Bežične mreže osjetila

Iterativna lokalizacija - trilateracija

Dario Barać, Mario Maričević
Diplomski sveučilišni studij računarstva, RITEH

Svibanj, 2022

Sadržaj

1 Opis problema

Za danu bežičnu mrežu osjetila definiranu skupom čvorova u 2D prostoru, vezama između čvorova i udaljenostima između čvorova koji su povezani, potrebno je pomoću postupka trilateracije odrediti položaj svih čvorova u relativnom koordinatnom sustavu sa čvorom inicijatorom u ishodištu (lokalizacija bez sidara). Pretpostavlja se da šum nije prisutan kod mjerenja udaljenosti i da je formacija grafa globalno kruta, odnosno da je moguće jednoznačno odrediti položaje svih čvorova u danom koordinatnom sustavu.

1.1 Ulazi u algoritam

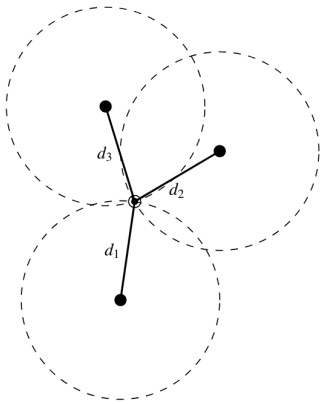
- Skup čvorova i veza
- Udaljenost između svakog para povezanih čvorova

1.2 Ograničenja

- Jedinstveni inicijator
- Dvosmjerna komunikacija između povezanih čvorova
- Apsolutna pouzdanost
- Globalno kruta formacija ulaznog grafa

2 Odabrani pristup

U radu [1] opisani su razni pristupi za mjerenje udaljenosti među čvorova i algoritmi za lokalizaciju. Za određivanje položaja trilateracijom (slika 1) potrebno je znati udaljenosti od čvorova koji znaju svoj položaj do čvora čiji položaj nije poznat. Neki od načina procjene udaljenosti su pomoću snage dobivenog signala (*Received Signal Strength - RSS*) ili vremena propagacije signala (*ToA, TDoA).*



Slika 1: Određivanje položaja trilateracijom [1]

Metode za lokalizaciju mogu se podijeliti na centralizirane i distribuirane. Kod centraliziranih metoda svi čvorovi udaljenosti do svojih susjeda prosljeđuju glavnom čvoru, koji određuje položaj

svih ostalih čvorova. Kod distribuiranih metoda čvorovi sami određuju svoj položaj pomoću informacija dobivenih od susjeda. Prednost centraliziranih metoda je to da se smanjuju zahtjevi (procesorska moć i memorija) svih osim glavnog čvora, ali u odnosu na distribuirane metode zahtjevaju značajno veći broj poslanih poruka i manje su pouzdane (prestanak rada glavnog čvora onemogućuje lokalizaciju). Zbog toga je odabran distribuirani pristup lokalizaciji.

2.1 Opis algoritma

Pristup koji je odabran kao rješenje problema sastoji se od dvije faze:

1. Određivanje inicijalnog krutog segmenta:
   - odabere se inicijator prije pokretanja algoritma npr. sa ID=0
   - inicijator pronađe dva susjeda koji su osim s njim povezani i međusobno (listu svih svojih susjeda šalje svim susjedima, koji odgovaraju odgovaraju sa listama zajedničkih susjeda i udaljenostima do njih)
   - inicijator koristi znanje o udaljenostima (do svoja dva susjeda i između njih) za izračun inicijalnog krutog segmenta - trokuta (Assumption Based Coordinate (ABC) algorithm [2]):
     - sebe stavi u ishodište (0,0)
     - prvog susjeda stavi na x os ($distance_{01}$, 0)
     - za drugog susjeda su moguća 2 položaja, odabere se položaj sa pozitivnom y vrijednosti
   - pomoću znanja o zajedničkim susjedima i udaljenostima do njih, inicijator u petlji trilateracijom dodaje u inicijalni kruti segment susjede za koje je to moguće (oni koji imaju barem 2 zajednička susjeda s inicijatorom, koji već jesu u krutom segmentu)
   - inicijator pošalje tim susjedima njihov položaj
2. Dodavanje ostalih čvorova u kruti segment (lokalizacija preostalih čvorova):
   - inicijator i susjedi iz prve faze šalju svoj položaj svim svojim susjedima koji nisu već lokalizirani
   - ostali čvorovi računaju svoj položaj kada saznaju položaj od tri susjeda i šalju ga svojim susjedima čiji položaj ne znaju

