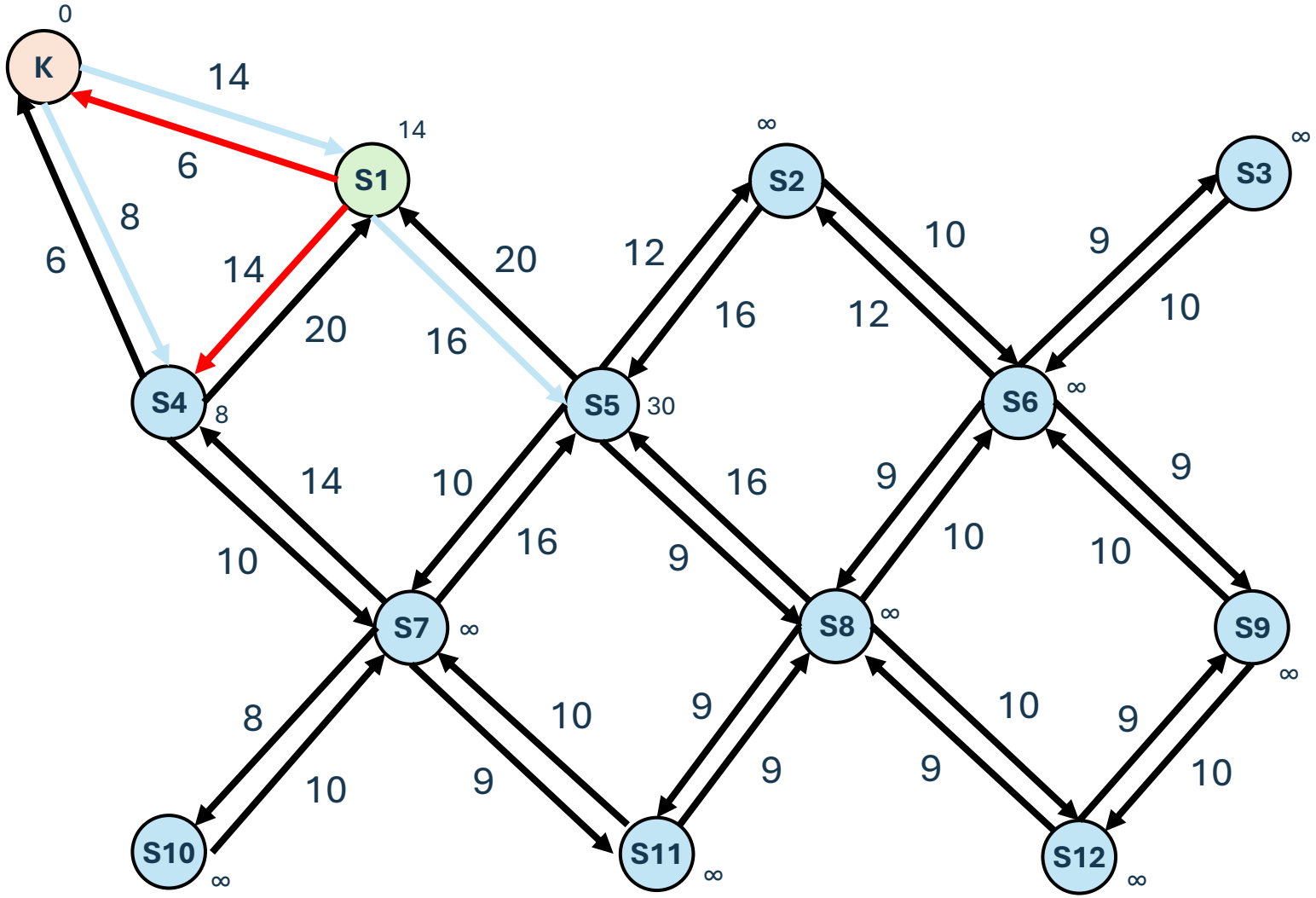
S1: 40%, S2: 80%, S3: 95%,
S4: 70%, S5: 60%, S6: 90%,
S7: 90%, S8: 95%, S9: 95%,
S10: 100%, S11: 95%,
S12: 90%, K: 100%



Dijkstra-algoritmus

$$\omega_{ij} = TX_i + RX_j + \lambda(100 - b_j)$$

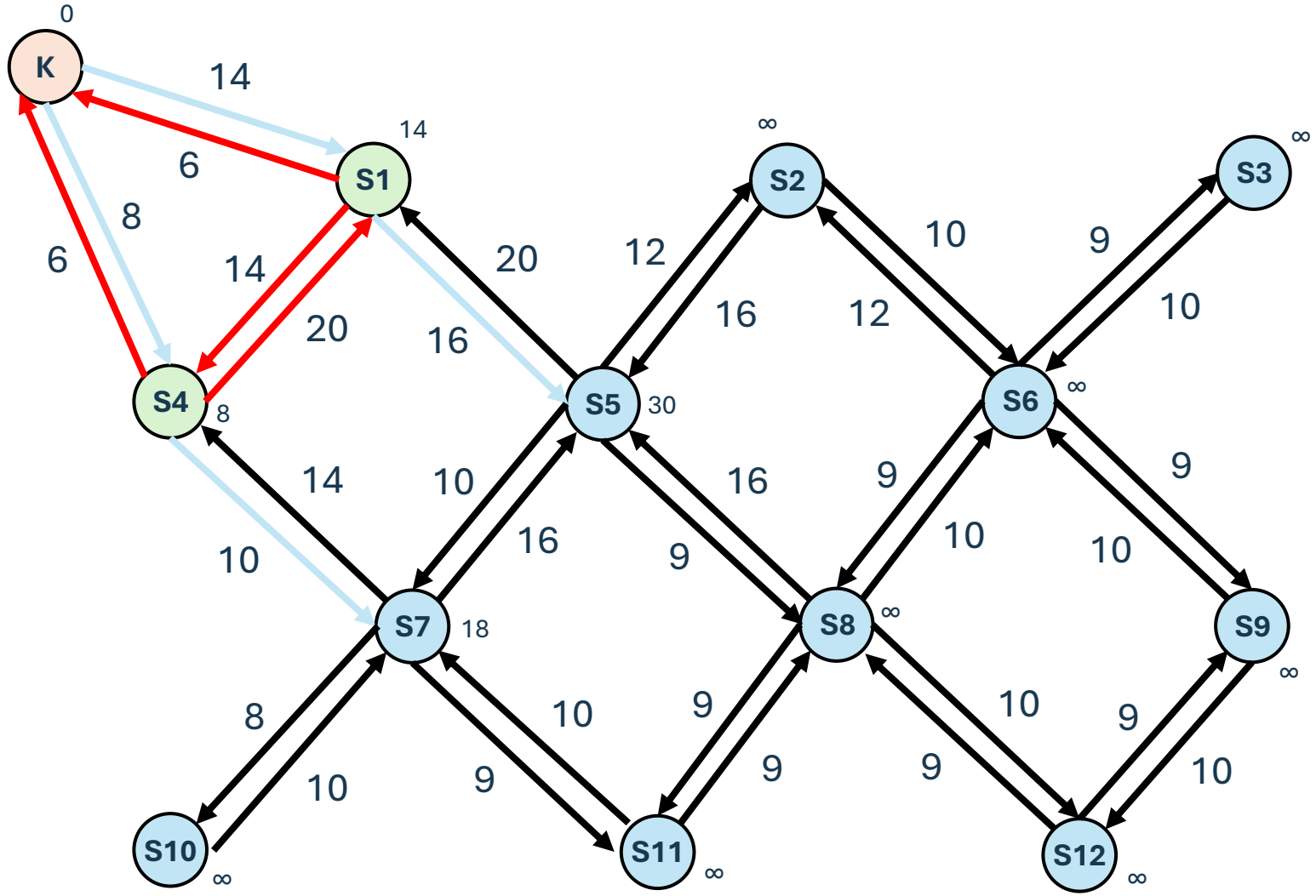
S1: 40%, S2: 80%, S3: 95%,
S4: 70%, S5: 60%, S6: 90%,
S7: 90%, S8: 95%, S9: 95%,
S10: 100%, S11: 95%,
S12: 90%, K: 100%



