Elosztott rendszerek: Alapelvek és paradigmák
Distributed Systems: Principles and Paradigms

Maarten van Steen[1] Kitlei Róbert [2]

[1]VU Amsterdam, Dept. Computer Science
[2]ELTE Informatikai Kar

2. rész: Architektúrák

2015. május 24.

Tartalomjegyzék

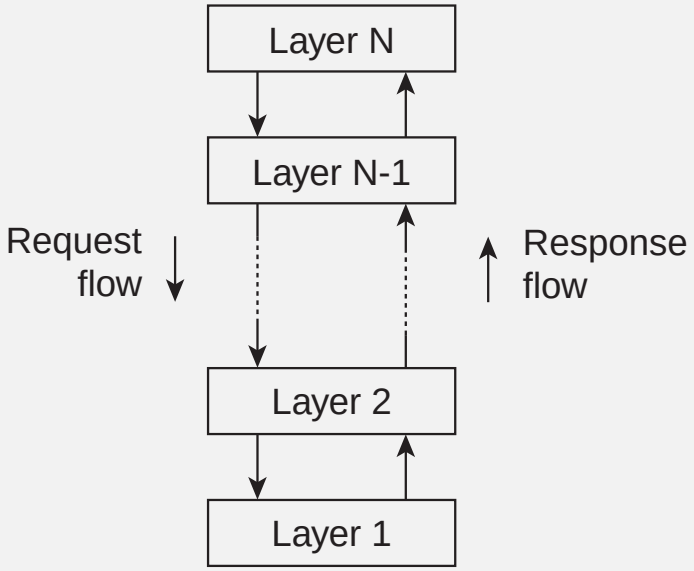
Architektúrák

- Architektúrafajták
- Szoftverarchitektúrák
- Architektúrák és köztesréteg
- Az elosztott rendszerek önszervezése

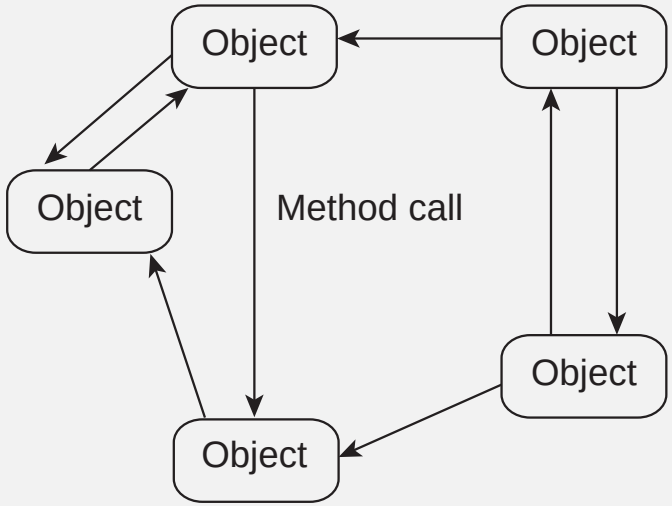
Architektúrafajták

Alapötlet

A rendszer elemeit szervezzük logikai szerepük szerint különböző komponensekbe, és ezeket osszuk el a rendszer gépein.



(a)



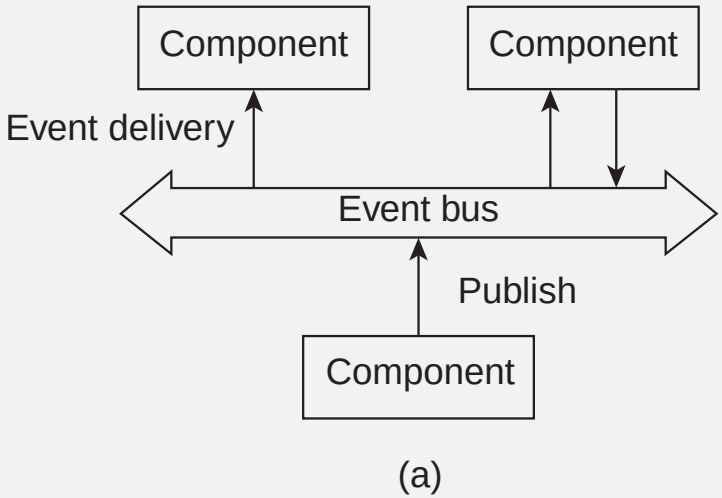
(b)

(a) A többrétegű megközelítés kliens-szerver rendszerek esetén jól működik
(b) Itt a komponensek (objektumok) összetettebb struktúrában kommunikálnak, mindegyik közvetlenül küld üzeneteket a többieknek.

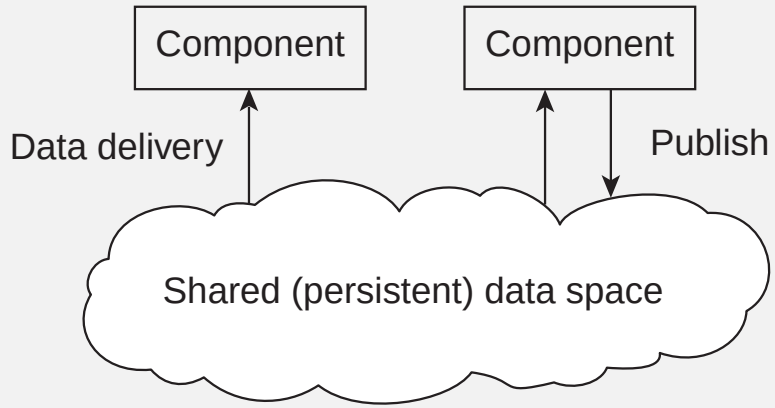
Architektúrafajták

További architektúrafajták

A komponensek közötti kommunikáció történhet közvetlen kapcsolat nélkül („anonim”), illetve egyidejűség nélkül („aszinkron”).



(a)



(b)

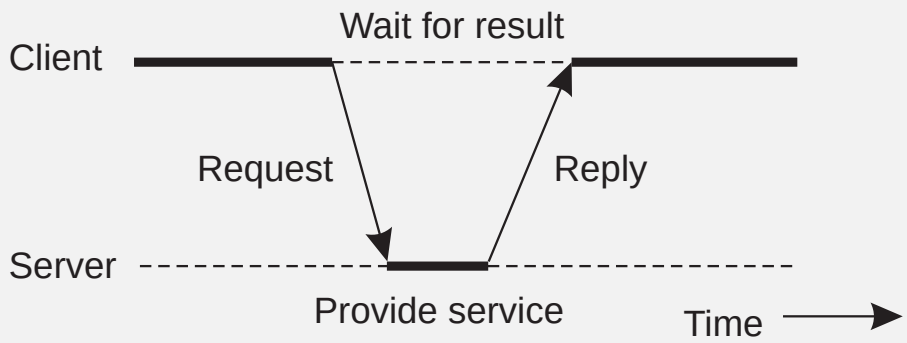
(a) Publish/subscribe modell (térben független)
(b) Megosztott, perzisztens adattár (térben és időben független)

Központosított architektúrák

Egyszerű kliens–szerver modell

Jellemzői:

- egyes folyamatok szolgáltatásokat ajánlanak ki (ezek a szerverek)
- más folyamatok ezeket a szolgáltatásokat szeretnék használni (ezek a kliensek)
- a kliensek és a szerverek különböző gépeken lehetnek
- a kliens kérést küld (amire a szerver válaszol), így veszi igénybe a szolgáltatást



Többrétegű architektúrák

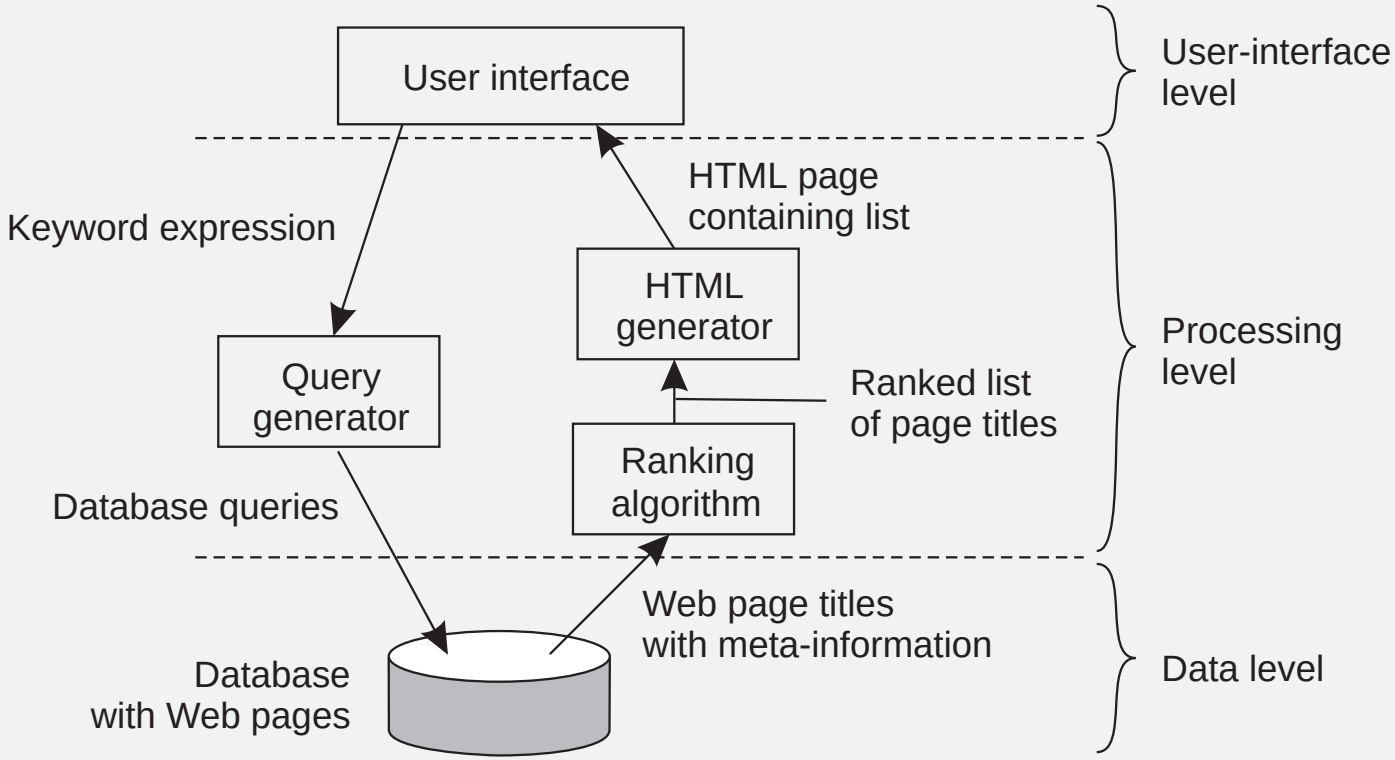
Elosztott információs rendszerek rétegelése

