10. Fluid forgalmak, forgalom-szabályozás

Kommunikációs hálózatok teljesítményének elemzése

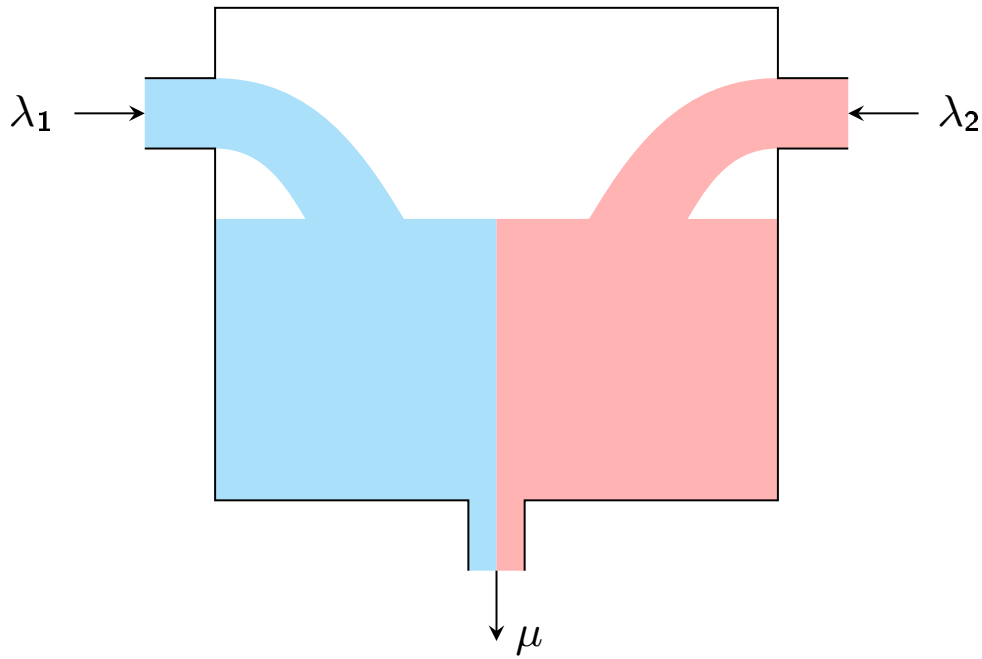
Horváth Illés

2024/11/13

1. Forgalmak priorizálása
2. Statikus fluid hálózat elemzése
3. Bufferek
4. Token bucket
5. Adaptív és elasztikus forgalmak
6. Random early drop (RED)

Forgalomszabályozás

Mi történik, ha egy szerverhez több forrásból is érkezik forgalom?



Ha $\sum_i \lambda_i > \mu$, akkor forgalom priorizálásra van szükség (congestion control).

Forgalom priorizálás

Több forrásból érkező forgalmak priorizálásra két fő típusú elvet szoktak használni:

- Szigorú priorizálás. Ilyenkor az egyik forgalom teljes elsőbbséget élvez a másikkal szemben.
- WFQ (weighted fair queueing): Ilyenkor minden egyes forgalomnak van egy előre adott súlya, és a kiszolgálási sebességet a súlyok arányában kapják meg.

A gyakorlatban egy nagy hálózatban (több forgalom, több szerver) ezt úgy szokás megvalósítani, hogy minden forgalom néhány forgalmi osztály valamelyikébe tartozik, és az osztályokat prioritizáljuk, illetve azonos prioritású osztályokhoz WFQ súlyokat rendelünk (a WFQ súlyok akár szerverenként eltérhetnek).

Fluid hálózat elemzése

Egy olyan hálózatot, ahol több forgalom és több szerver is van, lehet prioritási szintenként elemezni (a legnagyobb prioritástól kezdve), mivel az alacsonyabb prioritású forgalmak nem hatnak a magasabb prioritású forgalmakra. Gyakorlatilag az alacsonyabb prioritású forgalmak szemszögéből nézve a magasabb prioritású forgalmak tekinthetők a hálózat részének.

Ha egy prioritási szinten belül több forgalmi osztály is van, akkor azokra feltesszük, hogy WFQ súlyok szerint történik a kiszolgáló kapacitás megosztása.