Dijkstra-algoritmus

$$\omega_{ij} = TX_i + RX_j + \lambda(100 - b_j)$$

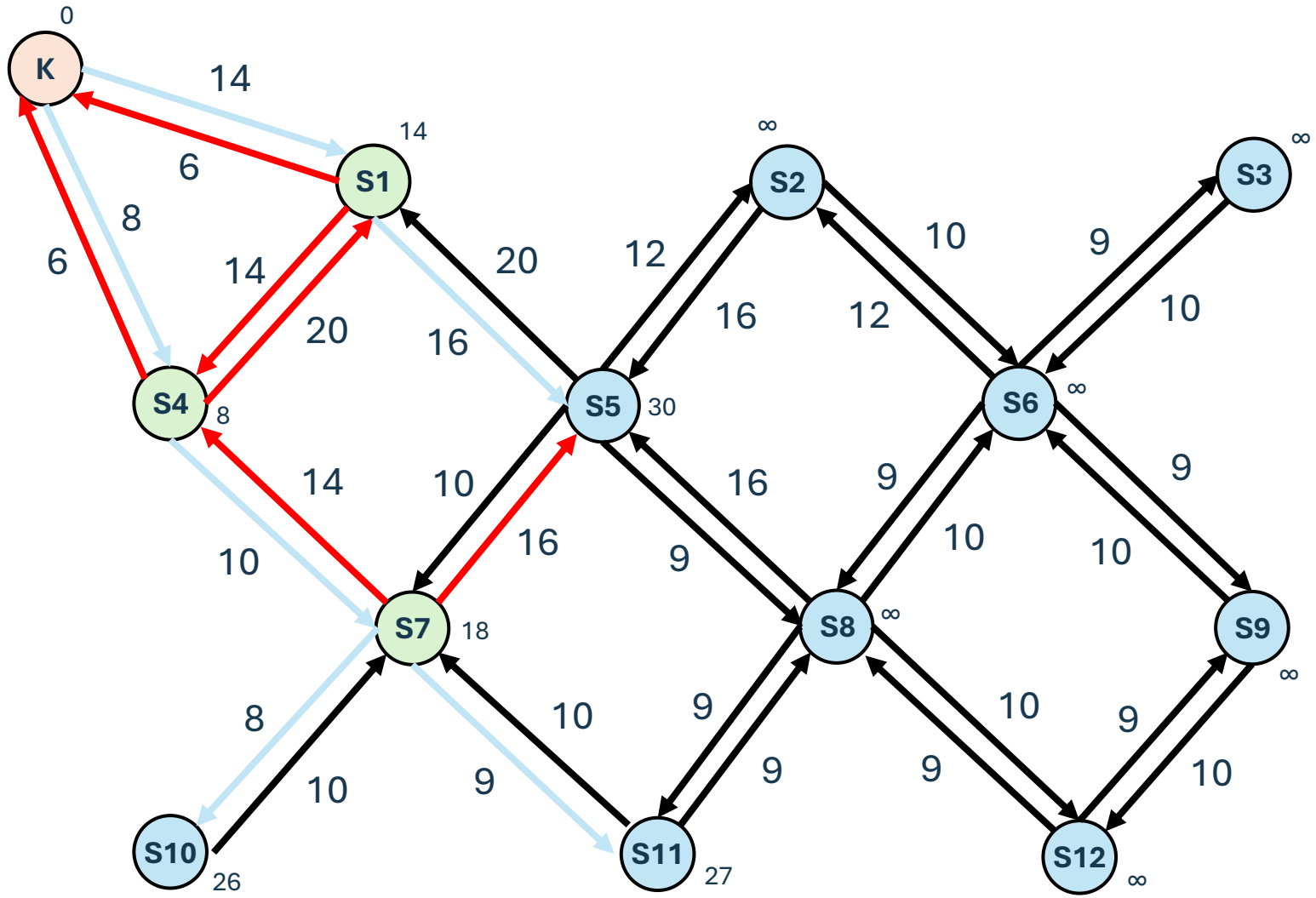
S1: 40%, S2: 80%, S3: 95%,
S4: 70%, S5: 60%, S6: 90%,
S7: 90%, S8: 95%, S9: 95%,
S10: 100%, S11: 95%,
S12: 90%, K: 100%