2.2 Statusi

- `INITIATOR`, čvor koji definira inicijalni kruti segment lokalizirane mreže i određuje položaj svojih susjeda koji su u inicijalnom segmentu
- `WAITING_FOR_FIX`, inicijalno stanje svim čvorovima osim inicijatoru, označava da su potrebne dodatne informacije kako bi čvor mogao odrediti svoj položaj
- `LOCALIZED`, čvor koji zna svoj položaj - algoritam je završen kada svi čvorovi u mreži imaju ovaj status

2.3 Poruke

- *CommonNeighborQuery* - sadrži popis svih susjeda inicijatora, inicijator šalje ovu poruku svojim susjedima
- *CommonNeighborResponse* - odgovor susjeda inicijatoru, `None` ako nemaju zajedničkog susjeda ili `(neighbors, distances)` ako imaju zajedničkog susjeda (`neighbors` je lista zajedničkih susjeda, `distances` je lista udaljenosti do njih)
- *OwnPosition* - inicijator šalje ovu poruku čvorovima koji su dio inicijalnog krutog segmenta mreže, sadrži `((x,y), localizedNodes)` - `(x,y)` je položaj čvora primatelja a `localizedNodes` skup čvorova koji su također dio krutog segmenta
- *NeighborPosition* - sadrži `(x,y)`, položaj susjeda koji šalje poruku

2.4 Sadržaj memorije čvorova

- *neighbors* - lista susjeda
- *neighborDistances* - lista udaljenosti do svih susjeda čvora
- *neighborPositions* - lista duljine jednake kao lista susjeda, svaki element je `None` ili `(x,y)`, ovisno o tome zna li čvor položaj tog susjeda
- *nKnownNeighborPositions* - broj susjeda za koje čvor zna položaj
- *position* - sadrži `(x,y)`, položaj čvora
- *unvisitedNeighbors* - lista koju čvor inicijator koristi kod konstrucije početnog krutog segmenta formacije mreže, sadrži sve čvorove koji još nisu odgovorili na *CommonNeighborQuery* i inicijalno je jednaka listi svih susjeda
- *rigidSegment* - skup elemenata tipa `(node_id, node_pos)` u memoriji inicijatora, sadrži susjede inicijatora koji su dio inicijalnog krutog segmenta
- *commonNeighborLists* - skup parova `(node_id, common)` u memoriji inicijatora, `node_id` je ID susjeda, `common` je lista susjeda zajedničkih inicijatoru i tom susjedu. Svaki element liste `common` sadrži ID susjeda i udaljenost do njega

2.5 Pseudokod

Distribuirani algoritam za iterativnu lokalizaciju (1) - status INITIATOR

```
INITIATOR
    Sponetaneously
    begin
        send(CommonNeighborQuery{neighbors}) to neighbors;
    end
    Receiving(CommonNeighborResponse)
    begin
        unvisitedNeighbors ←unvisitedNeighbors − sender;
        if commonNeighborResponse != None then
            common ←commonNeighborResponse;
            commonNeighborLists ←commonNeighborLists + (sender, common);
        end if
        if unvisitedNeighbors = ∅ then
            n1, common1 ←x ∈R commonNeighborLists; /* any neighbor with common
            neighbors */
            dist01 ←dist(n1);
            n2, dist12 ←x ∈R common1; /* any common neighbor */
            dist02 ←dist(n2);
            rigidSegment ←defineInitialRigidSegment(dist01, dist02, dist12, n1, n2);
            commonNeighborLists -= {(n1, common1), (n2, common2)};
            addNeighborsToRigidSegment(rigidSegment, commonNeighborLists);
            for (neigh, pos) in rigidSegment do
                send(OwnPosition{pos, rigidSegmentids}) to neigh;
            end for
            send(NeighborPosition{position}) to neighbors − rigidSegmentids;
            become LOCALIZED;
        end if
    end
```