Statikus WFQ hálózat

Egy statikus WFQ hálózat a következő:

- Adott k szerver $\mu_1, \ldots, \mu_k$ kiszolgáló sebességekkel. A szervereknek nincs buffere.
- Adott n forgalom, $\lambda_1, \ldots, \lambda_n$ érkezési sebességekkel (amelyeket konstansnak tekintünk).
- Az A $n \times k$ méretű 0–1 mátrix adja meg, hogy mely forgalmak mely szervereken haladnak keresztül. (Egy forgalom egy szerveren maximum egyszer halad át.)
- A WFQ súlyokról most feltesszük, hogy minden szerverre egyformák; a WFQ súlyokat a w n-hosszú vektor tartalmazza: az i-edik igény súlya w_i.

Ütközés esetén a forgalmak visszaszabályozzák az érkezési sebességet (a hálózatba való belépéstől számítva az összes szerven, azonnal).

A feladat: λ, μ, A, w bemenetre számítsuk ki, hogy melyik forgalom mekkora kiszolgáló sebességet kap a hálózatban.

Statikus WFQ hálózat elemzése

Az algoritmus során monoton módon növelni fogjuk a $\lambda_{i,\text{curr}}$ hozzárendelt sebességeket, és közben minden forgalomról számon tartjuk, hogy kész ($a_i = 0$) vagy aktív ($a_i = 1$).

Kezdetben minden forgalom aktív. Az aktív forgalmakat a WFQ súlyok arányában növeljük amíg lehet. Amit nem lehet, azt késznek jelöljük és a továbbiakban nem növeljük.

Véges sok lépésben minden forgalmat késznek fogunk jelölni, és amikor az utolsó is kész, akkor a végső $\lambda_{i,\text{curr}}$ értékek a ténylegesen megvalósuló sebességek.

Statikus WFQ hálózat elemzése

1. Inicializálás: $\lambda_{i,\text{curr}} = 0,\ a_i = 1 \quad \forall i = 1, \ldots, n.$
2. h értékét korlátozza az egyes forgalmak saját sebessége, illetve a szerverek kiszolgálási sebessége:

$$\lambda_{i,\text{curr}} + a_i w_i h \leq \lambda_i \qquad \forall i = 1, \ldots, n,$$
$$\sum_i (\lambda_{i,\text{curr}} + a_i w_i h) A_{ij} \leq \mu_j \qquad \forall j = 1, \ldots, k.$$

   h értékét a lehető legnagyobbra állítjuk úgy, hogy az összes korlátot teljesítse.
3. Az aktív i forgalmak sebességét $\lambda_{i,\text{curr}} + w_i h$-ra módosítjuk. Ha a maximális h értéket egy forgalom saját sebessége adta, akkor azt késznek jelöljük; ha egy szerver sebessége adta, akkor az adott szerverben az összes aktív forgalmat késznek jelöljük.
4. Ha még van aktív forgalom, a 2. lépéssel folytatjuk, egyébként készen vagyunk.

Statikus WFQ hálózat

Az előző algoritmus eredménye leírja a hálózat pillanatnyi állapotában megvalósuló, WFQ szerint kiosztott sebességeket.

Ha a λ_i értékek megváltoznak, akkor újra kell számolni a kiosztást is. Ha a λ_i értékek gyakran változnak, akkor a sebességkiosztás (vagyis a hálózat viselkedése) is szélsőséges ingadozásokat mutathat.

Az algoritmus során kihasználtuk, hogy minden forgalom pillanatnyi sebességét egyetlen érték jellemzi a teljes hálózatban. Ha bufferek is vannak a hálózatban, akkor ez nem teljesül, és a fenti algoritmus nem használható.

Bufferek

Egy hálózatban a bufferek fő szerepe, hogy a forgalmi igény-ingadozásokat kisimítsák.

A bufferek mérete jellemzi, hogy milyen időtávon simítanak ki ingadozásokat. Kis bufferek rövid, nagy bufferek hosszabb időtávú ingadozásokat simítanak ki.

Bufferek jelenlétében az instant visszaszabályozásnak nincs értelme, minden egyes forgalom érkezhet nagy rátával is (legfeljebb ideiglenesen eltárolódik egy bufferben); ilyenkor a releváns kérdés a stabilitás feltétele.