Dijkstra-algoritmus

$$\omega_{ij} = TX_i + RX_j + \lambda(100 - b_j)$$

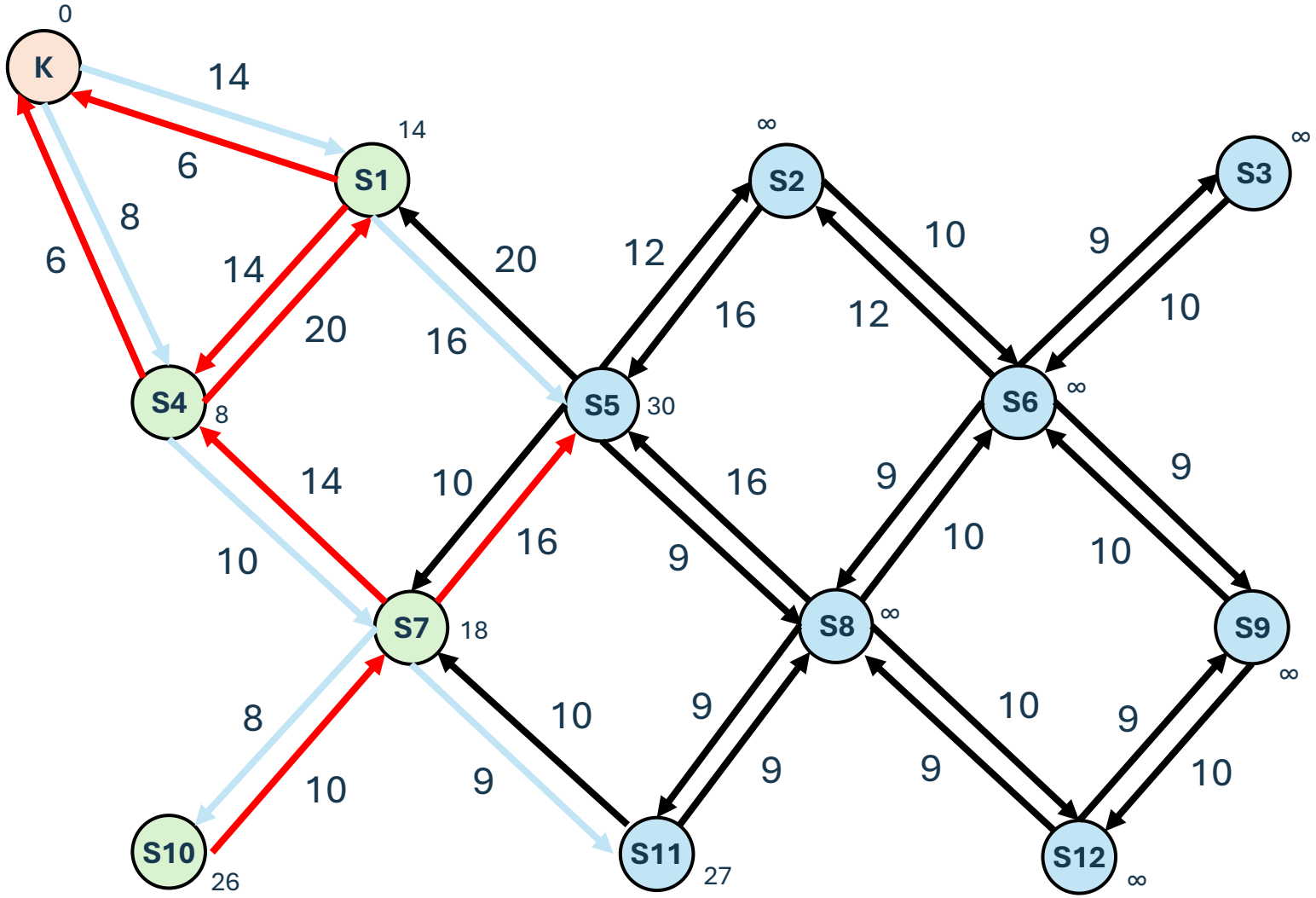
S1: 40%, S2: 80%, S3: 95%,
S4: 70%, S5: 60%, S6: 90%,
S7: 90%, S8: 95%, S9: 95%,
S10: 100%, S11: 95%,
S12: 90%, K: 100%



Dijkstra-algoritmus

$$\omega_{ij} = TX_i + RX_j + \lambda(100 - b_j)$$

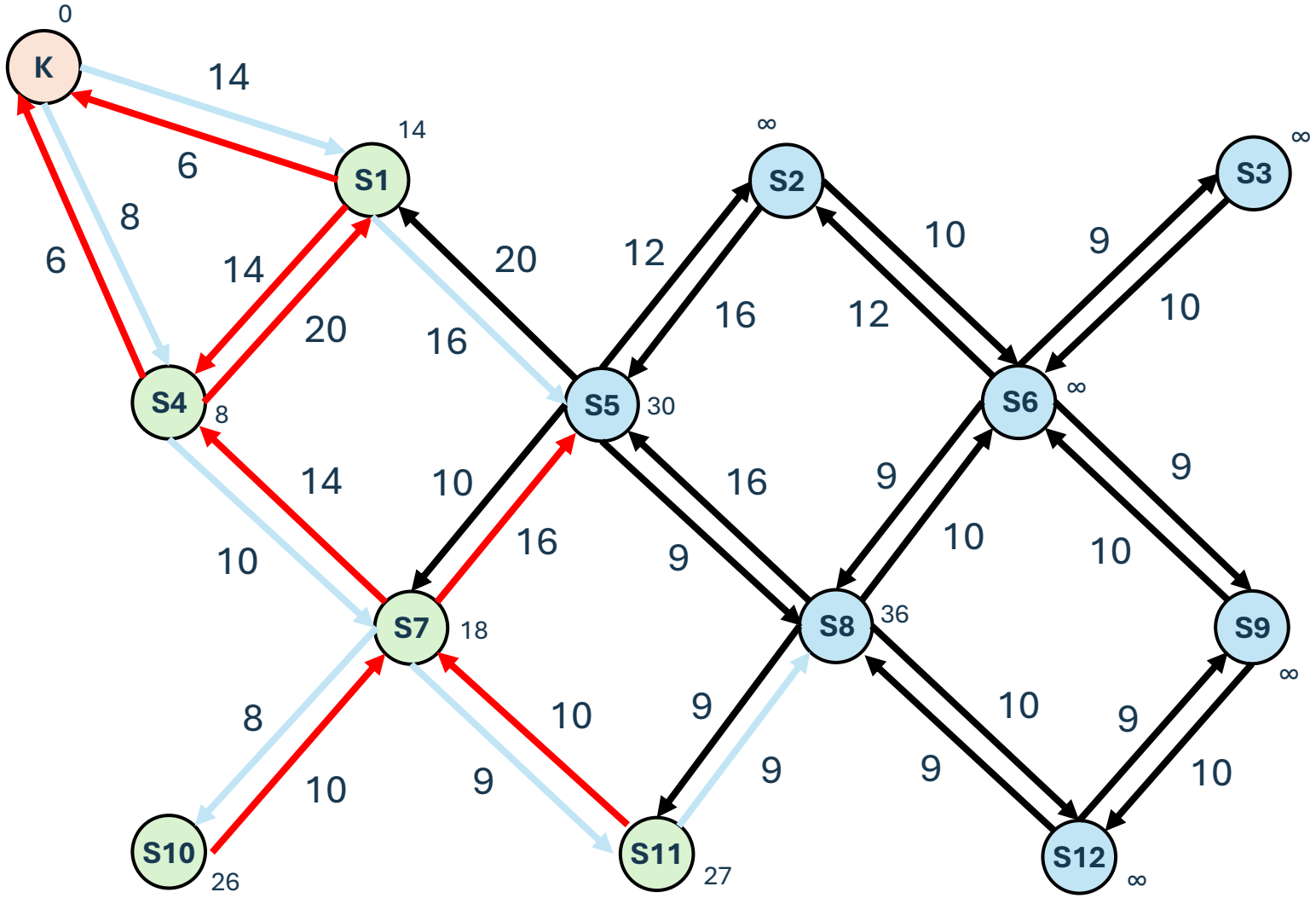
S1: 40%, S2: 80%, S3: 95%,
S4: 70%, S5: 60%, S6: 90%,
S7: 90%, S8: 95%, S9: 95%,
S10: 100%, S11: 95%,
S12: 90%, K: 100%