Az elosztott információs rendszerek gyakran három logikai rétegre („layer” vagy „tier”) vannak tagolva.

Háromrétegű architektúra

- Megjelenítés (user interface): az alkalmazás felhasználói felületét alkotó komponensekből áll
- Üzleti logika (application): az alkalmazás működését írja le (konkrét adatok nélkül)
- Perzisztencia (data layer): az adatok tartós tárolása

Többrétegű architektúrák

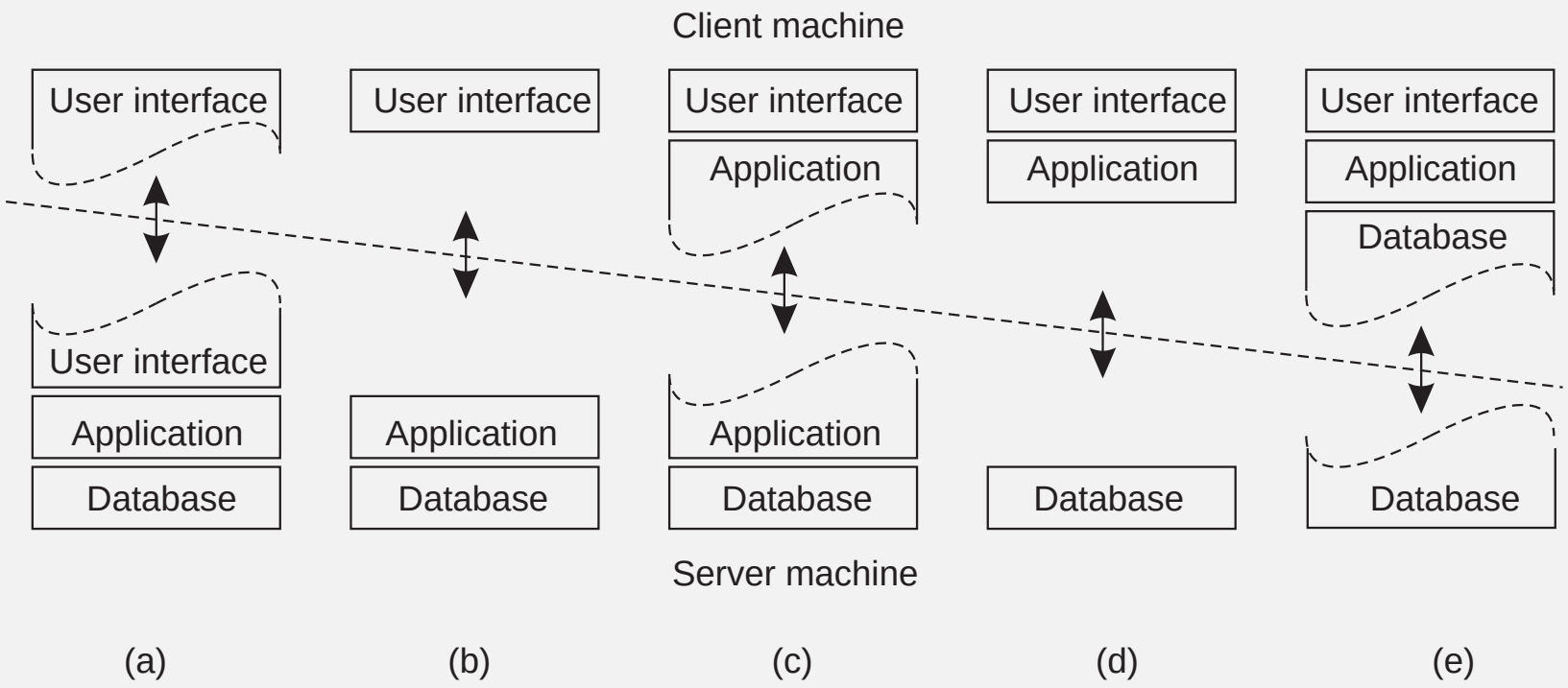


Többrétegű architektúrák

A három rétegből néha több is egy gépen található meg.

Kétrétegű architektúra: kliens/egyszerű szerver
Egyrétegű architektúra: nagygépre (mainframe) kötött terminál

A kétrétegű architektúra többféleképpen bonthatja fel a három réteget:



Decentralizált architektúrák

Peer-to-peer architektúra

Az utóbbi években a peer-to-peer (P2P) architektúra egyre népszerűbbé válik. A „peer” szó arra utal, hogy a csúcsok között (többnyire) nincsenek kitüntetett szerepűek.

- strukturált P2P: a csúcsok által kiadott gráfszerkezet rögzített
- strukturálatlan P2P: a csúcsok szomszédai véletlenszerűek
- hibrid P2P: néhány csúcsnak speciális szerepe van, ezek a többitől eltérő szervezésűek

Overlay hálózat

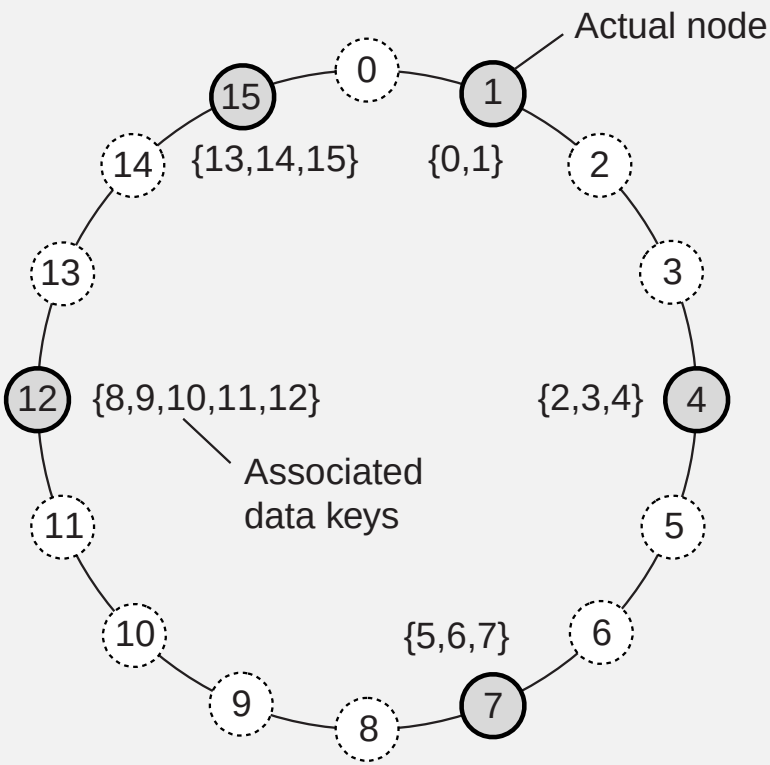
overlay: A gráfban szomszédos csúcsok a fizikai hálózaton lehetnek távol egymástól, a rendszer elfedi, hogy a köztük levő kommunikáció több gépet érintve történik.

- A legtöbb P2P rendszer overlay hálózatra épül.

Strukturált P2P rendszerek

Alapötlet

A csúcsokat valamilyen struktúra szerint overlay hálózatba szervezzük (pl. logikai gyűrű), és a csúcsoktól az azonosítójuk alapján lehet szolgáltatásokat igénybe venni.



