Hálózatok II
2006

9: Online Hozzáféréskontrol és Útvonalválasztás ATM Hálózatokban

Asynchronous Transfer Mode ATM

- Videokonferencia, video-on-demand:
- Hogyan lehet szélessávú kommunikációs hálózatban (pl. ADSL) ilyen applikációknál garantálni egy megadott Quality of Service-t (QoS)?
- Asynchron Transfer Mode (ATM): Az adatok egyenlő hosszúságú cellákban (53 Byte) asynchron módon kerülnek átvitelre.
- Különböző adatfolyamok cellái meg tudják osztani egy link kapacitását, a cellák a linken egymás után kerülnek átvitelre.
- Aszinkron itt azt jelenti, hogy egy adatfolyam cellái nem szükségszerűen periódikusan kapnak hozzáférést a linkhez.

Történet

- Telefontársaságok a hangátvitelt analóg 4 kHz, digitálisan 64 kbps támogatták
- A telefontársaságok vonalakat bocsátották rendelkezésre adatátvitelhez
  - ISDN: 64 + 64 + 16 kbps
  - T1 (1.544 Mbps)
  - T3 (44.736 Mbps)
- Ők kívántak lenni az elsődleges szolgáltatók adatátvitelhez is.
- Sokféle felhasználás, melyek különböző minőségi követelményeket támasztanak az adatforgalommal kapcsolatban
  - file átvitel: löketszerű forgalom (bursty), magas csúcsráta (peak)
  - Adatbázis hozzáférés: löketszerű forgalom, rövid várakozási idő
  - Multimedia: szinkronizált
  - Video: 6 MHz analog, 1.2-200 Mbps digital
- Hogyan lehet ezeknek eleget tenni?

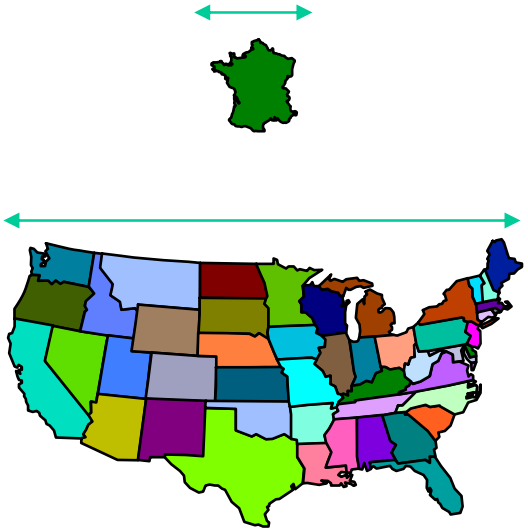
ATM

- **Asynchronous Transfer Mode**
- Minden csomagot ellátunk egy virtuális kapcsolat azonosítóval (Virtual Connection ID: VCI)

- Kis csomagok jó valós idejű viselkedést engednek meg
- Fix méretű csomagok (cellák) gyors kapcsolást tesznek lehetővé

ATM cella -- Miért 53 byte?

- Hangátviteli felhasználásokhoz kis cellák előnyösek
  - 10 ms –nél nagyobb késésnél visszhang elnyomásra van szükség (bonyolltabb)
  - minden adat-byte (payload byte) 125 µs –ot használ fel (64kbit/sec = 8kbyte/sec)
- Adatátviteli felhasználásokhoz nagy cellák előnyösek
  - Minden cellában 5 byte fejléc
- Franciaország (Europai országok) 32 byte-ot szertett volna
  - 32 byte = 4 ms
  - Franciaország „~6 ms széles”
- USA, Australia 64 byte-ot javasolt
  - 64 bytes = 8 ms
  - USA „>20 ms széles”
- Kompromisszum: 48 byte „payload”



ATM-Hálózatok – Sávszélesség-foglalás, CAC

- Egy kommunikációs kapcsolat felépítésénél egy adatfolyamhoz a hívó és a fogadó között egy útvonalon minden linken a megfelelő sávszélesség lefoglalásra kerül, ami az adatfolyamnak garantáltan rendelkezésére áll.
- Az adatfolyamok sávszélességigényének összege egy linken nem lépheti túl a link kapacitását.
- ATM-hálózatokon egy hozzáféréskontrol (ang. Connection Admission Control CAC) mechanizmus szükséges, azaz a kapcsolatkéréseket, amelyek sávszélességigénye nem garantálható, el kell utasítani.

