BİL465 BİLGİSAYAR AĞ YÖNETİMİ LABORATUVARI DERSİ

FİNAL PROJESİ

Projenin veriliş tarihi: 10.12.2019-12:00

Projenin son teslim tarihi: 14.01.2020-17:00

Proje sunumuna gelirken uygulamanızın çalışan halini göstermeniz yeterli olacaktır.

Distributed DFS with Neighbor Knowledge

The DFS algorithm *Neigh_DFS* uses a token to traverse the nodes of the network in a sequential manner. Since we need a way to know which nodes have been visited so that they are not visited again, a token may be used for this purpose. The token includes the visited node list which is appended by the node identifier of a node that is visited for the first time. The algorithm *Neigh_DFS* is depicted in Algorithm 3, where node *i* chooses a neighbor *j* to send the token only if it is not included in the list (*vislist*) of already visited nodes of the token.

Figure 3 shows the operation of *Neigh_DFS* in a sample network, where *n* equals 8, and edges of the tree are labeled by the time frame token traverses them. The contents of the token when it is first received by a node and the final token as received by the root node 4 are also shown.

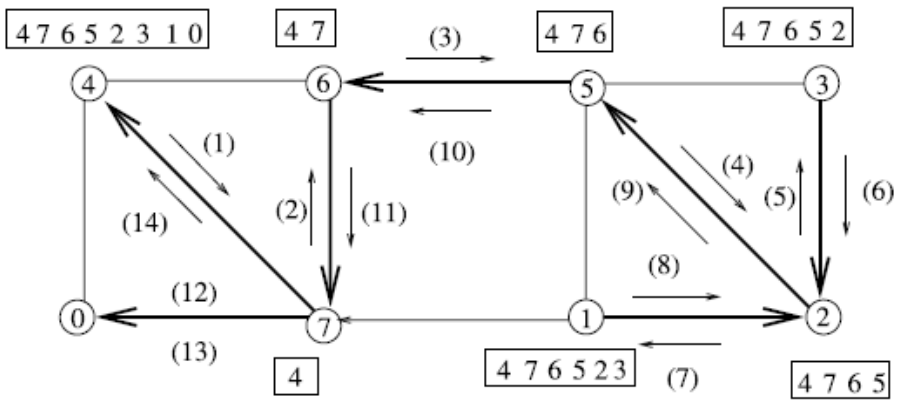

Fig. 3. *Neigh_DFS* execution example

Algorithm 3. *Neigh_DFS*

```
1:  int parent ← ⊥
2:  set of int vislist ← ∅
3:  message types token
4:
5:  if i = root then                                   ▷ root starts the algorithm
6:      parent ← i, choose j ∈ Γ(i)
7:      send token({i}) to j
8:  end if
9:
10: while true do
11:     receive token(j, vislist)
12:     if parent = ⊥ then                             ▷ token received first time
13:         parent ← j
14:     end if
15:     if ∃j ∈ Γ(i) \ {vislist} then                  ▷ choose an unsearched node if any
16:         choose j ∈ Γ(i) \ {vislist}
17:         send token(vislist ∪ {i}) to j
18:     else if i = root then
19:         exit                                       ▷ if all searched and root, terminate
20:     else                          ▷ if all searched and not root, return token to parent
21:         send token(vislist ∪ {i}) to parent
22:         exit                                       ▷ all nodes except root terminate
23:     end if
24: end while
```