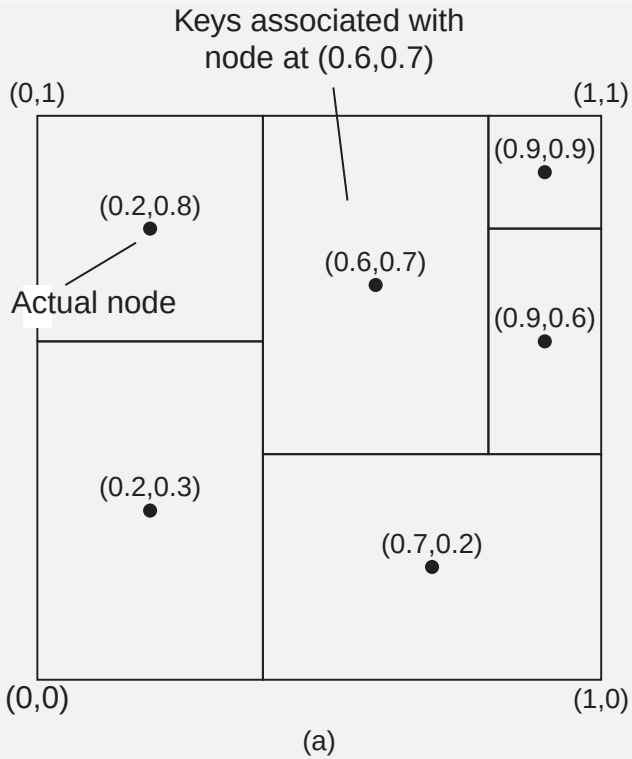
Példa: elosztott hasítótábla

Ebben a rendszerben kulcs-érték párokat tárolunk.
Az adott értéket tároló csúcsot hatékonyan meg lehet keresni a kulcsa alapján, akármelyik csúcsra is érkezik be a kérés.

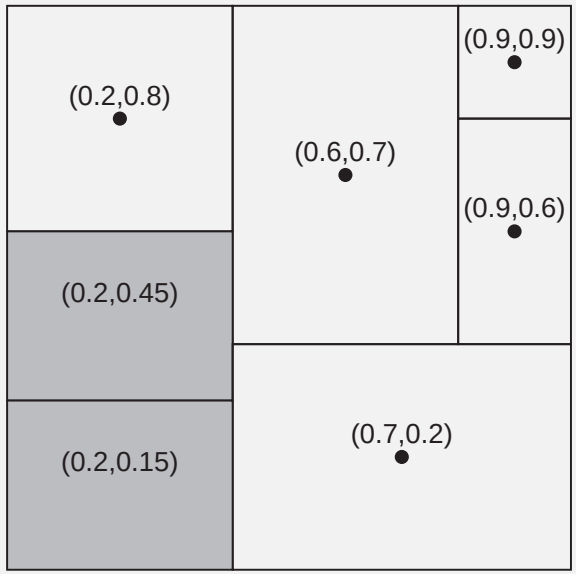
Strukturált P2P rendszerek

Példa: *d* dimenziós particionált tér

Az adatoknak most *d* mezője van, kulccsal nem rendelkeznek.
Az így adódó tér fel van osztva annyi tartományra, ahány csúcsunk van; minden csúcs valamelyik tartomány adataiért felelős.
Ha egy új csúcs érkezik, kettébontunk egy tartományt.



(a)



(b)

Strukturálatlan P2P rendszerek

Strukturálatlan P2P rendszer

A strukturálatlan P2P rendszerek igyekeznek véletlen gráfstruktúrát fenntartani.

- Mindegyik csúcsnak csak részleges nézete van a gráfról (a teljes hálózatnak csak egy kis részét látja).
- Minden P csúcs időközönként kiválaszt egy szomszédos Q csúcsot
- P és Q információt cserél, valamint átküldik egymásnak az általuk ismert csúcsokat

Megjegyzés

A rendszer hibatűrését és a gráf véletlenszerűségét nagyban befolyásolja az, hogy a harmadik lépésben pontosan milyen adatok kerülnek át.

Strukturálatlan P2P: pletykálás

Aktív szál

```
selectPeer(&B);
selectToSend(&bufs);
sendTo(B, bufs);

receiveFrom(B, &bufr);
selectToKeep(cache, bufr);
```

Passzív szál

```
receiveFromAny(&A, &bufr);
selectToSend(&bufs);
sendTo(A, bufs);
selectToKeep(cache, bufr);
```

selectPeer:	A részleges nézetből kiválaszt egy szomszédot.
selectToSend:	Az általa ismert szomszédok közül kiválaszt *n* darabot.
selectToKeep:	(1) A megkapott csúcsokat eltárolja lokálisan. (2) Eltávolítja a többszörösen szereplő csúcsokat. (3) A tárolt csúcsok számát *m* darabra csökkenti. Erre többfajta stratégia lehetséges.

Strukturálatlan P2P: pletykálás

Aktív szál

```
selectPeer(&B);
selectToSend(&bufs);
sendTo(B, bufs);

receiveFrom(B, &bufr);
selectToKeep(cache, bufr);
```

Passzív szál

```
receiveFromAny(&A, &bufr);
selectToSend(&bufs);
sendTo(A, bufs);
selectToKeep(cache, bufr);
```

selectPeer:	A részleges nézetből kiválaszt egy szomszédot.
selectToSend:	Az általa ismert szomszédok közül kiválaszt n darabot.
selectToKeep:	(1) A megkapott csúcsokat eltárolja lokálisan. (2) Eltávolítja a többszörösen szereplő csúcsokat. (3) A tárolt csúcsok számát m darabra csökkenti. Erre többfajta stratégia lehetséges.

Strukturálatlan P2P: pletykálás

Aktív szál

```
selectPeer(&B);
selectToSend(&bufs);
sendTo(B, bufs);

receiveFrom(B, &bufr);
selectToKeep(cache, bufr);
```

Passzív szál

```
receiveFromAny(&A, &bufr);
selectToSend(&bufs);
sendTo(A, bufs);
selectToKeep(cache, bufr);
```

selectPeer:	A részleges nézetből kiválaszt egy szomszédot.
selectToSend:	Az általa ismert szomszédok közül kiválaszt n darabot.
selectToKeep:	(1) A megkapott csúcsokat eltárolja lokálisan. (2) Eltávolítja a többszörösen szereplő csúcsokat. (3) A tárolt csúcsok számát m darabra csökkenti. Erre többfajta stratégia lehetséges.

Strukturálatlan P2P: pletykálás

Aktív szál

```
selectPeer(&B);
selectToSend(&bufs);
sendTo(B, bufs);

receiveFrom(B, &bufr);
selectToKeep(cache, bufr);
```

Passzív szál

```
receiveFromAny(&A, &bufr);
selectToSend(&bufs);
sendTo(A, bufs);
selectToKeep(cache, bufr);
```

selectPeer:	A részleges nézetből kiválaszt egy szomszédot.
selectToSend:	Az általa ismert szomszédok közül kiválaszt n darabot.
selectToKeep:	(1) A megkapott csúcsokat eltárolja lokálisan. (2) Eltávolítja a többszörösen szereplő csúcsokat. (3) A tárolt csúcsok számát m darabra csökkenti. Erre többfajta stratégia lehetséges.

Strukturálatlan P2P: pletykálás

Aktív szál

```
selectPeer(&B);
selectToSend(&bufs);
sendTo(B, bufs);

receiveFrom(B, &bufr);
selectToKeep(cache, bufr);
```

Passzív szál

```
receiveFromAny(&A, &bufr);
selectToSend(&bufs);
sendTo(A, bufs);
selectToKeep(cache, bufr);
```

selectPeer:	A részleges nézetből kiválaszt egy szomszédot.
selectToSend:	Az általa ismert szomszédok közül kiválaszt n darabot.
selectToKeep:	(1) A megkapott csúcsokat eltárolja lokálisan. (2) Eltávolítja a többszörösen szereplő csúcsokat. (3) A tárolt csúcsok számát m darabra csökkenti. Erre többfajta stratégia lehetséges.

Strukturálatlan P2P: pletykálás

Aktív szál

```
selectPeer(&B);
selectToSend(&bufs);
sendTo(B, bufs);

receiveFrom(B, &bufr);
selectToKeep(cache, bufr);
```

Passzív szál

```
receiveFromAny(&A, &bufr);
selectToSend(&bufs);
sendTo(A, bufs);
selectToKeep(cache, bufr);
```

selectPeer:	A részleges nézetből kiválaszt egy szomszédot.
selectToSend:	Az általa ismert szomszédok közül kiválaszt n darabot.
selectToKeep:	(1) A megkapott csúcsokat eltárolja lokálisan. (2) Eltávolítja a többszörösen szereplő csúcsokat. (3) A tárolt csúcsok számát m darabra csökkenti. Erre többfajta stratégia lehetséges.

Strukturálatlan P2P: pletykálás

Aktív szál

```
selectPeer(&B);
selectToSend(&bufs);
sendTo(B, bufs);

receiveFrom(B, &bufr);
selectToKeep(cache, bufr);
```

Passzív szál

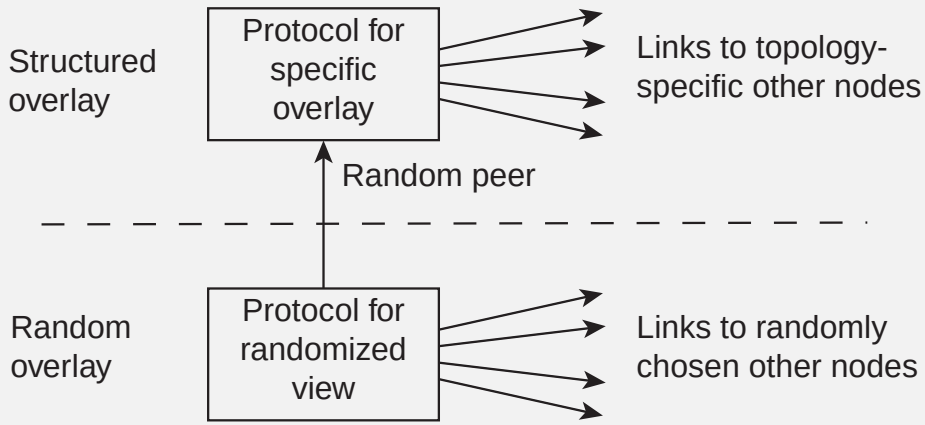
```
receiveFromAny(&A, &bufr);
selectToSend(&bufs);
sendTo(A, bufs);
selectToKeep(cache, bufr);
```

selectPeer:	A részleges nézetből kiválaszt egy szomszédot.
selectToSend:	Az általa ismert szomszédok közül kiválaszt n darabot.
selectToKeep:	(1) A megkapott csúcsokat eltárolja lokálisan. (2) Eltávolítja a többszörösen szereplő csúcsokat. (3) A tárolt csúcsok számát m darabra csökkenti. Erre többfajta stratégia lehetséges.