QoS-osztályok a Gyakorlatban

Megjegyzés: A gyakorlatban a szituáció összetett: ATM-hálózatok különböző QoS-osztályokat támogatnak, pl.

- CBR (constant bit rate): a megfelelő bit-rátát garantálni kell.
- rt-VBR(real time variable bit rate) interaktív kommunikáció esetén video- vagy audioátvitelnél az adatokat nem szükséges mindig ugynazzal a sebességgel küldeni, de garantálni kell
  - a maximális késést egy cella küldése és fogadása között,
  - a maximális késést egy adatfolyam két egymást követő cellája között és
  - az átlagos bit-rátát.
- nrt-VBR (non real time variable bit rate) pl. video- vagy audio-streaming esetén csak
  - a maximális késés két egymást követő cella között és
  - az átlagos bit-ráta releváns.
- UBR (undefined bit rate): pl. file-átvitel (ftp) esetén. Csak az a fontos, hogy megérkezzenek az adatok…

CAC és Útvonalválasztás ATM-Hálózatokban

- Az ATM-hálózatokat egy absztrakt szinten vizsgáljuk:
  a sávszélesség-foglalás problémát úgy tekintjük, mintha csak CBR osztály létezne.

A Probléma:

- Minden kapcsolatkérés specifikálja a küldőt, a fogadót és a sávszélességigényt.
- A hálózatnak minden kapcsolatkérésnél el kell dönteni, hogy a kérést elfogadja-e vagy elutasítja (CAC).
- Minden elfogadott híváshoz hozzá kell rendelni a küldő és a fogadó között egy útvonalat, amelyet az adatfolyam használ (routing).
- A döntéseket a jövőbeli kapcsolatkérések ismerete nélkül kell meghozni.
  Ezt a szcenáriót online-nak nevezzük. (Ellentéte: u.n. offline-szenarióban az algoritmus ismeri az input adatokat a kezdettől fogva.)
- Optimalizálási kritériumok: az elfogadott híváskérések
  - számát (vagy profitját) maximalizálni, vagy
  - sávszélességének összegének maximumát egy linken (congestion) minimalizálni,…

Online Algoritmusok Minőségének Értékelése

Hogyan lehet a megoldás minőségét értékelni egy online-szcenárióban?
A jövőbeli kapcsolatkérések nem ismertek.

- Legyen *I* egy maximalizálási- (minimalizálási-) probléma *P* egy instanciája.
- Legyen *A* egy online-algoritmus a *P* problémához és
- $M(A,I)$ az *A* algoritmus által kiszámított célfüggvényérték *I*-hez.
- Legyen OPT_I az optimális célfüggvényérték az offline-szcenárióban (amit egy algoritmus a futási idő korlátozása nélkül kiszámíthatna, ha az algoritmus *I*-t teljesegészében előre ismerné.)
- $\max_I \{ OPT_I / M(A,I) \}$ értékét az *A* algoritmus kompetitív rátájának (competitive ratio) nevezzük (minimalizálási problémáknál a kompetetív rátát mint $\max_I \{ M(A,I) / OPT_I \}$ definiáljuk).

A hozzáférés kontrollnál az elfogadott hívások profitjának maximalizálását vizsgáljuk (a hálózat üzemeltető profitjának maximalizálása).

Példa Online-Algoritmusra: Síkölcsönzés Probléma

- Egy kezdő síelő problémája:
  - Egy sífelszerelés ára: 30.000 Ft
  - Kölcsönzés ára: 3.000 Ft / nap
- Rossz stratégiák:
  - Azonnal vásárolni
    - Költség: 30.000 Ft
    - Az első nap után elolvad a hó, nem tetszik, vagy baleset történik…
    - A legjobb (offline) stratégia: egy nap kölcsönzés: 3.000 Ft
    - Kompetitív ráta: 30.000 Ft / 3.000 Ft = 10
  - Minden nap kölcsönözni
    - Kompetitív ráta (ha tetszik... Opt. stratégia: azonnal megvenni) $\rightarrow\infty$
- Optimális stratégia:
  - 10 napig kölcsönözni, 11. nap vásárlás
  - Optimalis: kompetetív ráta minden nap ≤ 2

CAC és Routing ATM-Hálózatokban [Awerbuch, Azar, Plotkin 93]

A problema:

- A hálózat a $G=(V,E)$ gráf által adott, $|V|=n$, $|E|=m$. Minden $e \in E$ élnek van egy kapacitása $u(e)$.
- A hálózathoz egymás után k kapcsolatkérés $\beta_1,\beta_2,\ldots,\beta_k$ érkezik. $\beta = (\beta_1,\beta_2,\ldots,\beta_k)$ a k kapcsolatkérés sorozata.
- Minden β_i kapcsolatkérés a $\beta_i = (s_i,t_i,b_i,\rho_i)$ négyes által adott, ahol
  - $s_i \in V$ a küldő,
  - $t_i \in V$ a fogadó,
  - b_i a sávszélesség-igény (átviteli ráta) és
  - ρ_i a hálózat üzemeltető profitja, ha β_i elfogadásra kerül. Feltesszük, hogy $\rho_i = n \cdot b_i$, azaz a profit egyenesen arányos a sávszélességhez. (A szorzó n egy normalizálás, amely később hasznos lesz).
- Az egyszerűség kedvéért feltesszük, hogy a kapcsolat minden elfogadott kapcsolatkérés végtelen ideig (a k-adik kérés beérkezése utánig) megmarad.

CAC és Routing ATM-Hálózatokban

Megjegyzés: A valóságot reálisabban leíró modelleket hasonló módszerekkel kezelhetünk és analizálhatunk. A modell kiterjesztéséhez azokra az esetekre, amikor

- a kapcsolatok legfeljebb $\leq T$ ideig tartanak, vagy
- a sávszélesség a kapcsolat fennállásának ideje alatt változik, vagy
- a profit legfeljebb egy F szorzóval tér el $n \cdot b_i$ -től,

lásd [Awerbuch, Azar, Plotkin 93] és [Plotkin 95].

Az alapul szolgáló technika ugyanaz:

Routing a legrövidebb utakon egy olyan élköltség-függvény szerint, amely exponenciális az él már lefoglalt kapacitásának a részarányában.

CAC és Routing ATM-Hálózatokban

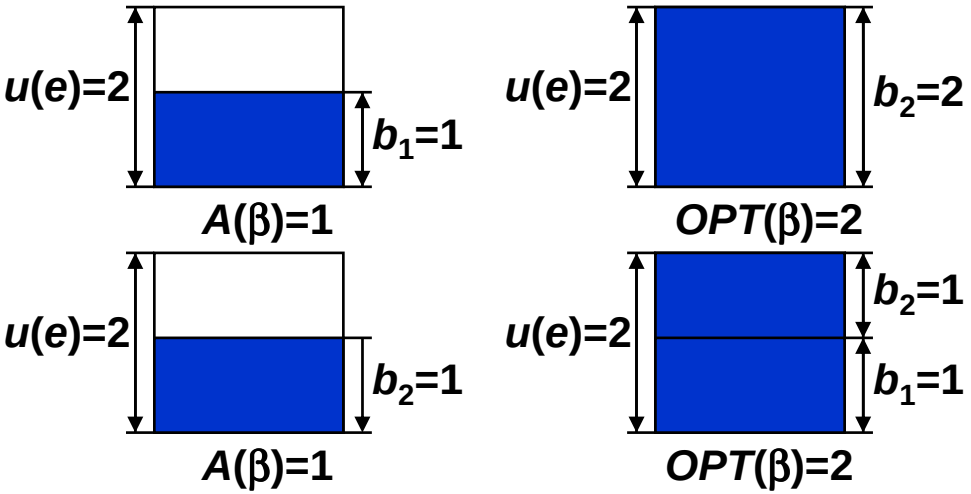
- Egy *A* algoritmus az online-CAC problémához a $\beta_1,\beta_2,\ldots,\beta_k$ kapcsolatkéréseket egymás után dolgozza fel ebben a sorrendben.
- Minden β_i kapcsolatkéréshez *A*-nak el kell dönteni a későbbi kapcsolatkérések ismerete nélkül, hogy β_i -t elfogadja, vagy elutasítja.
- Ha β_i -t elfogadja, akkor meg kell határozni egy P_i útat s_i -től t_i -hez *G*-ben, amin β_i -hez a sávszélesség lefoglalása kerül (routing).
- Minden időpontban minden $e \in E$ élre érvényesnek kell lenni, hogy az elfogadott β_i kérések b_i sávszélességeinek az összege, melyek az *e* élet használják, legfeljeb $u(e)$.
- Elfogadott kapcsolatkéréseket később nem szabad megszakítani.

