

Better Online Deterministic Packet Routing on Grids

Számítógép-hálózatok és osztott rendszerek

Kádár Tamás Csaba, Kedves Nándor

November 24, 2016

1. Bevezetés
2. Modell és probléma
3. Algoritmus
4. Konklúzió

Bevezetés

Hálózatunkat a következő modellel írjuk le, melyet [1] cikk alapján építjük fel:

- $G = (V, E)$ irányított gráf
- B buffer méret, c élek kapacitása, ahol $B, c > 0$

A hálózat topológiája irányított egyenes, amely n vertexből áll

$$V = \{v_0, v_1, \dots, v_{n-1}\}, E = \{(v_{i-1}, v_i) \mid 0 < i < n\}$$

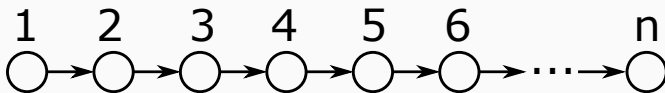


Figure 1: Lineáris hálózatmodell

Kérés (Request)

A kérést egy számhármassal adhatjuk meg, $r = (a_i, b_i, t_i)$

- a_i a forrás csomópont
- b_i a cél csomópont
- t_i az időpont amikor a kérés érkezik

, ahol $a_i, b_i \in V, t_i \in \mathbb{N}$

Minden time stepben, a routing algoritmus:

- törli a célba érkezett csomagokat
- minden más csomagra, beleértve az éppen beérkezőket is eldönti, hogy:
 - törli
 - tárolja az aktuális csomópont bufferjében
 - továbbküldi a következő vertexnek

Modell és probléma

Kiindulunk a már említett modellből és a következő modellt építjük fel:

- $G^{st} = (V^{st}, E^{st})$ irányított aciklikus végtelen gráf, amiben $c^{st}(e)$ az élek kapacitása. $V^{st} := V \times \mathbb{N}$, ahol minden $v \in V$ vertexnek végtelen számú másolata van a G^{st} -ben, melyet a $(v, t) \in V^{st}$ azonosít. $E^{st} := E_0 \cup E_1$, ahol az E_0 tartalmazza a csomópontok közötti éleket, melyek c kapacitással rendelkeznek és a E_1 a ugyanazon csomópont time steppek közötti élet tartalmazza, mely kapacitása B
- a kérés a következőképpen alakul $r_i^{st} = ((a_i, t_i), \text{row}(b_i))$, ahol a $\text{row}(b_i)$, a cél csomópont sorát jelöli

Az egyenes modelltől a rácsmodellig

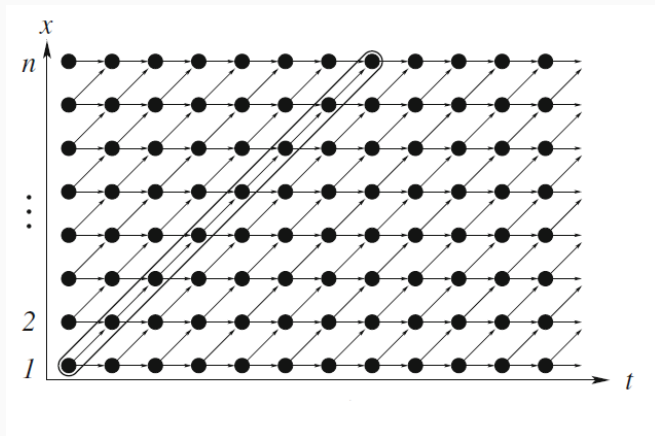


Figure 2: Döntött rácsos hálózatmodell [2]

Az egyenes modelltől a rácsmodellig

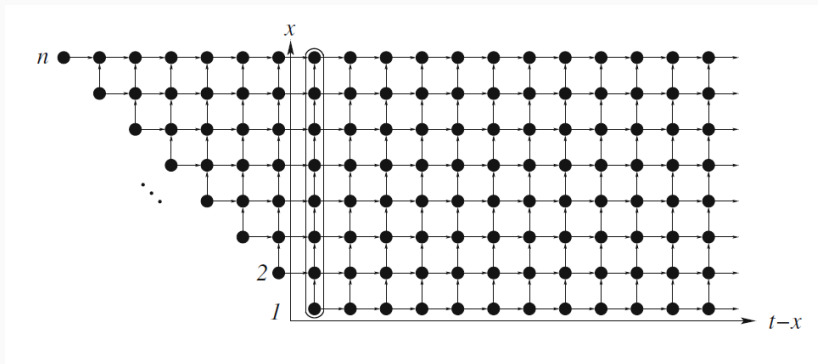


Figure 3: Nem döntött rácsos hálózatmodell [2]

Felépítjük a *sketch gráfot*, mely egy durvább megközelítése a rács modellnek. Felépítéséhez úgynevezett *tilingokat* használunk.