Distribuirani algoritam za iterativnu lokalizaciju (2) - status WAITING_FOR_FIX

WAITING_FOR_FIX

Receiving(CommonNeighborQuery)

begin

$initiatorNeighbors \leftarrow$CommonNeighborQuery;

$allCommon \leftarrow neighbors \cap initiatorNeighbors$;

if $allCommon$!= $\emptyset$ **then**

send(CommonNeighborResponse{$allCommon$, **dist**($allCommon$)}) **to** sender;

else

send(CommonNeighborResponse{$None$}) **to** sender;

end if

end

Receiving(OwnPosition)

begin

$position, localizedNodes \leftarrow$OwnPosition;

$localizedNeighbors \leftarrow (neighbors \cap localizedNodes)$ + sender;

send(NeighborPosition{$position$}) **to** $neighbors - localizedNeighbors$;

become LOCALIZED;

end

Receiving(NeighborPosition)

begin

$nPos \leftarrow$NeighborPosition;

$neighborPositions[sender] \leftarrow nPos$;

$nKnownNeighborPositions+ = 1$;

if $nKnownNeighborPositions = 3$ **then**

$position \leftarrow$**trilaterate**($neighborPositions, neighborDistances$);

$knownNeighbors \leftarrow \{n \in neighbors | neighborPositions[n] \neq None\}$;

send(NeighborPosition{$position$}) **to** $neighbors$ - $knownNeighbors$;

become LOCALIZED;

end if

end

Distribuirani algoritam za iterativnu lokalizaciju (3) - procedure

procedure **defineInitialRigidSegment**$(r_{01}, r_{02}, r_{12}, neigh_1, neigh_2)$
begin
$position \leftarrow (0,0)$; `/* set initiator position */`;
$x_1, y_1 \leftarrow (r_{01}, 0)$;
$x_2 \leftarrow \frac{r_{01}^2 + r_{02}^2 - r_{12}^2}{2r_{01}}$;
$y_2 \leftarrow \sqrt{r_{02}^2 - x_2^2}$;
$rigidSegment \leftarrow \{(neigh_1, (x_1, y_1)), (neigh_2, (x_2, y_2))\}$;
return $rigidSegment$;
end

procedure **trilaterate**$((x_1, y_1), (x_2, y_2), (x_3, y_3), (r_1, r_2, r_3))$
begin
$A \leftarrow -2x_1 + 2x_2$;
$B \leftarrow -2y_1 + 2y_2$;
$C \leftarrow r_1^2 - r_2^2 - x_1^2 + x_2^2 - y_1^2 + y_2^2$;
$D \leftarrow -2x_2 + 2x_3$;
$E \leftarrow -2y_2 + 2y_3$;
$F \leftarrow r_2^2 - r_3^2 - x_2^2 + x_3^2 - y_2^2 + y_3^2$;
$x \leftarrow \frac{CE - FB}{EA - BD}$;
$y \leftarrow \frac{CD - AF}{BD - AE}$;
return (x, y);
end

procedure **addNeighborsToRigidSegment**$(rigidSegment, commonNeighborLists)$
begin
$rigidSegmentUpdated \leftarrow true$;
repeat
$rigidSegmentUpdated \leftarrow false$;
for $(neigh, common)$ **in** $commonNeighborLists$ **do**
$rigidCommon \leftarrow \{(n_{id}, n_{dist}) \in common | n_{id} \in rigidSegment\}$;
if $|rigidCommon| \geq 2$ **do**
$ids \leftarrow \{n_{id} | n \in rigidCommon\}$;
$positions \leftarrow \{n_{pos} \in rigidSegment | n_{id} \in ids\} + (0,0)$;
$distances \leftarrow \{n_{dist} | n \in rigidCommon\} +$ **dist**(neigh);
$pos \leftarrow$ **trilaterate**$(positions, distances)$;
$rigidSegment \leftarrow rigidSegment + (neigh, pos)$;
$commonNeighborLists$ -= $(neigh, common)$;
$rigidSegmentUpdated \leftarrow true$;
endif
end for
until $rigidSegmentUpdate = false$
end

3 Analiza vremenske i komunikacijske složenosti

Kod analize mrežu čvorova predstavljamo neusmjerenim grafom (V, E) - skupom čvorova V i skupom veza između čvorova E. Broj čvorova je $n = |E|$, a broj veza je $m = |E|$. Algoritam se može podijeliti u dvije faze - izgradnja inicijalnog krutog segmenta i lokalizacija preostalih čvorova (dodavanje u kruti segment).