CAC és Routing ATM-Hálózatokban

- Legyen $A(\beta)$ az összes elfogadott β_i kapcsolatkérés ρ_i profitjának összege, amit az A online-algoritmus β inputra elfogad.
- Legyen $OPT(\beta)$ az összes elfogadott β_i kapcsolatkérés ρ_i profitjának összege egy optimális megoldásban β inputra.
- Mivel egy online-algoritmus a jövőbeli kapcsolatkéréseket nem ismeri, nem találhat optimális megoldást.

  Példa.: Álljon a hálózat két csomópontbol és egyetlenegy linkből G=({*u,v*},{*e*={*u,v*}}), amely kapacitása $u(e)=2$. Tekintsünk két kapcsolatkérést $\beta=(\beta_1,\beta_2)$. Legyen $b_1=1$.

  - 1. eset: Ha az online-algoritmus β_1-t elfogadja, akkor legyen $b_1=2$. Ekkor $A(\beta)=1$ és $OPT(\beta)=2$.
  - 2. eset: Ha az online-algoritmus β_1-t nem fogadja el, akkor legyen $b_1=1$. Ekkor $A(\beta)=1$ és $OPT(\beta)=2$.



CAC és Routing ATM-Hálózatokban

- Azt mondjuk, hogy az *A* online-algoritmus kompetitív rátája $c \geq 1$, ha minden β inputra teljesül, hogy $OPT(\beta) / A(\beta) \leq c$.
- Egy ilyen algoritmust *c*-kompetitív-nek nevezünk.
- A profit, amit egy *c*-kompetitív online-algorithmus a CAC és routing problémához garantál, legfeljebb *c*-szer kisebb, mint az optimális profit.

CAC és Routing Algoritmus

Notáció:

- Legyen $\mu = 2n+2$ és $u_{min} = \min_{e \in E} u(e)$.
- Szükségünk van arra a feltételre, hogy minden β_i-ra teljesül, hogy $b_i \leq u_{min} / \log \mu$.
  Azaz, minden kapcsolat legfeljebb $(1 / \log \mu)$ részét foglalhatja le egy él kapacitásának. Ez a feltétel (legtöbbször) teljesül a gyakorlatban.
- Legyen $I(j)$ az online-algoritmus által elfogadott kapcsolatkérések indexeinek halmaza direkt β_j feldolgozása előtt.
- A normalizált terhelés $\lambda_e(j)$ az e élen közvetlenül β_j feldolgozása előtt $\lambda_e(j) = (\Sigma_{i \in I(j),\, e \in P} b_i) / u(e)$, az elfogadott β_i, $i < j$, kérések b_i sávszélességeinek összege, melyek P_i útvonala az e élet tartalmazza, osztva $u(e)$-vel.
- $I(k+1)$ jelöli az elfogadott kapcsolatkérések indexeinek halmazát $\lambda_e(k+1)$ pedig a normalizált terhelést az e élen a $\beta = (\beta_1, \beta_2, \ldots, \beta_k)$ sorozat feldolgozása után.

CAC és Routing Algoritmus

Az Algoritmus:

- Legyen β_j a következő feldolgozandó kapcsolatkérés. Definiáljuk minden *e* élhez $c_e(j) = u(e)\cdot(\mu^{\lambda_e(j)} - 1)$.
- Utaljuk minden *e* élhez a $w_j(e) = (b_j/u(e))\cdot c_e(j)$ költséget.
- Számítsunk ki egy legrövidebb *P* utat s_j -től t_j -hez a $w_j(e)$ élköltségek szerint. (pl. Dijkstra algoritmusával).
- Ha *P* költsége legfeljebb ρ_j, azaz $\Sigma_{e\in P} w_j(e) \le \rho_j$, akkor fogadjuk el β_j -t, egyébként utasítsuk el β_j-t.

Az alapötlet tehát:

- minden *e* élhez definiálunk egy költséget, amely exponenciális a $\lambda_e(j)$ normalizált élterhelésben.
- Ekkor a β_j kapcsolatkérést pontosan fogadjuk el, ha a legrövidebb út a küldőtől a fogadóig ezen költség alapján nem drágább, mint β_j profitja ρ_j.