Tiling

- $\ell_h \times \ell_v$ részrács, ahol $\ell_h = \lceil \frac{6k}{5c'} \rceil$ és $\ell_v = \lceil \frac{6k}{5B'} \rceil$ ($c' = \lfloor c/5 \rfloor$, $B' = \lfloor B/5 \rfloor$ és $k = \log(1 + 3 \cdot p_{\max})$, ahol a p_{\max} később kifejtjük)
- ϕ_x és ϕ_y 2 *offset* paraméter segítségével határozzuk meg $((\phi_x + i \cdot \ell_h, \phi_y + j \cdot \ell_v), \text{ ahol } i, j \in \mathbb{N})$

A cikk által feldolgozott algoritmus 4 offsetet használ

$(\phi_x, \phi_y) \in \{-\ell_h/2.0\} \times \{-\ell_v/2.0\}$, ezeket nevezzük T_1, \dots, T_4 -nek.

Az egyenes modelltől a rácsmodellig

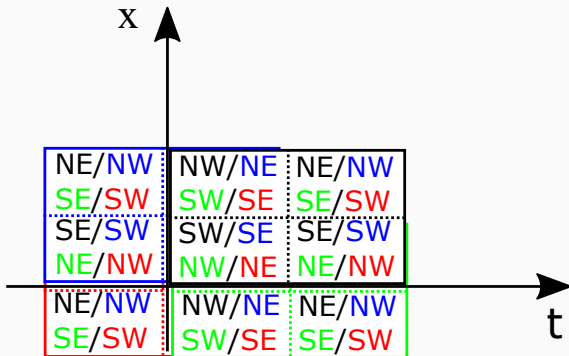


Figure 4: Sketch gráf

Definíció

Az $r_i = (a_i, b_i, t_i)$ kérés SW_j -ben található, ha a forrás vertex (a_i, t_i) a T_j csempe délnyugati részéhez tartozik.

Egy sketch gráfot indukál a T_j , melyet jelöljük $S_j := (V(S_j), E(S_j))$, ahol a $V(S_j)$ egy csempe halmaz a T_j -ből és nekik van $(s_1, s_2) \in E(S_j)$, ha $s_1 \neq s_2$ és $E^{st} \cap (s_1 \times s_2) \neq \emptyset$. Minden élhez egy egység kapacitást rendelünk.

- A sketch gráfot használjuk fel az *path packing* probléma megoldásához. Intuitíve a path packing modell hasonlít a packet routing modellhez, kivéve hogy ott nincsenek bufferek és hogy minden link e különböző kapacitással rendelkezik, melyet a következőképpen jelölünk $c(e)$.
- Formálisan egy kérés a következő alakba írható fel a G gráfban (a_i, D_i) , ahol $a_i \in V$ a forrás vertex és a $D_i \subseteq V$ célrészhalmoz.
- Legyen $P(r_i)$, mely jelölje azon pathek halmazát, melyek kiszolgálják a r_i kérést. Minden $p \in P(r_i)$ az a_i vertexel kezdődik és a vége a D_i halmazban található.

Algorithmus

Packet routing algorithmus pseudokód

1. Let R_t be a list of new requests, sorted by source-destination distance.
2. For each vertex v , let $R'_t(v)$ the first $B' + c'$ requests in R_t whose source is v . // filter requests
3. **for** each request $r_i \in \cup_v R'_t(v)$ **do**
4. **if** $r_i \in \text{Near}$ **then** ROUTE-NEAR(r_i)
5. **else**
6. Let $j \in \{1, \dots, 4\}$ be s.t. $r_i \in SW_j$ // classify r_i
7. $\text{sketch}_i \leftarrow \text{IPP}(S_j, \text{accepted}_j, r_i)$ // lengths bounded by p_{\max}

Packet routing algorithmus pseudokód

8. $init_i \leftarrow \text{INITIAL-ROUTE}(accepted_j, r_i)$
9. **if** $sketch_i \neq \text{REJECT}$ and $init_i \neq \text{REJECT}$ **then**
10. add r_i to $accepted_j$
11. $\text{DETAILED-ROUTE}(r_i; init_i; sketch_i)$ // update routes
12. **else** Reject r_i
13. **end if**
14. **end if**
15. **end for**

Az algoritmus hatékonyságának növelése érdekében a t időben érkező kéréseket megsűrjünk a következő módon:

- $R_t = t$ időben érkezett kérések
- $R'_t =$ kérések kezdőpont-végpont közötti távolság szerint rendezve
- input = minden v csomópont esetében az első $B' + C'$ kérés R'_t -ből, ahol a kérés kezdőpontja v

A kéréseket két csoportba soroljuk:

- NEAR kérés, abban az esetben ha a kezdőpont és a végpont is egy csempében vannak, vagyis $b_i - a_i \leq l_v$
- FAR kérés, ha a kérés két végpontja különböző csempékbe tartozik

Amennyiben FAR kérésről van szó, kiválasztjuk a megfelelő sketch gráfot aminek a SW kvadrantjába esik ez az r_i request.