Dijkstra-algoritmus

$$\omega_{ij} = TX_i + RX_j + \lambda(100 - b_j)$$

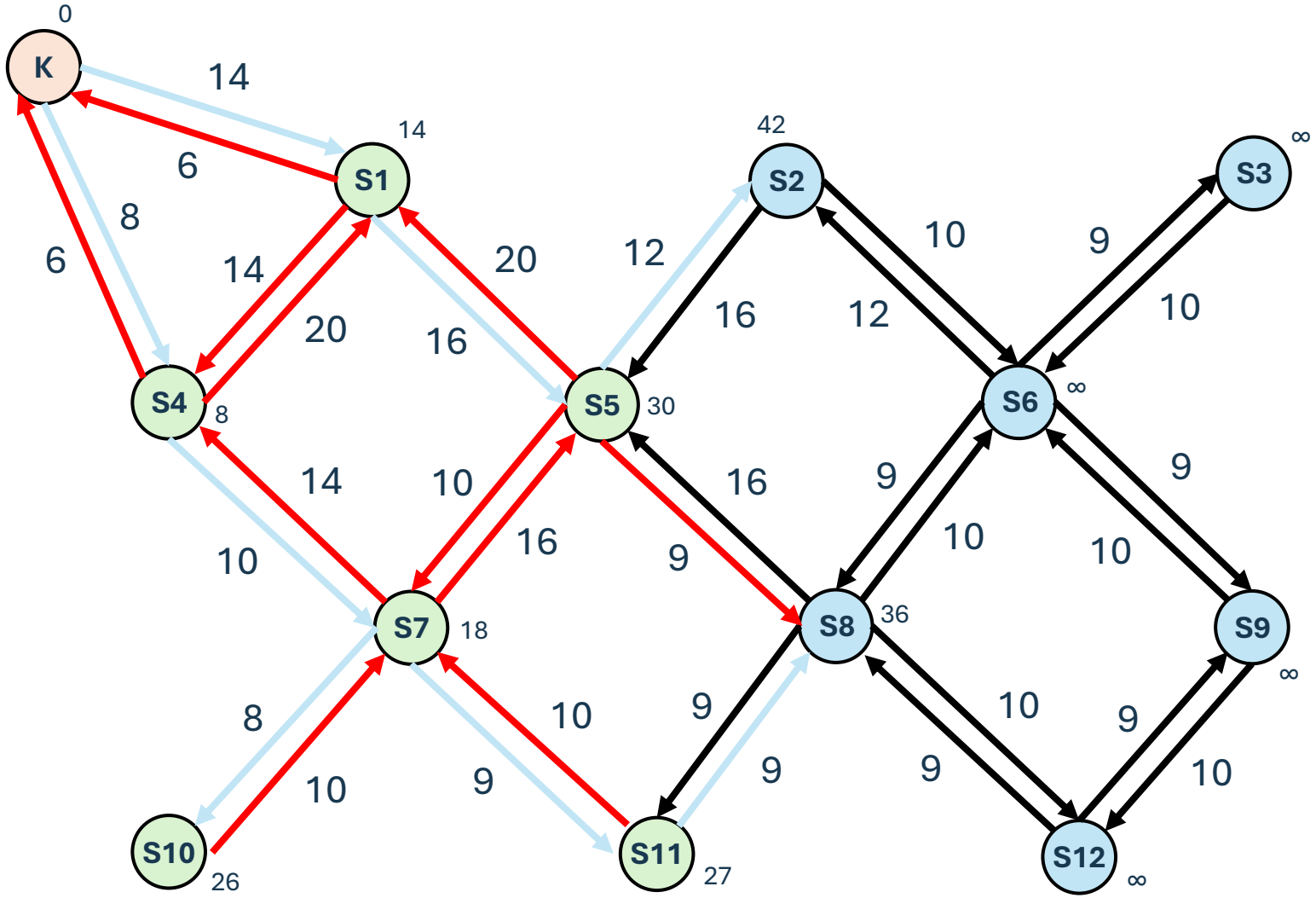
S1: 40%, S2: 80%, S3: 95%,
S4: 70%, S5: 60%, S6: 90%,
S7: 90%, S8: 95%, S9: 95%,
S10: 100%, S11: 95%,
S12: 90%, K: 100%