3.1 Analiza komunikacijske složenosti - broj poruka

U prvoj fazi inicijator I šalje $CommonNeighborQuery$ svim susjedima i svi odgovaraju sa $CommonNeighborResponse$ (ukupno $2 \times |neighbors_I|$ poruka). Nakon toga inicijator svim susjedima koje je dodao u kruti segment javlja njihov položaj ($|initialRigid - \{I\}|$ poruka - minimalno 2, maksimalno $|neighbors_I|$).

U drugoj fazi preostali čvorovi postaju lokalizirani kada saznaju položaj od barem tri susjeda, a nakon toga svoj položaj šalju preostalim susjedima. Svaki čvor koji nije u skupu $initialRigid$ od svakog susjeda sazna njegov položaj ili mu javi svoj položaj.

Broj poruka je jednak broju veza za koje barem jedan čvor nije u $initialRigid$:

$$m_{nIR} = |\{(i,j) \in E | i \notin initialRigid \vee j \notin initialRigid\}| \tag{1}$$

U rubnom slučaju kada dva susjeda neovisno saznaju svoje položaje i svaki od njih u istom trenutku drugom pošalje svoj položaj, broj poruka bi bio veći. Tada bi na vezi između tih susjeda bile poslane dvije poruke umjesto jedne kao što je očekivano. Pretpostavlja se da je komunikacija između dva čvora uvijek jednosmjerna u odabranom trenutku, što sprječava nepotrebno slanje dodatne poruke.

Za **ukupni broj poruka** vrijedi:

$$M = 2|neighbors_I| + |initialRigid| - 1 + m_{nIR} \tag{2}$$

$$M \leq 2|neighbors_I| + m \tag{3}$$

$$M = O(n^2) \tag{4}$$

3.2 Analiza vremenske složenosti

Za upit inicijatora i odgovor susjeda su potrebne dvije vremenske jedinice. Nakon toga (u trećem koraku) inicijator javlja položaj čvorovima u inicijalnom krutom segmentu i počinje lokalizacija preostalih čvorova u mreži.

U najgorem slučaju čvorovi u mreži su povezani tako da je u svakom koraku moguće lokalizirati samo jedan novi čvor (čvor lokaliziran u prethodnoj iteraciji je potreban za lokalizaciju sljedećeg).

Uz jediničnu vremensko kašnjenje, za **ukupni broj koraka** (vrijeme) vrijedi:

$$T \leq 3 + n - |initialRigid| \tag{5}$$

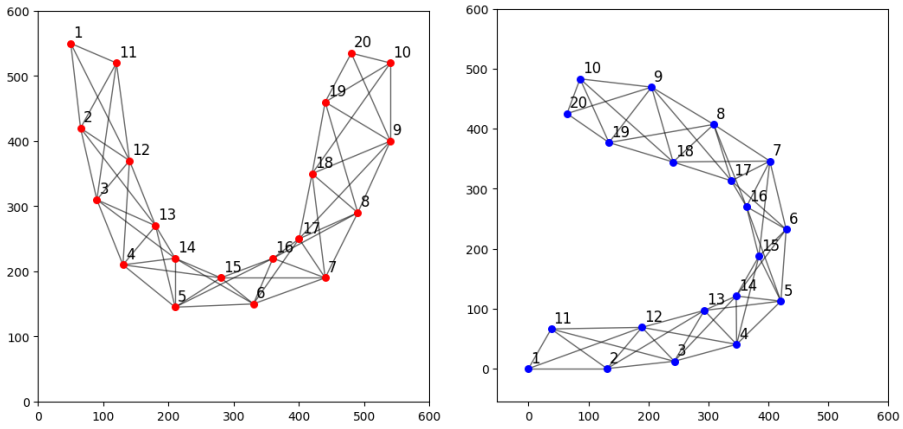
$$T \leq 3 + d((V,E)) \tag{6}$$

$$T = O(n) \tag{7}$$

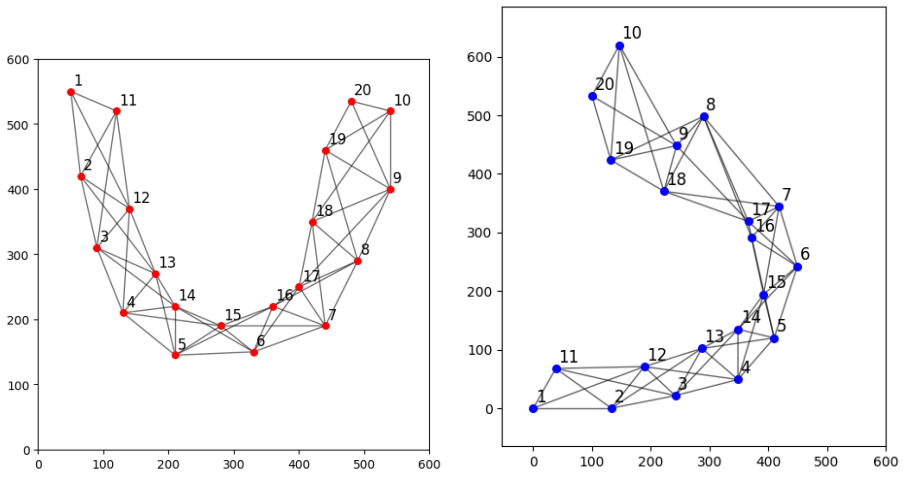
3.3 Testiranje implementacije

Algoritam je implementiran u Python-u i testiran pomoću Pymote [3] simulatora za bežične mreže osjetila. Lokalizacija je testirana na mrežama specifičnog oblika (*U-shaped*, *Ring-shaped* i potpuno povezani graf) i na raznim slučajno generiranim mrežama sa 20 čvorova. Simulacija lokalizacije bez šuma kod mjerenja udaljenosti je odrađena na svim mrežama. Simulacija sa šumom kod mjerenja je odrađena na mrežama specifičnog oblika.

Prije simulacije provjereno ako je moguće lokalizirati sve čvorove u mreži korištenjem testova za krutost i generičku globalnu krutost formacije mreže iz [4]. Svi čvorovi su uspješno lokalizirani u svim pokrenutim simulacijama.



Slika 2: Primjer lokalizacije bez šuma kod mjerenja (*U-shaped* mreža) - crveni čvorovi označavaju stvarni položaj, plavi čvorovi prikazuju čvorove lokalizirane u relativnom koordinatnom sustavu kojeg je definirao inicijator.



Slika 3: Primjer lokalizacije sa šumom kod mjerenja udaljenosti - lokalizacija počinje u čvoru 1, vidljivo je da greška procjene položaja s vremenom postaje sve veća jer se greške kod mjerenja udaljenosti zbrajaju kod svakog dodavanja novog čvora.

Literatura

[1] Roudy Dagher, Roberto Quilez. Localization in Wireless Sensor Networks. Nathalie Mitton and David Simplot-Ryl. Wireless Sensor and Robot Networks From Topology Control to Communication Aspects, Worldscientific, pp.203-247, 2014, 978-981-4551-33-5. ⟨10.1142/9789814551342_0009⟩. ⟨hal-00926928⟩

[2] Savarese, C., Rabaey, J. M., & Beutel, J. (n.d.). Location in distributed ad-hoc wireless sensor networks. 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221). doi:10.1109/icassp.2001.940391

[3] Arbula, D. and Lenac, K.: Pymote: High Level Python Library for Event-Based Simulation and Evaluation of Distributed Algorithms, International Journal of Distributed Sensor Networks, Volume 2013

[4] Eren, Tolga & Whiteley, Walter & Belhumeur, Peter. (2004). A theoretical analysis of the conditions for unambiguous node localization in sensor networks.