Overlay hálózatok topológiájának kezelése

Alapötlet

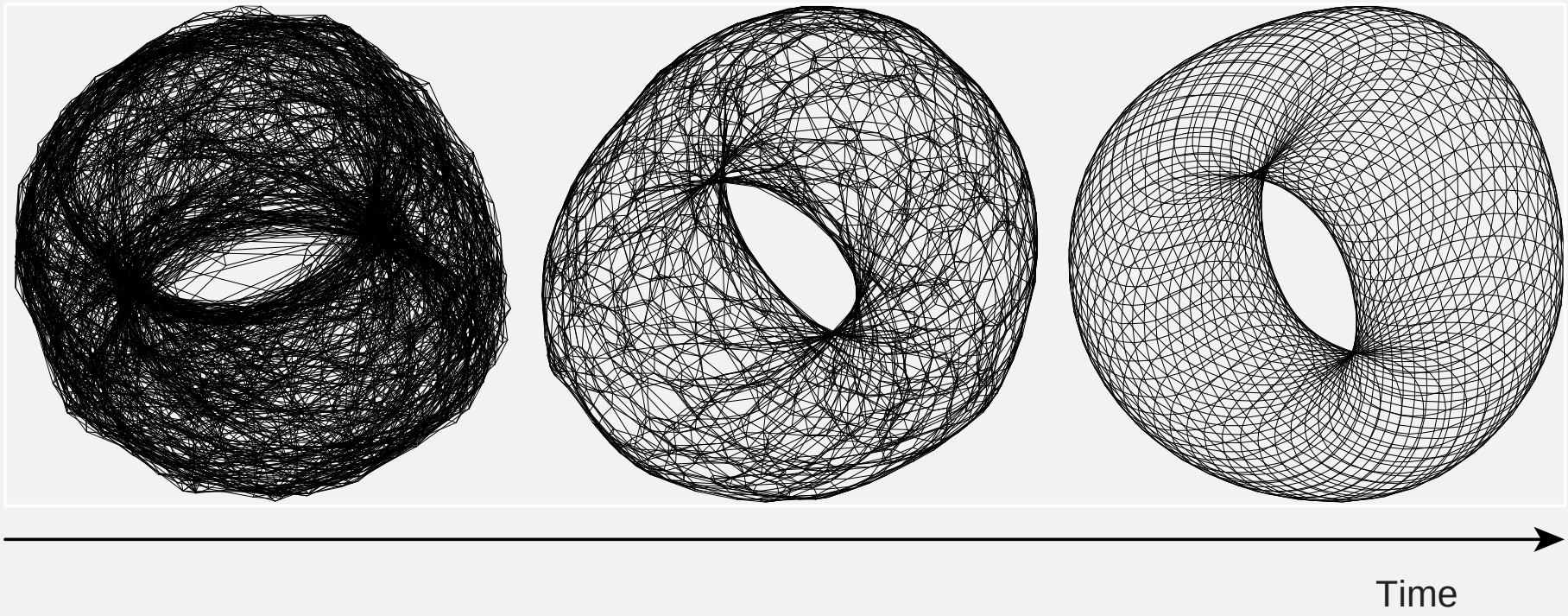
Különböztessünk meg két réteget:
(1) az alsó rétegben a csúcsoknak csak részleges nézete van;
(2) a felső rétegbe csak kevés csúcs kerülhet.
Az alsó réteg véletlenszerű csúcsokat ad át a felső rétegnek; a felső réteg ezek közül csak keveset tart meg.



Overlay topológia: példa: tórusz

Tórusz overlay topológia kialakítása

Ha megfelelően választjuk meg, milyen csúcsokat tartson meg a felső réteg, akkor a kezdetben véletlenszerű overlay kapcsolatok hamarosan szabályos alakba rendeződnek.
Itt egy távolságfüggvény szerinti megtartó szabály hat (az overlay a közelieket veszi át), és már az első néhány lépés után jól látszik a kijövő tórusz-alakzat.

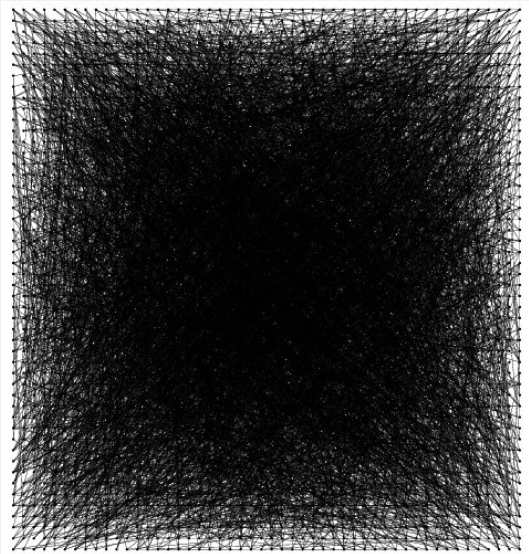


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.

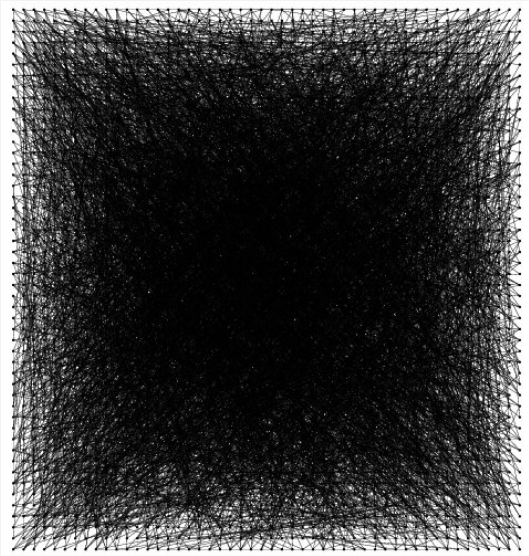


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.

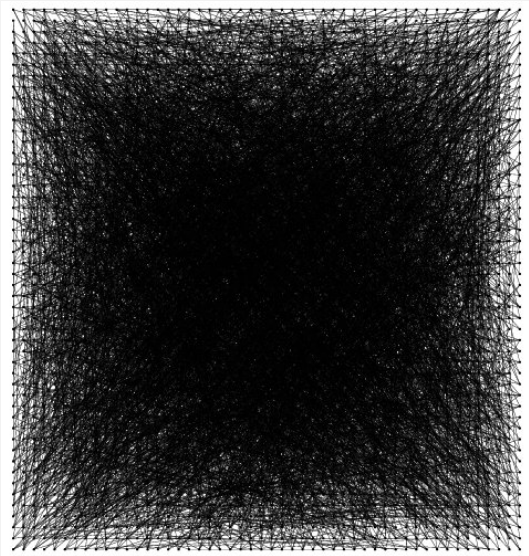


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.

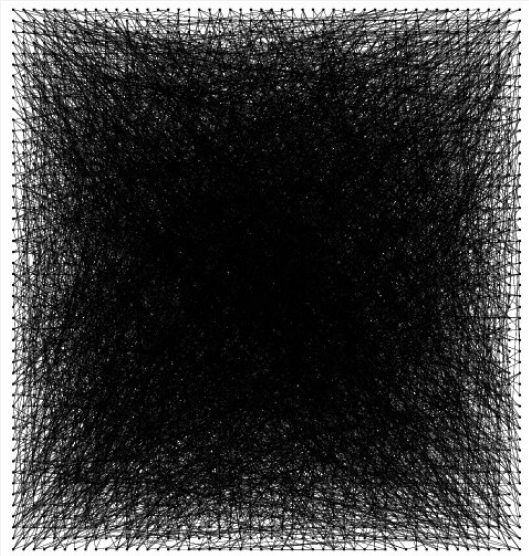


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.

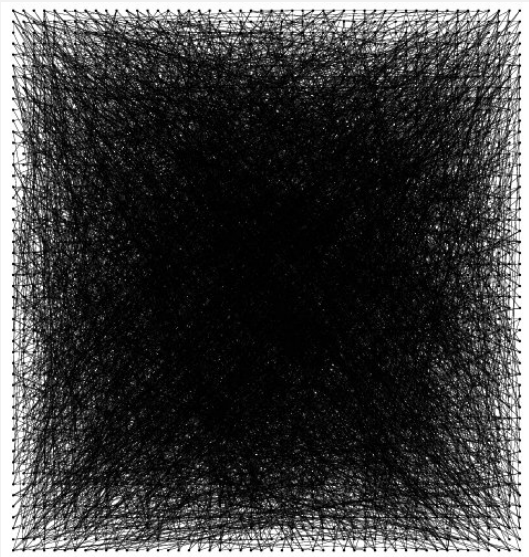


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.