IPP (Integral Path Packing) algoritmus

Az IPP algoritmus [3] vagy elutasítja a r_i kérést, vagy egy utat ad vissza egy csempék egy szekvenciáján a kezdő csempéből a cél csempébe.

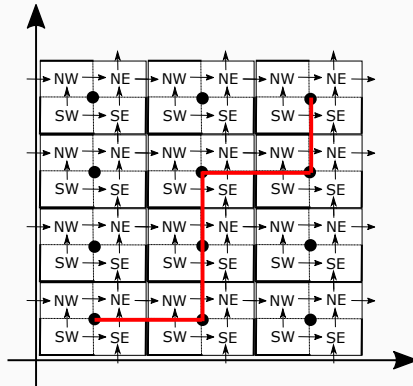


Figure 5: IPP algoritmus

Az Initial-Route algoritmus egy FAR típusú kérés esetében hajtódik végre.

Célja hogy egy egyenes mentén a kérést továbbítsa egy s csempe SW negyedéből.

Ha nem tudtuk egy egyenes mentén megtenni, csak tárolással vagy továbbküldéssel, a kérést eldobjuk

Egy FAR típusú kérést csak az IPP és az Initial-Route algoritmusok dobhatnak el.

Ha mindkét algoritmus megtartotta r_i -t akkor biztosak lehetünk hogy a csomag eljut a kezdőpontból a végpontba.

Céja hogy az előző két algoritmus eredményeit felhasználva meghatározza a tényleges utat G^{st} -ben.

Megjegyezzük hogy a leképezés során biztosak lehetünk, hogy lesz szabad út a kezdőpont és a végpont között.

Three different block environments are pre-defined and may be styled with an optional background color.

Default

Block content.

Alert

Block content.

Example

Block content.

Default

Block content.

Alert

Block content.

Example

Block content.

Konklúzió

Bemutattunk egy determinisztikus packet routing algoritmust, mely megoldott egy nyitott kérdést, de számos más kérdés még mindig nyitva hagyott. Például, hogy mi történik nem centralizált esetben?

| Ref. | Dim. | Comp. Ratio | Determ? | B, c |
|-----------|------|--------------------|---------|---------------------------------|
| [4, 5, 6] | 1 | $O(\log(n))$ | Yes | $B, c > \log n, B/c = n^{O(1)}$ |
| [7] | 1 | $O(\log^3(n))$ | No | $B \geq 2, c = 1$ |
| [8] | 1 | $O(\log^2(n))$ | No | $B \geq 2, c = 1$ |
| [4, 6] | 1 | $O(\log(n))$ | No | $B \in [1, \log n], c \geq 1$ |
| [5, 6] | 1 | $O(\log^5(n))$ | Yes | $[3, O(\log n)]$ |
| [5, 6] | d | $O(\log^{d+4}(n))$ | Yes | $[5, O(\log n)]$ |
| [1] | 1 | $O(\log(n))$ | Yes | $[5, O(\log n)]$ |

Table 1: Összehasonlítás más algoritmusokkal

References I

- [1] Guy Even, Moti Medina, and Boaz Patt-Shamir.
Better online deterministic packet routing on grids.
arXiv preprint arXiv:1501.06140, 2015.
- [2] Guy Even and Moti Medina.
Online packet-routing in grids with bounded buffers.
Algorithmica, pages 1–50, 2016.
- [3] Niv Buchbinder and Joseph Naor.
Improved bounds for online routing and packing via a primal-dual approach.
2006.

References II

- [4] Guy Even and Moti Medina.
An $o(\log n)$ -competitive online centralized randomized packet-routing algorithm for lines.
pages 139–150, 2010.
- [5] Guy Even and Moti Medina.
Online packet-routing in grids with bounded buffers.
pages 215–224, 2011.
- [6] Moti Medina Guy Even.
Online packet-routing in grids with bounded buffers.
abs/1407.4498, 2014.
- [7] Stanislav Angelov, Sanjeev Khanna, and Keshav Kunal.
The network as a storage device: Dynamic routing with bounded buffers.
Algorithmica, 55(1):71–94, 2009.

[8] Yossi Azar and Rafi Zachut.

Packet routing and information gathering in lines, rings and trees.

pages 484–495, 2005.

Kérdések?

Köszönjük a figyelmet!