Ilyenkor csak a hosszú távú viselkedés számít: ha $\bar{\lambda}_i$ jelöli az i-edik forgalom hosszú távú időátlagát, akkor a stabilitás feltétele:

$$\sum_i \bar{\lambda}_i A_{ij} < \mu_j \qquad \forall j = 1, \ldots, k$$

a priorizálástól (pl. WFQ vagy szigorú prioritás vagy egyéb) függetlenül!

Token bucket

A következő forgalomszabályozási eszköz, amit megvizsgálunk, a token bucket (nevezik leaky bucketnek is, viszont leaky bucket alatt néha mást is értenek. . .).

A legegyszerűbb esetben egyetlen szerver van μ kiszolgálási sebességgel, amibe egyetlen forgalom érkezik, változó (pl. Markov-modulált) λ sebességgel. Azonban ahhoz, hogy a forgalmat a szerver kiszolgálja, a szervernek fel kell használnia arányos mennyiségű tokent is.

A tokenek folyamatosan r sebességgel gyűlnek egy külön gyűjtőben (bucket), azonban amikor van forgalom, akkor a szerver felhasznál belőlük. Ha a tokenek elfogynak, akkor a szerver csak r sebességgel szolgálja ki a forgalmat (annak megfelelően, ahogy a tokenek gyűlnek).

Token bucket

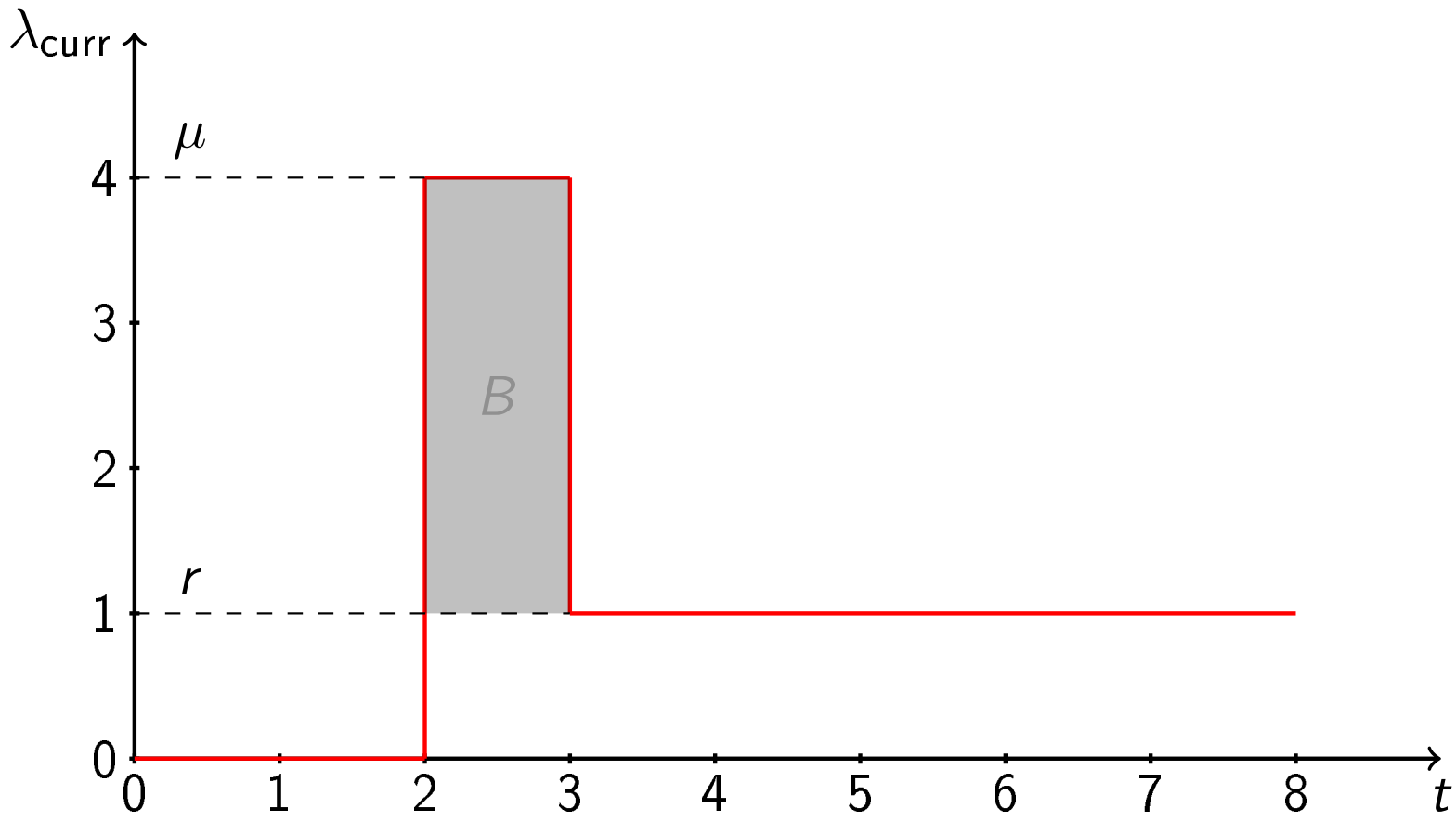
A token bucket egy forgalomnak hosszú távon r átlagos sebességet biztosít; azonban abban az esetben, ha egy ideig a forgalom r alatti sebességgel érkezett, akkor utána a forgalom sebessége igény esetén r-nél magasabb is lehet egy rövid ideig (amíg a tokenek kitartanak).