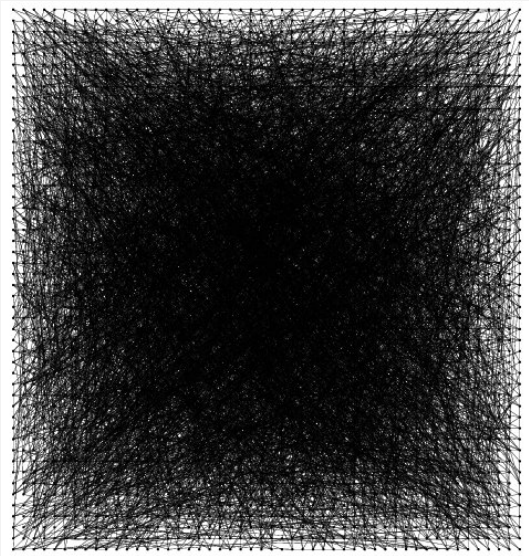


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.

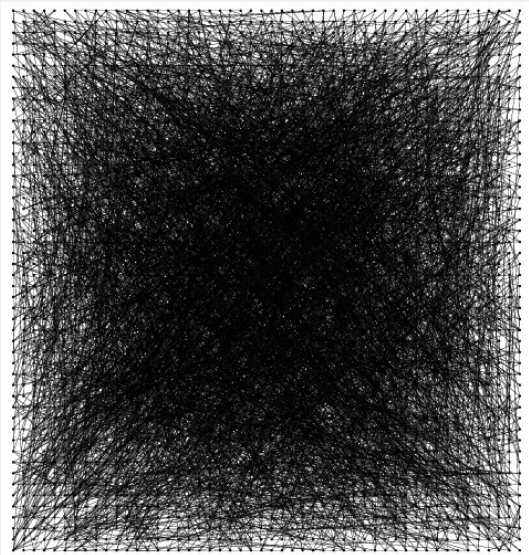


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.

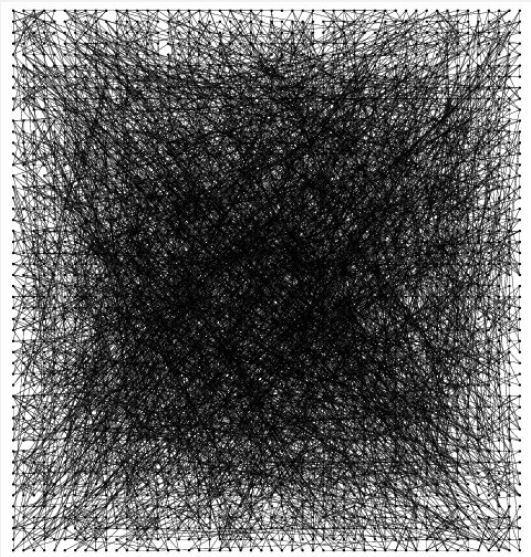


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.

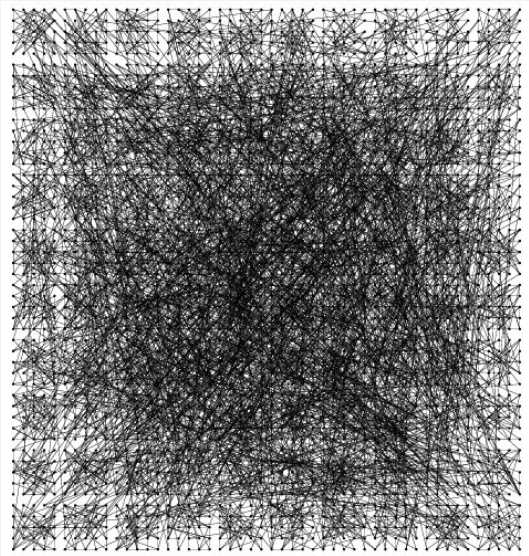


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.

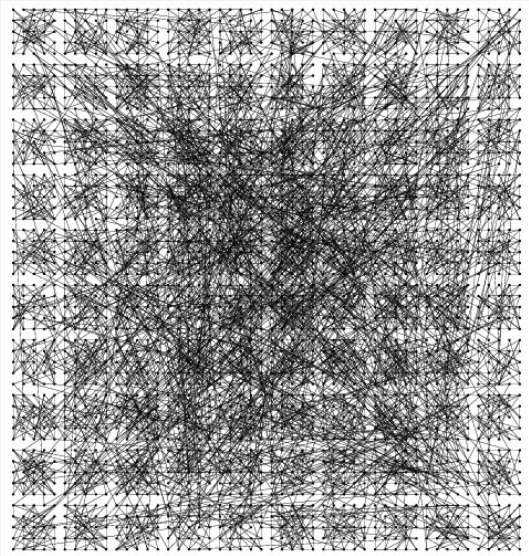


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.

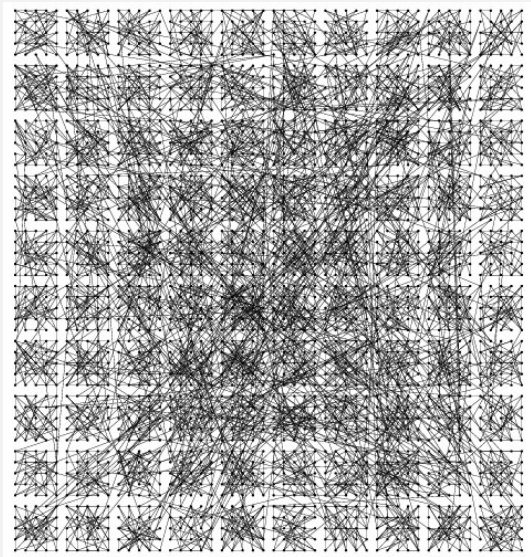


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.

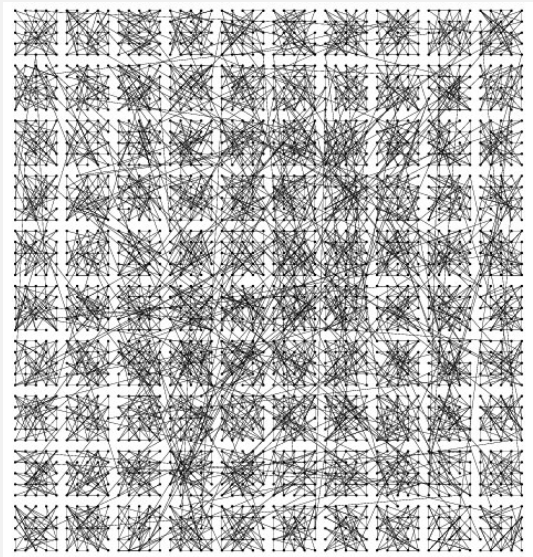


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.

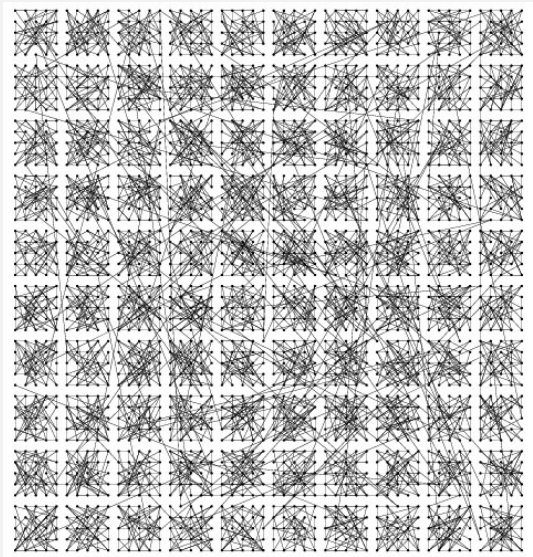


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.

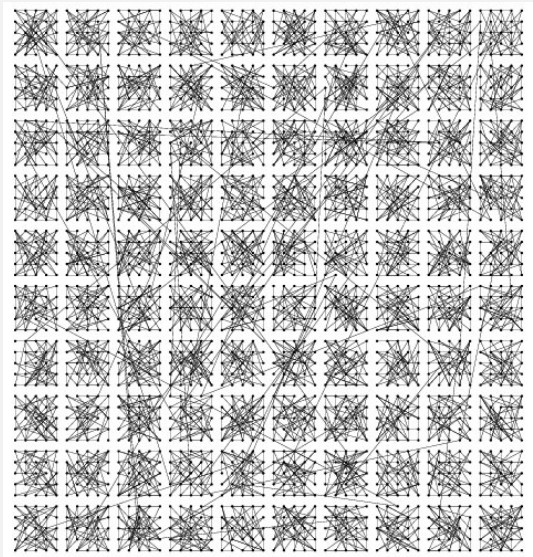


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.