Dijkstra-algoritmus

$$\omega_{ij} = TX_i + RX_j + \lambda(100 - b_j)$$

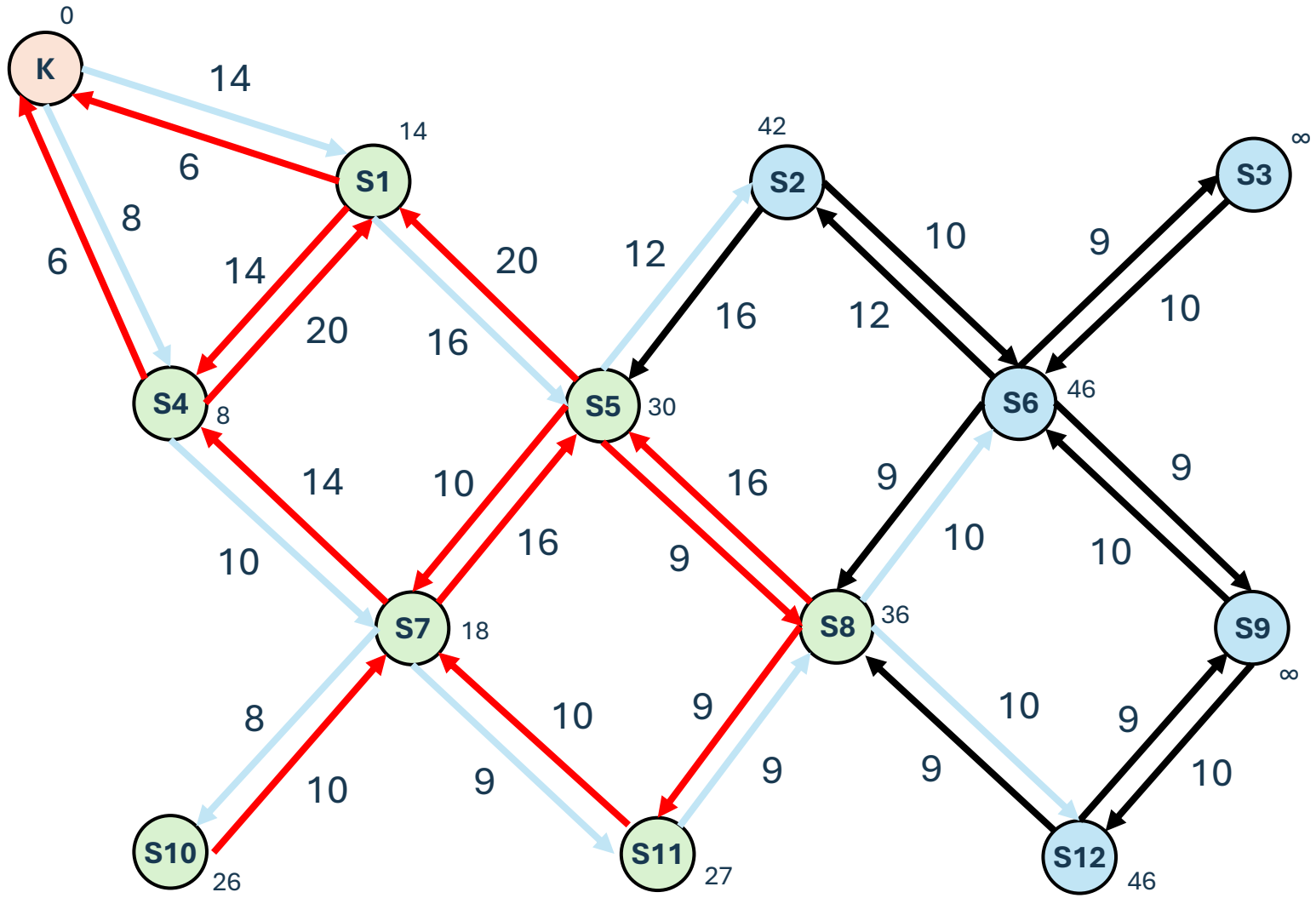
S1: 40%, S2: 80%, S3: 95%,
S4: 70%, S5: 60%, S6: 90%,
S7: 90%, S8: 95%, S9: 95%,
S10: 100%, S11: 95%,
S12: 90%, K: 100%



Dijkstra-algoritmus

$$\omega_{ij} = TX_i + RX_j + \lambda(100 - b_j)$$

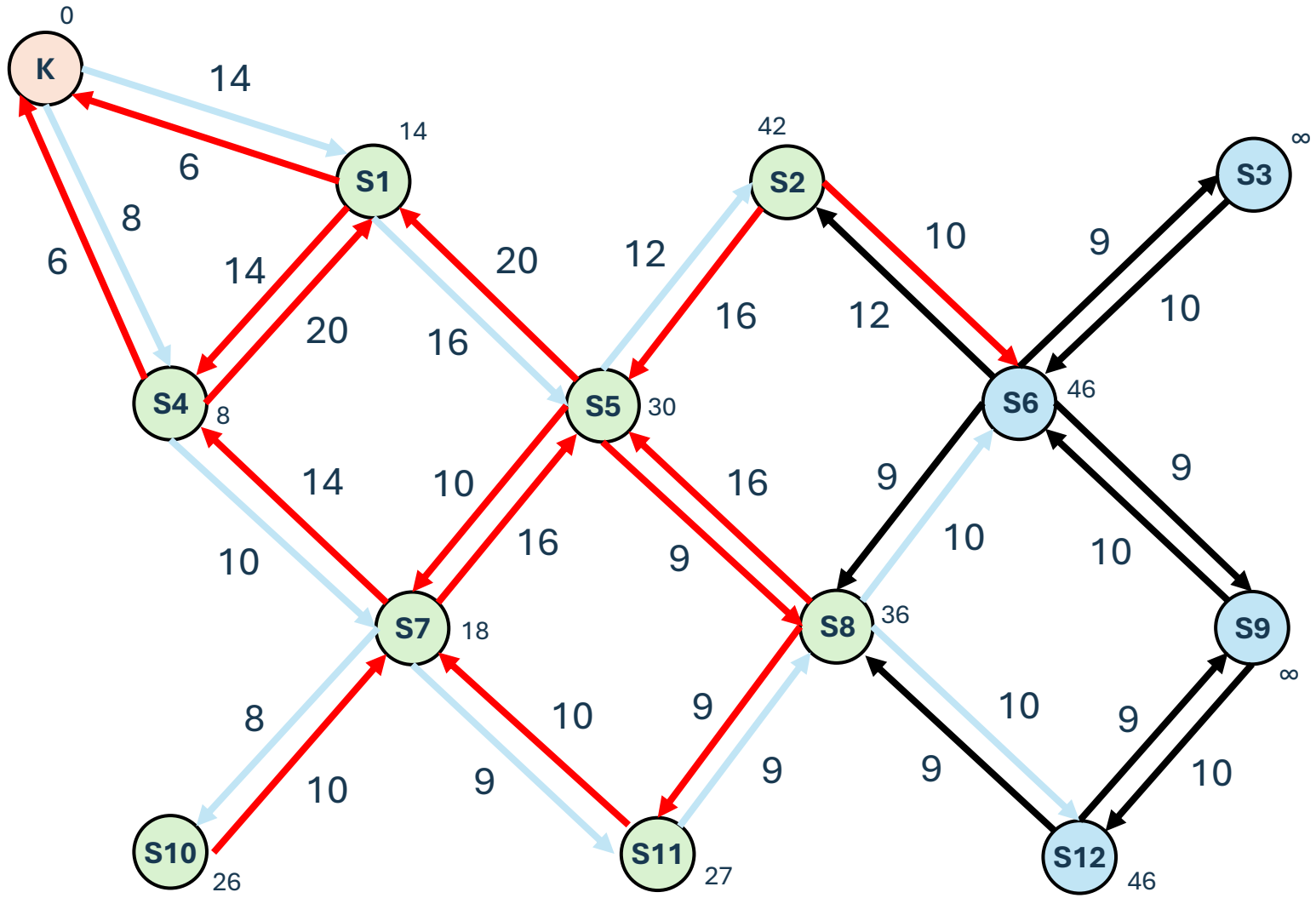
S1: 40%, S2: 80%, S3: 95%,
S4: 70%, S5: 60%, S6: 90%,
S7: 90%, S8: 95%, S9: 95%,
S10: 100%, S11: 95%,
S12: 90%, K: 100%