Végeredményben a token bucket hosszú távon r átlagos sebességet engedélyez, azonban a rövid ideig tartó magasabb sebességű forgalmat (burst) átengedi.

Ilyen szempontból pont ellentétes a szerepe a bufferhez képest; a buffer a rövid burst-ös forgalmakat kisimítja, a token bucket viszont átengedi.

Token bucket

Token bucket által szabályozott forgalom változása időben:



Mire jó a token bucket?

Mire jó a token bucket?

Egyrészt a token bucket használható arra, hogy ha éppen van szabad kapacitás a szerverben/hálózatban, akkor a burst hamar kiürül, és nem húzódik el későbbre, amikor esetleg nagyobb lesz a rendszer terhelése.

Másrészt a token bucket révén lehet egy szerver kimenetének burst-összégét szabályozni; ilyenkor az egész szerverhez tartozik egy token bucket, és minden átmenő forgalom ugyanazokat a tokeneket használja. A token bucket általában véges méretű, és a B mérete szabályozza azt, hogy mekkora méretű burst-öt szeretnénk átengedni, az r token érkezési ráta pedig azt, hogy a burst-ös időszakon kívül mekkora sebességet garantálunk.

Mire jó a token bucket?

Harmadrészt token bucketek használhatóak olyan helyzetben is, amikor egy nagy kapacitású szerver sok forgalmat szolgál ki. Ilyenkor minden egyes forgalomnak van egy saját token bucketje, amit csak az adott forgalom használ, és a többi token buckettől függetlenül töltődik.

Ilyenkor általában $\mu \gg \lambda, r$, és egy forgalom szempontjából a token bucket a fő szabályozó: rövid ideig ugyan a forgalom kaphat magas kiszolgálási sebességet, de hosszú távon átlagosan r sebességre van lekorlátozva.

Még token bucketek használata mellett is előfordulhat, hogy a forgalmak ütköznek; ilyenkor a további korlátozáshoz a forgalmak közötti priorizálás szükséges (pl. szigorú priorizálás vagy WFQ, ezeket már néztük).

Mire jó a token bucket?

Negyedrészt token bucketek használhatóak arra is, hogy egy forgalom múltját (némileg korlátozott mértékben) nyomon kövessük, és az egyes forgalmak között a múltjuk alapján priorizáljunk.