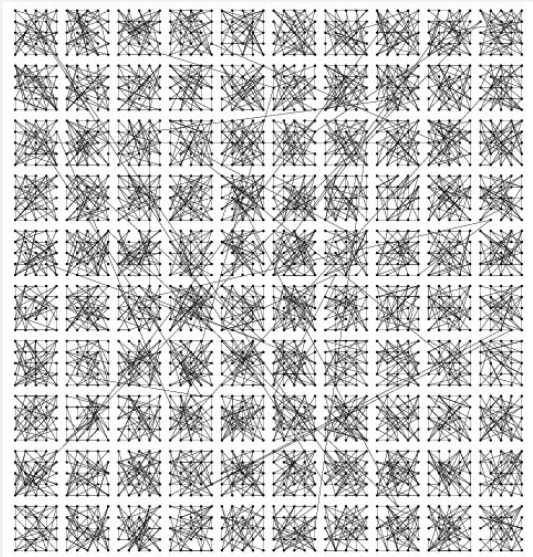


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.

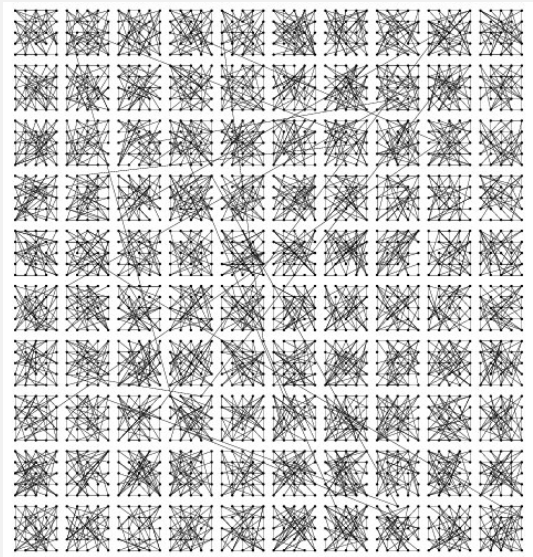


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.

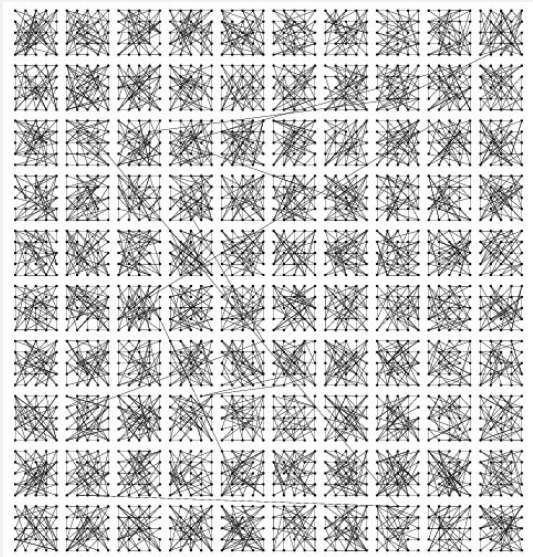


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.

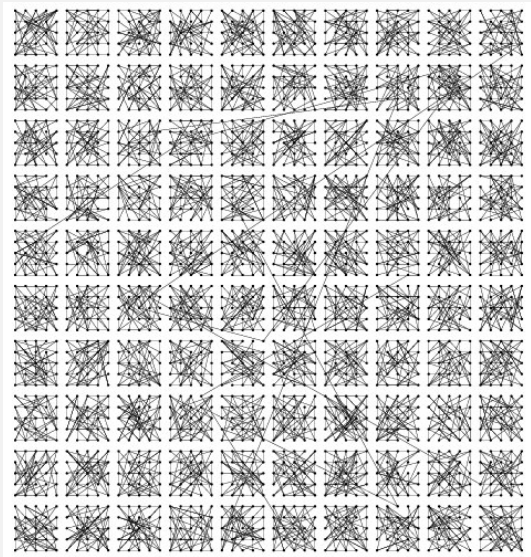


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.

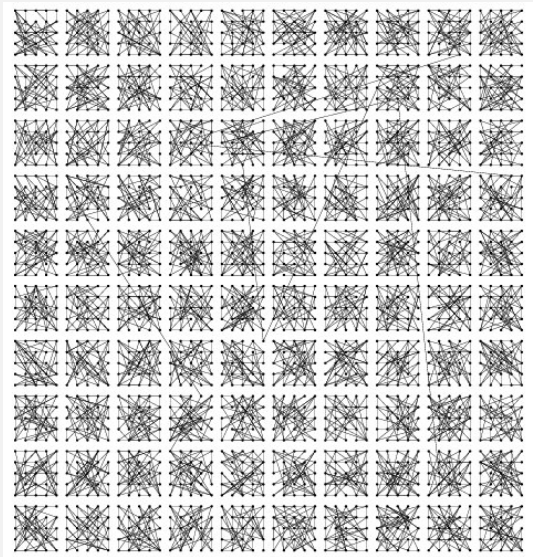


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.

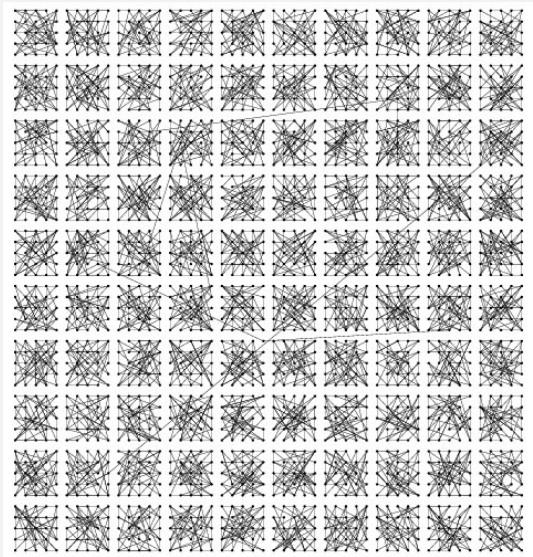


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.

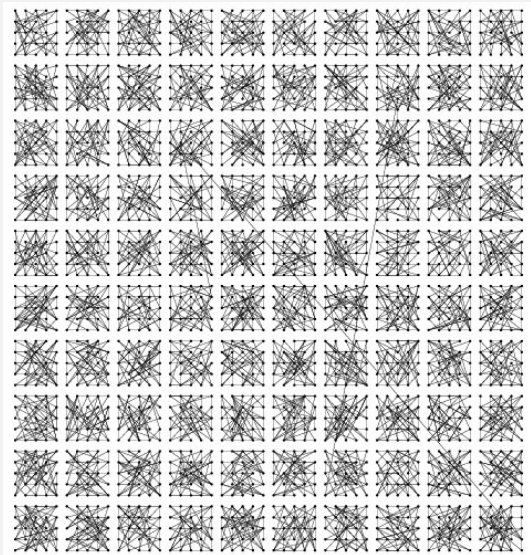


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.

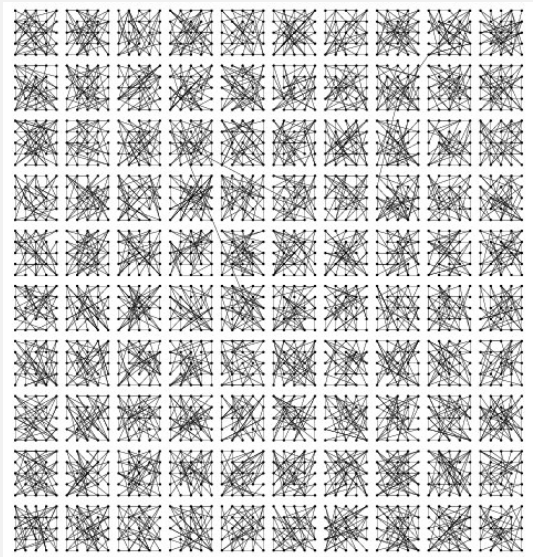


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.

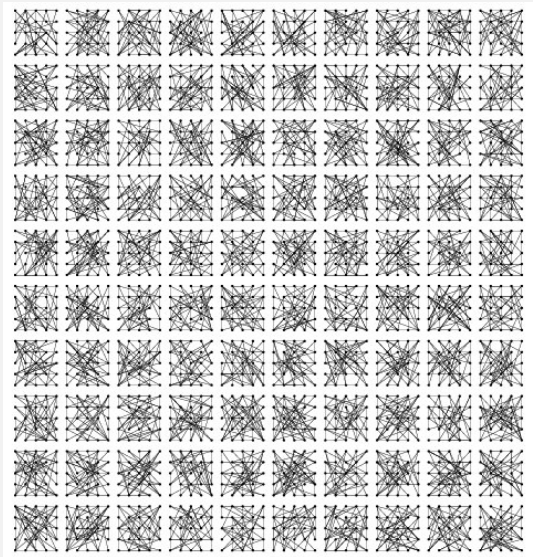


Overlay topológia: példa: clusterezés

Most mindegyik i csúcshoz hozzárendelünk egy $GID(i) \in \mathbb{N}$ számot, és azt mondjuk, hogy i a $GID(i)$ csoportba tartozik. Szintén távolságfüggvényt használunk:

$$dist(i,j) = \begin{cases} 1 & \text{ha } GID(i) = GID(j) \\ 0 & \text{ha } GID(i) \neq GID(j) \end{cases}$$

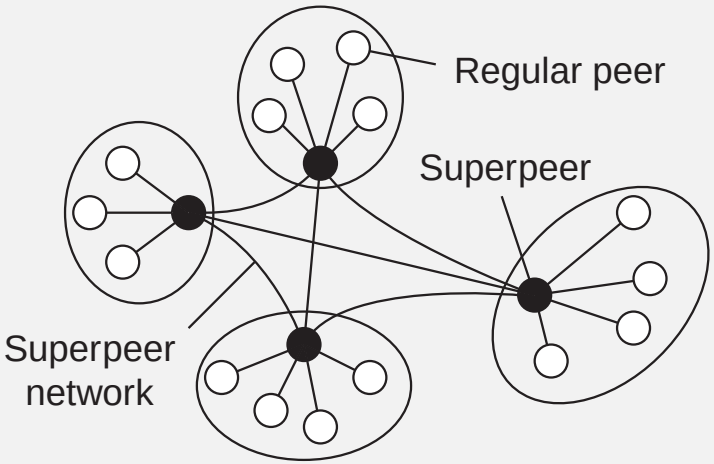
Itt is igen gyorsan kialakul a kívánt szerkezet: csak az azonos csoportbeli csúcsok között lesz kapcsolat, kialakulnak a clusterek.