Dijkstra-algoritmus

$$\omega_{ij} = TX_i + RX_j + \lambda(100 - b_j)$$

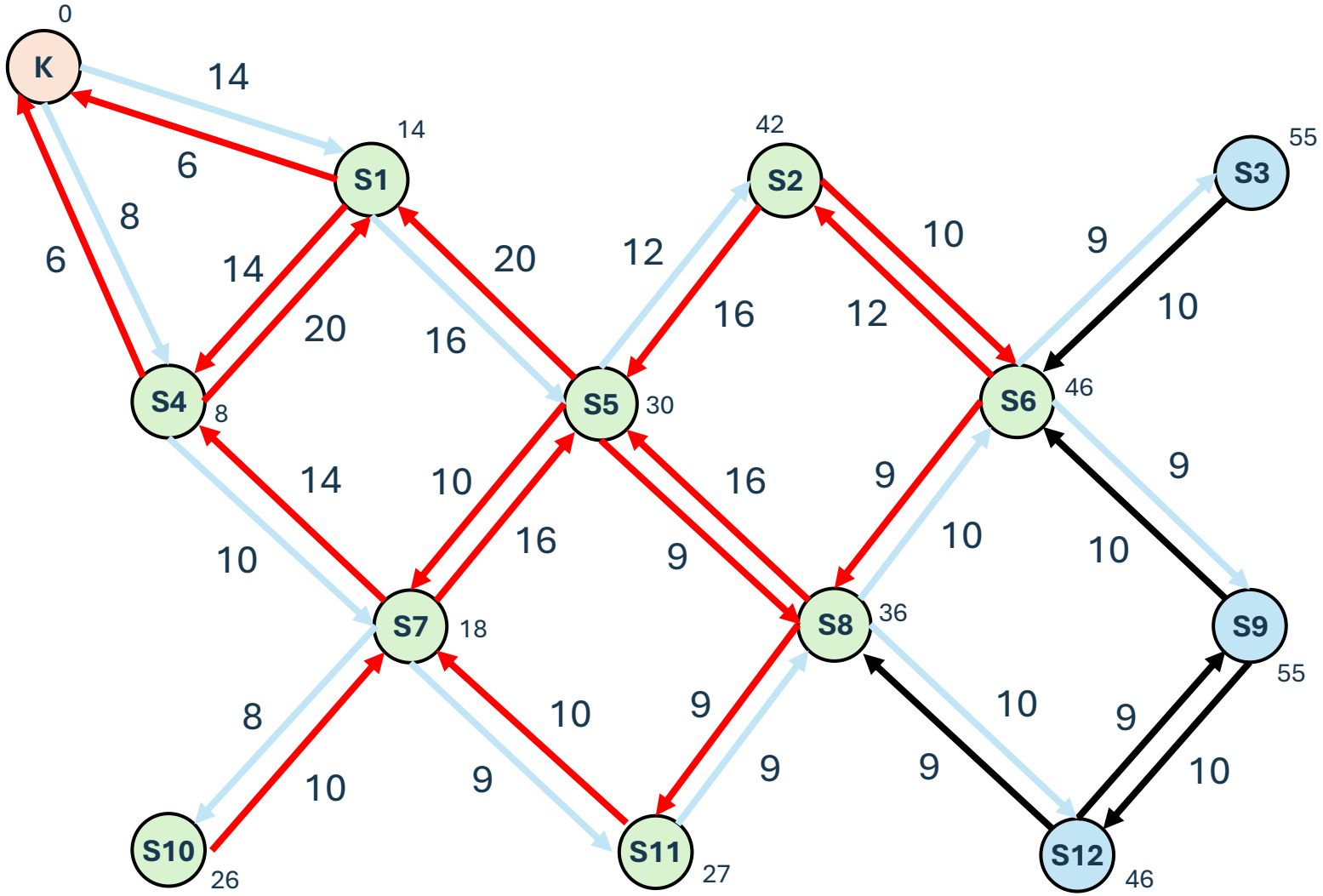
S1: 40%, S2: 80%, S3: 95%,
S4: 70%, S5: 60%, S6: 90%,
S7: 90%, S8: 95%, S9: 95%,
S10: 100%, S11: 95%,
S12: 90%, K: 100%