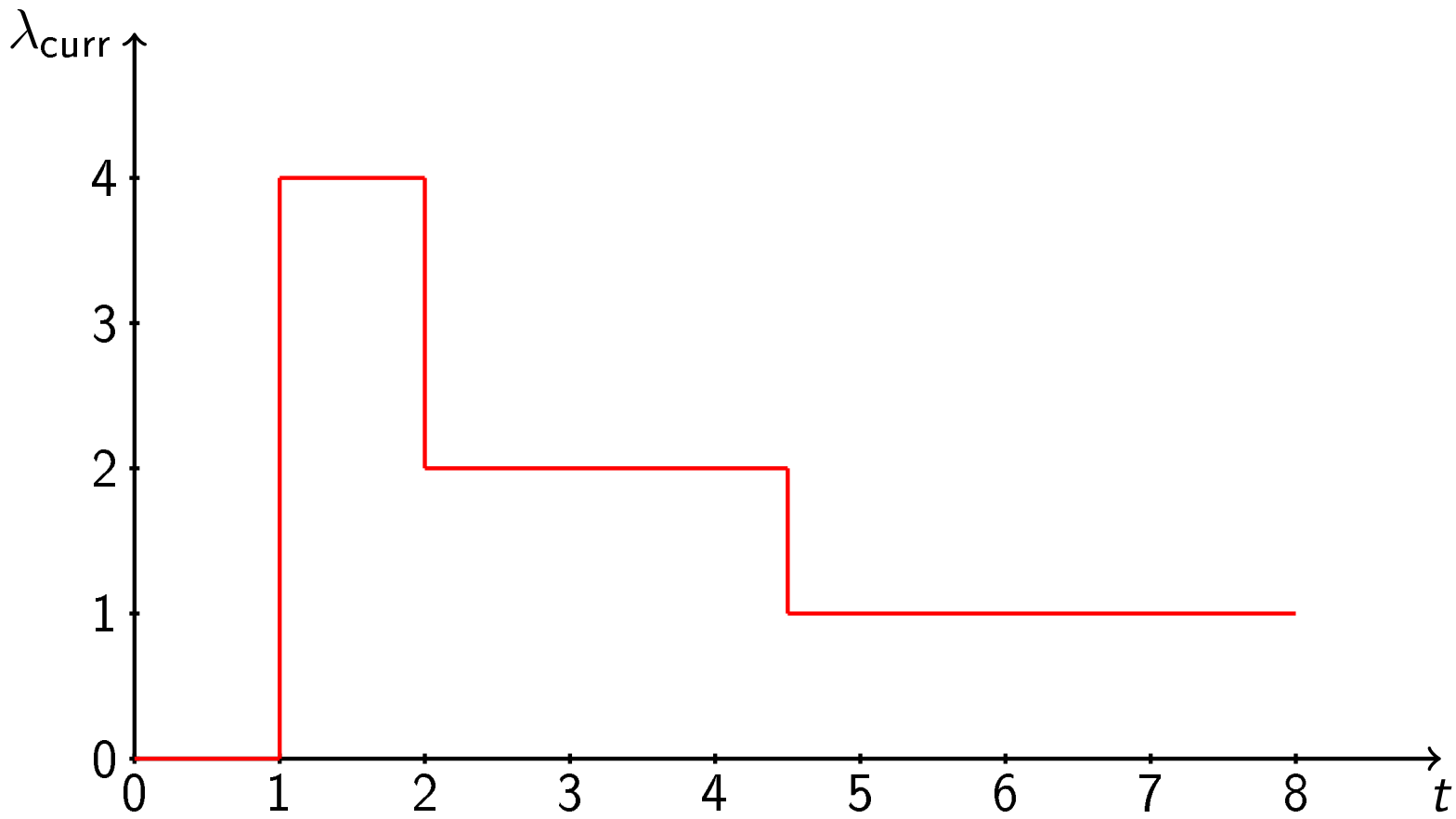
Ilyenkor egy forgalomhoz több token bucket is tartozhat különböző r és B értékekkel (kisebb r-hez nagyobb B), és a forgalom egyszerre az összes token bucketből elhasznál tokent.

A különböző token bucketek azt mérik, hogy különböző időskálákon mennyire volt aktív az adott forgalom. Egy ideig nagy sebességgel kap kiszolgálást, de ahogy sorban kiürülnek a token bucketek, lépcsőzetesen egyre kisebb sebességre korlátozódik.

Ezt még lehet kombinálni WFQ-val is $\rightarrow$ multi-timescale fairness.

Mire jó a token bucket?