Superpeer csúcsok

Superpeer

superpeer: olyan kisszámú csúcs, amelyeknek külön feladata van



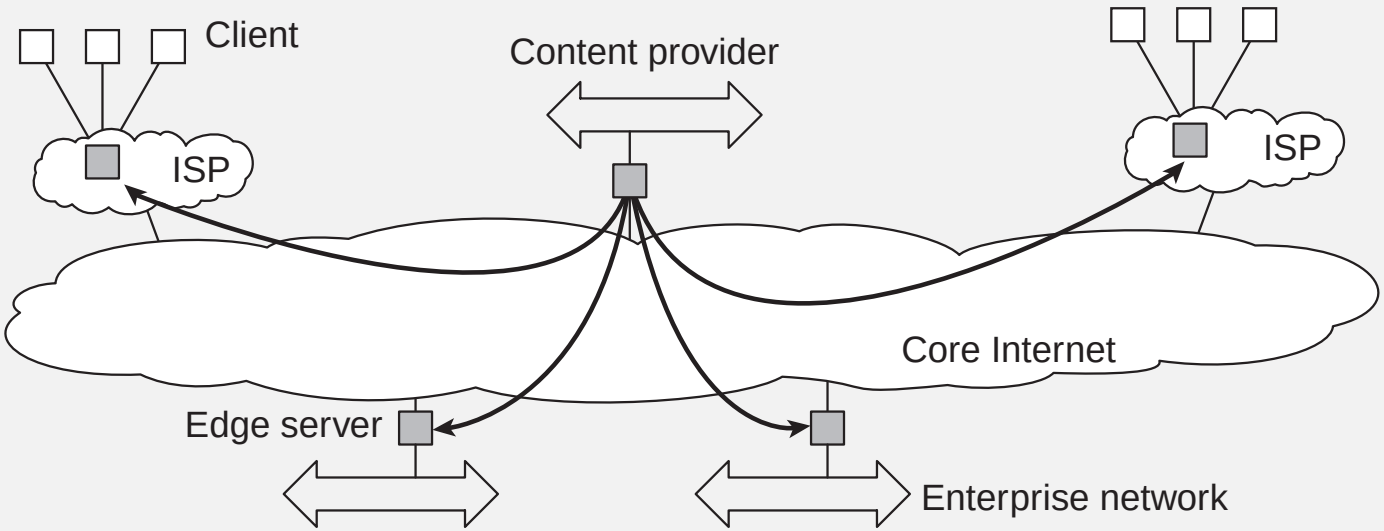
Néhány jellemző feladat

- kereséshez index fenntartása
- a hálózat állapotának felügyelete
- csúcsok közötti kapcsolatok létrehozása

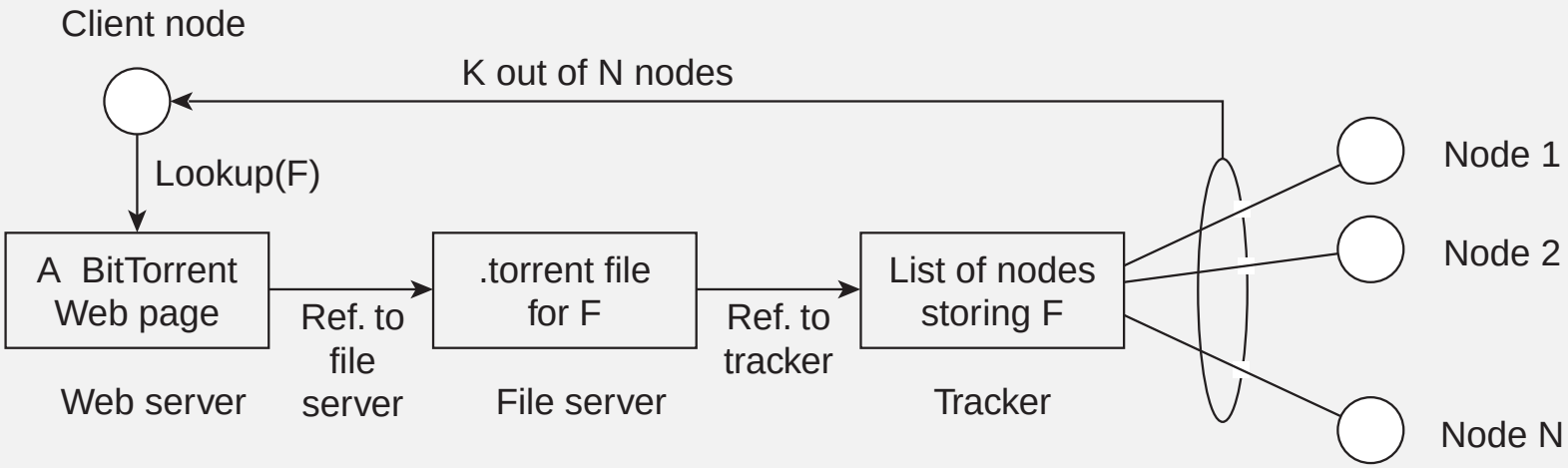
Hibrid arch.: kliens-szerver + P2P: edge szerver

Példa

edge szerver: az adatokat tároló szerver, a kliensekhez minél közelebb van elhelyezve, jellemzően ott, ahol egy nagyobb hálózat az Internetre csatlakozik
Content Delivery Network (CDN) rendszerekben jellemző, a tartalomszolgáltatás hatékonyságát növelik és költségét csökkentik.



Hibrid arch.: kliens-szerver + P2P: BitTorrent



Alapötlet

Miután a csúcs kiderítette, melyik másik csúcsok tartalmaznak részeket a kívánt fájlból, azokat párhuzamosan tölti le, és egyúttal önmaga is kiajánlja megosztásra.

Architektúrák és köztesréteg

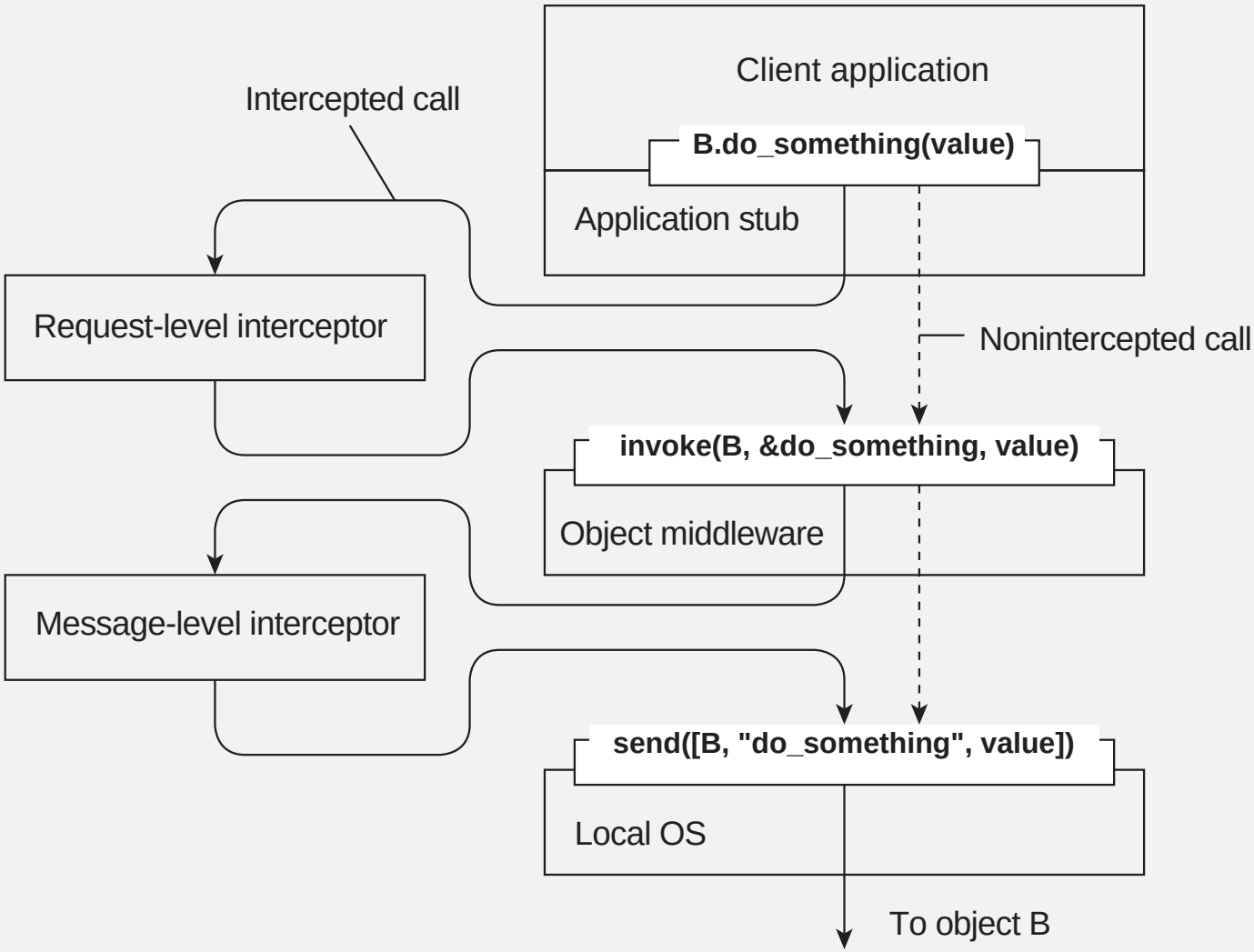
Probléma

Előfordulhat, hogy az elosztott rendszer/alkalmazás szerkezete nem felel meg a megváltozott igényeknek.
Ilyenkor legtöbbször nem kell újraírni a teljes rendszert: elegendő lehet (dinamikusan) adaptálni a köztesréteg viselkedését.