Dijkstra-algoritmus

$$\omega_{ij} = TX_i + RX_j + \lambda(100 - b_j)$$

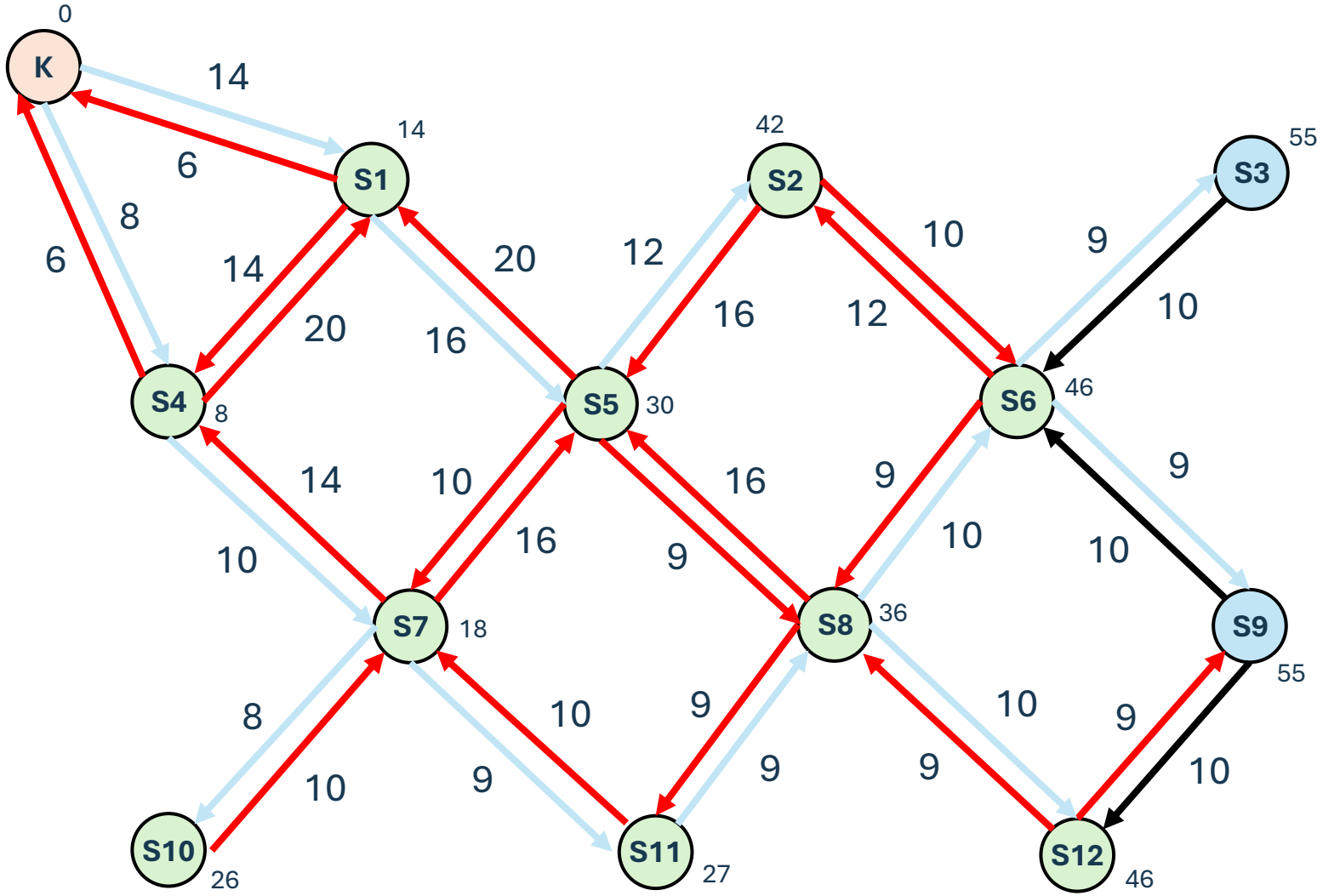
S1: 40%, S2: 80%, S3: 95%,
S4: 70%, S5: 60%, S6: 90%,
S7: 90%, S8: 95%, S9: 95%,
S10: 100%, S11: 95%,
S12: 90%, K: 100%



Dijkstra-algoritmus

$$\omega_{ij} = TX_i + RX_j + \lambda(100 - b_j)$$

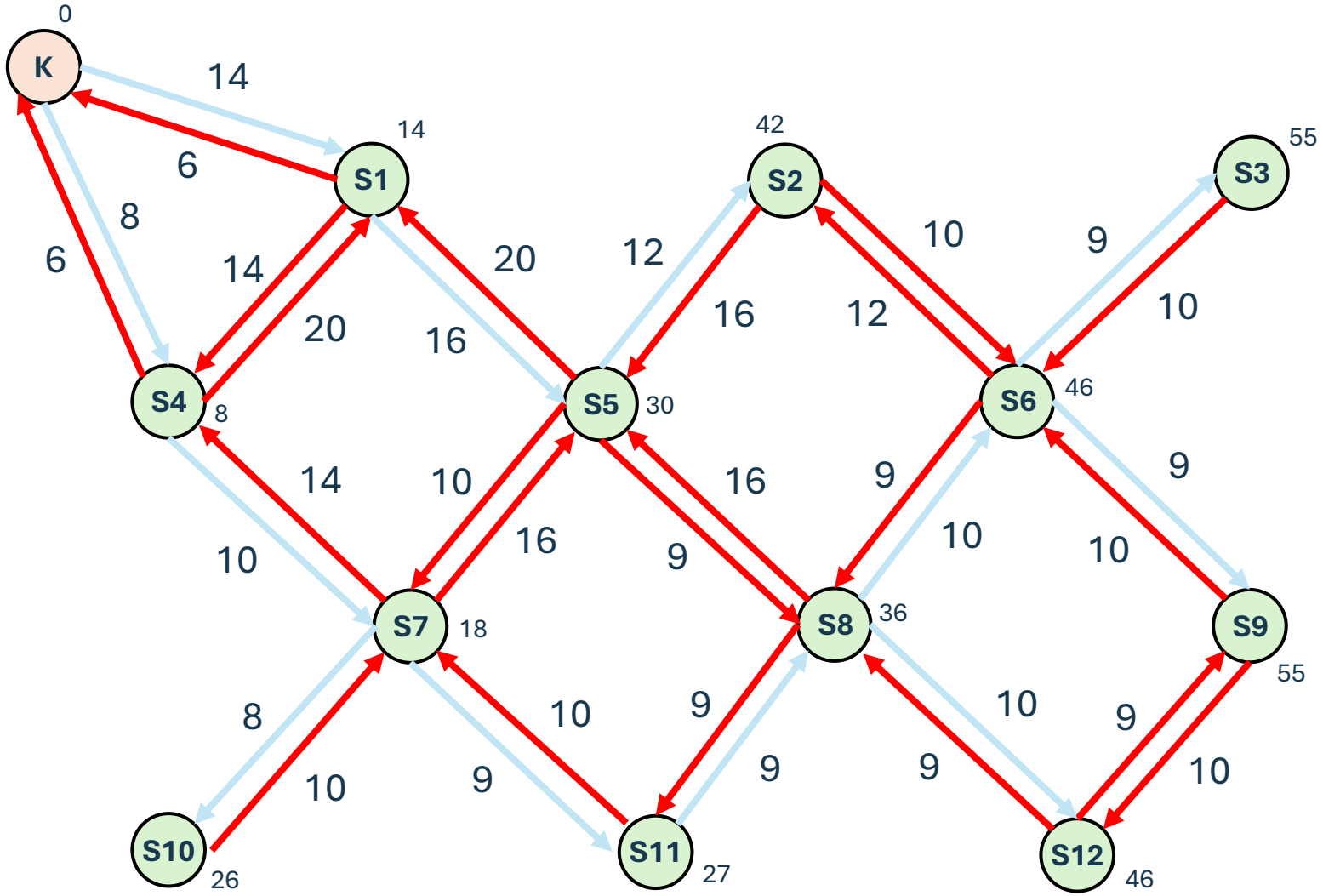
S1: 40%, S2: 80%, S3: 95%,
S4: 70%, S5: 60%, S6: 90%,
S7: 90%, S8: 95%, S9: 95%,
S10: 100%, S11: 95%,
S12: 90%, K: 100%