Két token bucket által szabályozott forgalom változása időben:



Token bucket = negatív buffer

Tekintsük a következő modellt: adott egy fluid szerver μ kiszolgálási sebességgel, amelyhez tartozik egy (véges vagy végtelen) buffer, ÉS egy token bucket is (ami r rátával töltődik).

A szerveren áthalad egy forgalom időben változó λ rátával (pl. Markov-modulált); tegyük fel, hogy λ összes lehetséges értéke kisebb, mint μ.

Ekkor...

- ha $\lambda > r$, akkor a token bucket ürül λ_r rátával, de a buffer üres marad egészen addig, amíg a token bucket ki nem ürül, és csak azután kezd el a forgalom a bufferben is gyűlni;
- ha $\lambda < r$, de van a bufferban adat, akkor a szerver r rátával szolgál ki, a buffer ürül $r - \lambda$ rátával, és amint kiürül, elkezdenek gyűlni a tokenek a token bucketben.

Vegyük észre, hogy bármilyen helyzetben a token bucket és a buffer közül pontosan az egyik üres!

Token bucket vs. buffer

Ez motiválja azt, hogy ha $X(t)$ a buffer szintje t-kor és $Y(t)$ a token bucketek mennyisége t-kor, akkor tekintsük a

$$Z(t) = X(t) - Y(t)$$

mennyiséget. Ekkor

$$\begin{aligned} Z(t) > 0 &\iff Z(t) = X(t) \text{ és } Y(t) = 0; \\ Z(t) < 0 &\iff Z(t) = -Y(t) \text{ és } X(t) = 0, \end{aligned}$$

és $Z(t)$ teljesen leírja a buffer és a token bucket együttes állapotát.

Ha λ Markov-modulált, akkor a háttér-folyamat és $Z(t)$ szintjének elemzése hasonlóan elvégezhető, mint a Markov-modulált fluid szerver esetében.

Adaptív és elasztikus forgalmak

Alapesetben egy buffer egészen addig fogad forgalmat, amíg meg nem telik, onnantól pedig a forgalom visszaszabályoz (vagy elvész).

A forgalom jellegétől függően a visszaszabályozásnak lehetnek rossz mellékhatásai a hálózatra. Kétféle forgalomtípust különböztetünk meg:

- adaptív: a visszaszabályozás nem változtatja meg a forgalom jövőbeli viselkedését (pl. a Markov-modulált feltételezésben ez is benne van);
- elasztikus: fix adatmennyiséget szeretne átvinni, kisebb sebesség esetén tovább tart.

Ha sok elasztikus forgalom van a hálózatban, akkor a bufferek jelenléte hajlamos ezeket „összehangolni”: egyszerre lesz az egész hálózat terhelt, és egyszerre is ürülnek ki.

Random Early Drop

Ennek kezelésére szolgáló eszköz a Random Early Detection/Drop (RED). Ilyenkor egy buffer, ha már közel van a telítődéshez, már azelőtt elkezdi visszaszabályozni az érkezést alacsonyabb sebességre (technikailag telítődés előtt eldobál csomagokat, innen a neve).

Ezzel ugyan a hálózat teljesítményét picit visszafogja, cserébe viszont megelőzi az előző fajta „hullámzás” kialakulását.

Mi nem modelleztük, de pl. TCP-nél ezek a hullámzások ráadásul még jobban visszafogják a hálózat teljesítményét, mert az éppen elérhető sebességre való lejjebb/feljebb szabályozás nem instant, és emiatt a hullámok jelentős kihasználatlanságot is eredményeznek a hálózatban, amihez képest határozottan előnyösebb a RED.

Teljesítményelemzés

Teljesítményelemzés: ha lehet, analitikus eszközökkel, egyébként (elsősorban bonyolultabb helyzetekben) szimulációval.

Az analitikus elemzéshez képest a szimuláció:

- rugalmasabb; szimulációval elemezhetőek olyan helyzetek is, amire az analitikus eredmények közvetlenül nem alkalmazhatóak (pl. többféle hálózati elem, többféle forgalom eltérő releváns teljesítményjellemzőkkel);
- kevésbé robusztus: az eredmények nem olyan általánosak.

Figyelni kell arra, hogy pontosan azt szimuláljuk és elemezzük, amire kíváncsiak vagyunk. A teljesítményjellemzők megválasztása is gondosságot igényel.