Interceptor

interceptor: Távoli objektum elérése során a vezérlés szokásos menetébe avatkozik bele, pl. átalakíthatja más formátumra a kérést.
Jellemzően az architektúra rétegei közé illeszthető.

Interceptors



Adaptív middleware

Funkciók szétválasztása (separation of concerns): A szoftver különböző jellegű funkciói váljanak minél jobban külön, így azokat könnyebb egymástól függetlenül módosítani.

Önvizsgálat (reflection): A program legyen képes feltárni a saját szerkezetét, és futás közben módosítani azt.

Komponensalapú szervezés: Az elosztott alkalmazás legyen moduláris, a komponensei legyenek könnyen cserélhetőek. A komponensek közötti függések legyenek egyértelműek, és csak annyi legyen belőlük, amennyi feltétlenül szükséges.

Önszervező elosztott rendszerek

Adaptív rendszer képességei

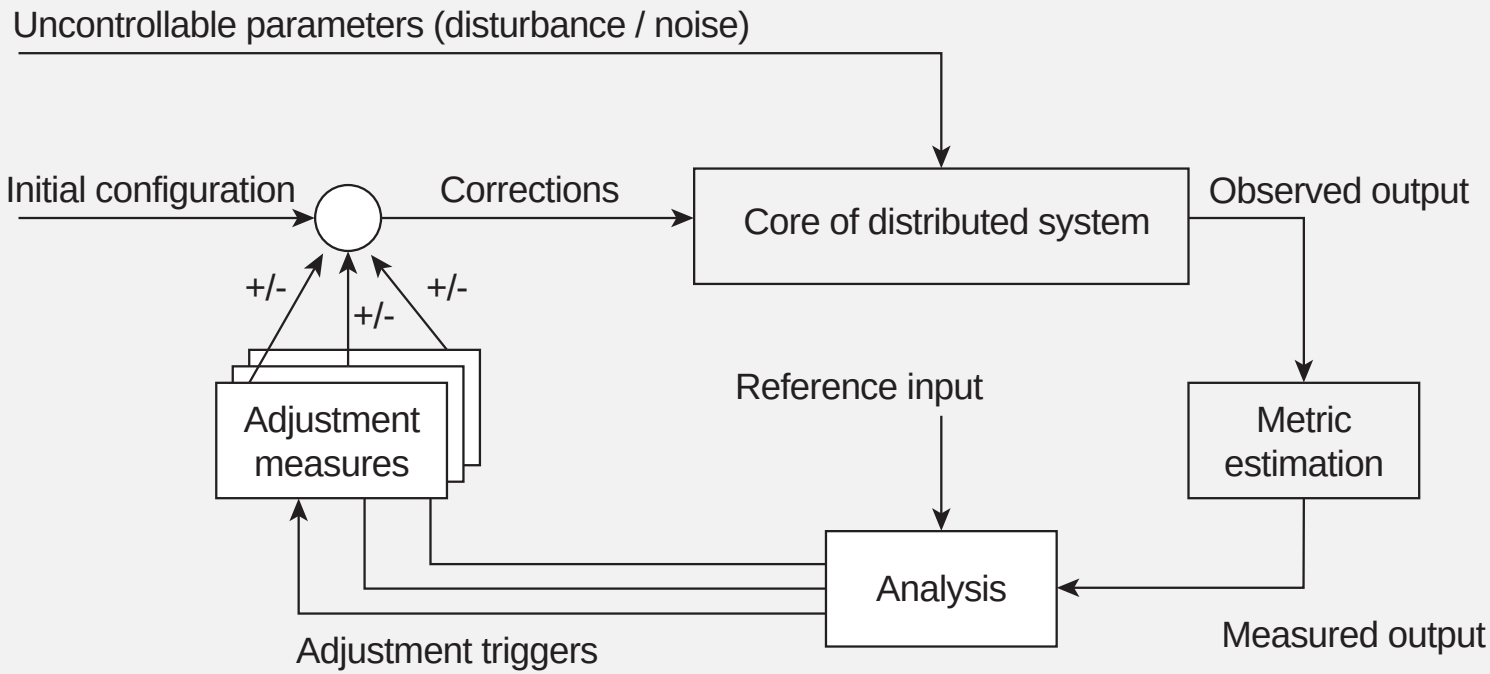
Az egyes szoftverelemek adaptivitása kihat a rendszerre, ezért megvizsgáljuk, hogyan lehet adaptív rendszereket készíteni. Különféle elvárásaink lehetnek:

- Önkonfiguráció
- Önkezelő
- Öngyógyító
- Önoptimalizáló
- Ön*

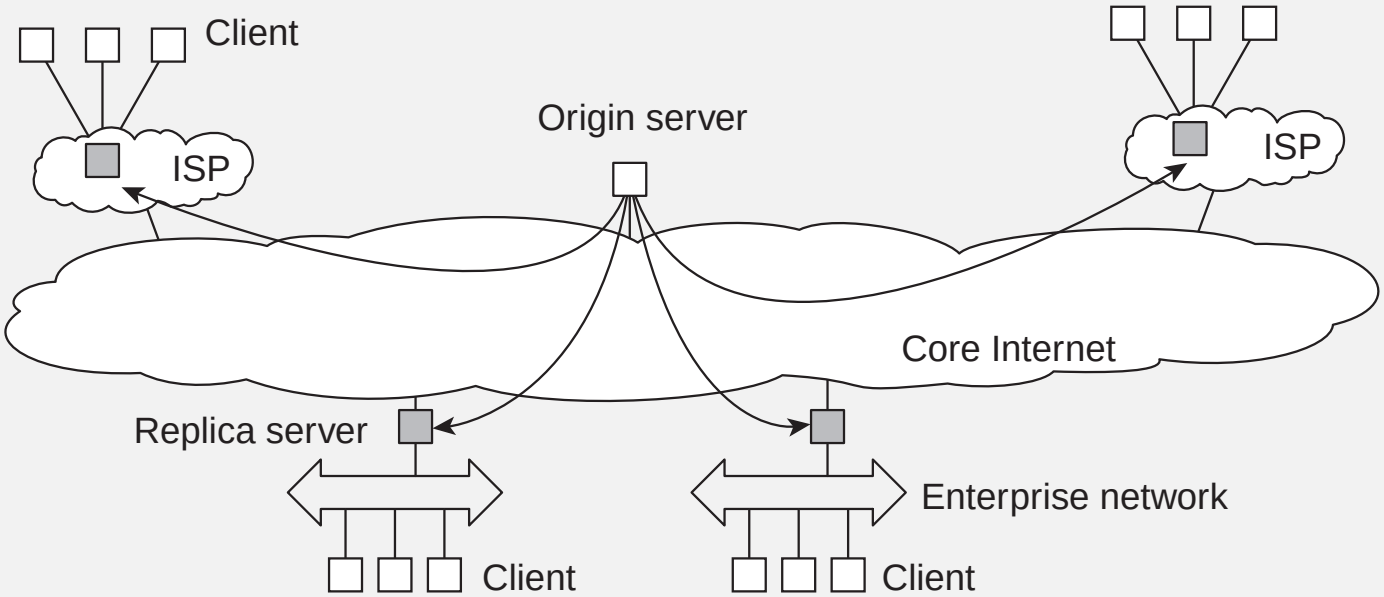
Adaptivitás visszacsatolással

Visszacsatolásos modell

Az ön* rendszerek sokszor az alábbi jellegű visszacsatolásos vezérléssel rendelkeznek: mérik, hogy a rendszer mennyire tér el a kívánt tulajdonságoktól, és szükség szerint változtatnak a beállításokon.



Példa: Globule



- Kollaboratív webes CDN, a tartalmakat költségmodell alapján helyezi el (minden szempontra: fontosság$\times$költség).
- A központi szerver (origin server) elemzi, ami történt, és az alapján állítja be a fontossági paramétereket, hogy mi történt volna, ha P oldalt az S edge szerver tárolta volna.
- A számításokat különböző stratégiákra végzi el, végül a legjobbat választja ki.