Dijkstra-algoritmus

$$\omega_{ij} = TX_i + RX_j + \lambda(100 - b_j)$$

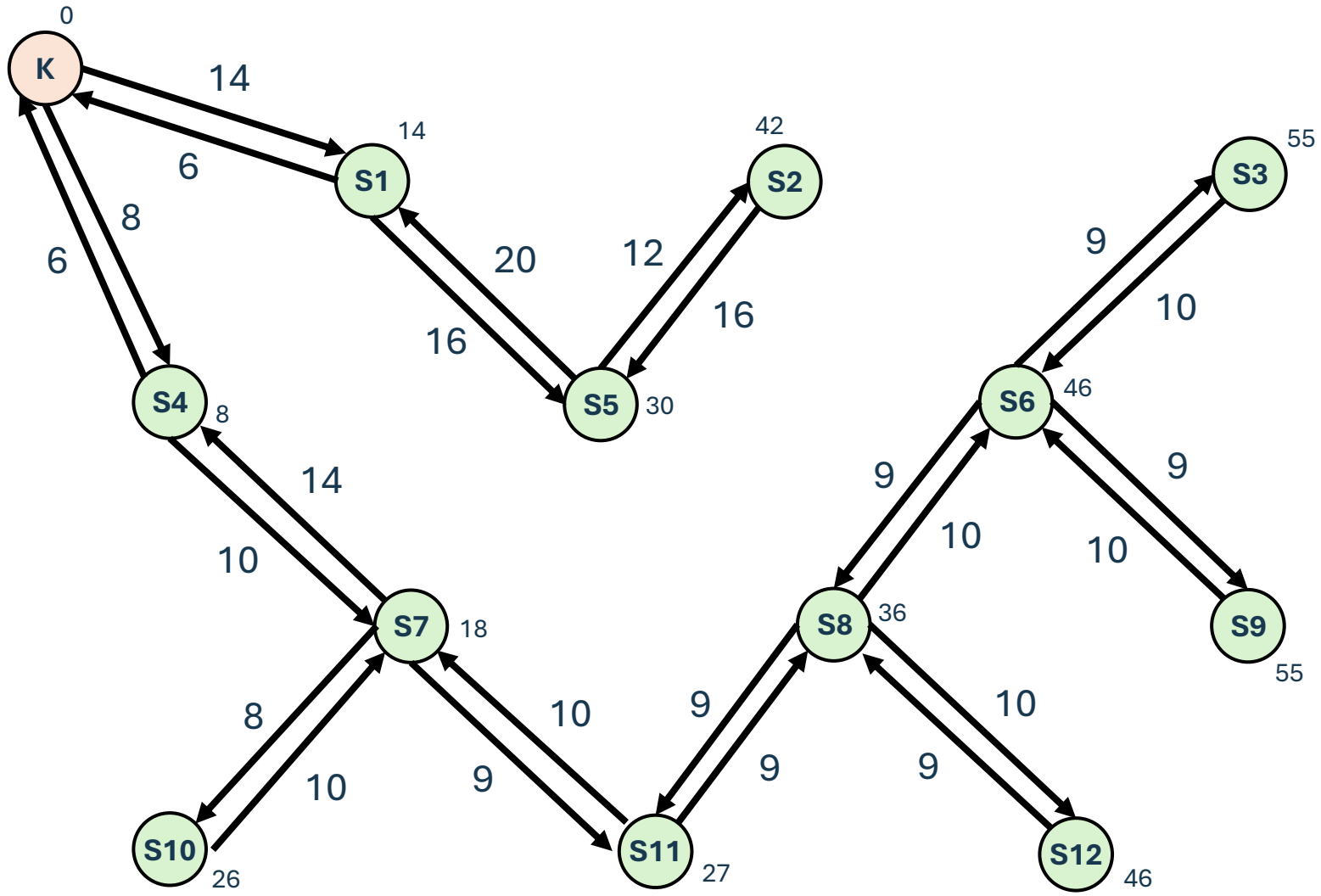
S1: 40%, S2: 80%, S3: 95%,
S4: 70%, S5: 60%, S6: 90%,
S7: 90%, S8: 95%, S9: 95%,
S10: 100%, S11: 95%,
S12: 90%, K: 100%



Dijkstra-algoritmus

$$\omega_{ij} = TX_i + RX_j + \lambda(100 - b_j)$$

S1: 40%, S2: 80%, S3: 95%,
S4: 70%, S5: 60%, S6: 90%,
S7: 90%, S8: 95%, S9: 95%,
S10: 100%, S11: 95%,
S12: 90%, K: 100%



Napenergia-termelés figyelembevétele

- Dőlésszög kedvezőtlenebb – domborzat
- Növényzet árnyékolja
- Rendszer hatásfoka – napelemek állapota
- Napszak változás

Két azonos akkumulátortöltöttségű szenzor nem feltétlenül azonos mértékben terhelhető, ha az egyik gyorsabban képes visszatermelni az elhasznált energiát.

Megoldás: minden szenzorhoz hozzárendelünk egy várható energiatermelési értéket, és nem jelenlegi akkumulátor szinttel számolunk, hanem például az 1 óra múlva várható szinttel

Útvonal frissítések

Mikor frissítsük az útvonalakat?

1. Teljes újra tervezés: új szenzor hozzáadása, egy hibás szenzor kijavítása
2. Időszakos optimalizálás: például 30 percenként
3. Kritikus helyzet esetén: például ha valamelyik szenzor 20% alá esik

A 2-3. esetben nem feltétlen jó egyből új útvonalakat kijelölni, például ha kicsi az eltérés, akkor csak energiát pazarlunk az átállás során, mondjuk csak akkor váltsunk ha legalább 10%-kal kedvezőbb az új útvonal.

További optimalizálási lehetőségek

- A Dijkstra az egyes útvonalak költségét optimalizálja, de nem veszi automatikusan figyelembe, hogy az összes kiválasztott útvonal együttesen milyen terhelést okoz.
- A javasolt megoldás: terhelést figyelembe vevő faépítés

Források

- https://www.agroinform.hu/gazdasag/minden-adat-szamit-iot-megoldasok-a-fenntarthato-mezogazdasagert-86063-032
- https://dfarm.hu
- https://hu.wikipedia.org/wiki/Szélességi_bejárás
- https://hu.wikipedia.org/wiki/Dijkstra-algoritmus

Köszönöm a figyelmet!