Tree Algorithms

Thursday, May 22, 2014
7:54 AM

Broadcast

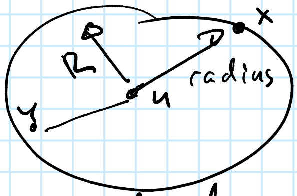
initiated by a single node : source
source wants to send a msg. to all other nodes

Distance: length (# hops) of a shortest path

Diameter D: largest distance between any 2 nodes

Radius R:

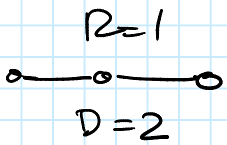
radius of a node u: largest distance from u to any other node



radius R of the network: smallest radius of a node

$$R \le D \le 2R$$

$R = D$: K_n $\qquad$ $D = 2R$:



time for broadcast

$\ge$ radius of source $\ge R \ge D/2 = \Omega(D)$

Message complexity of broadcast:

$\ge n-1$ (everyone has to recv. the msg.)

Clean network:

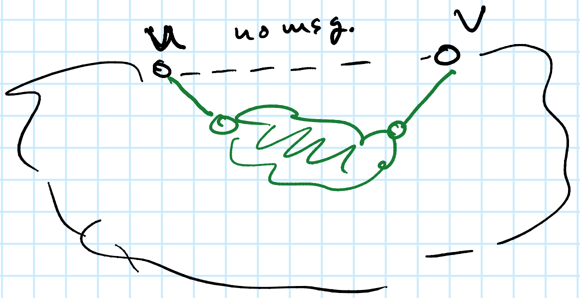
Graph (network) is clean if the nodes do not know (anything about) the topology.

Msg. compl. of broadcast in clean networks:

$\ge m$ (# edges)

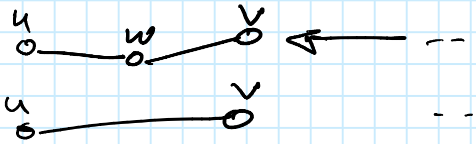
Proof:

at least one message over every edge



cannot distinguish edge $\{u,v\}$ from a whole subgraph hidden

(e.g.



$\Longrightarrow$ #msg. $\geq m$

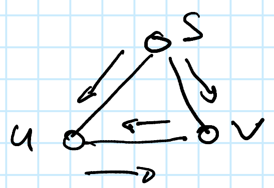
Flooding

1. Source (root) sends msg. to all neighbors
2. each other node v:
   when rcv. msg. for the first time, forward the message to all other neighbors

   (if you recv. msg. again, discard it)

Flooding solves broadcast

$\hookrightarrow$ works in an asynchr. network



message complexity:

at most one msg. per edge in each direction

msg. compl. $\leq 2m = O(m)$

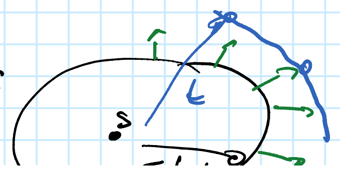
time complexity:

$\leq$ radius of source

show that by time $t \in \mathbb{N}$, all nodes at dist $\leq t$ from the source have received the message.

Proof by ind. on t

$t=0$: | $t \geq 1$: I.H.

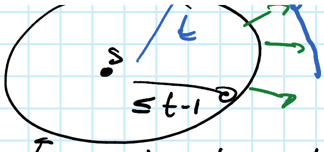


nodes at dist. t recv. these msg.

$\underline{t=0:}$

✓

$\underline{t \geq 1:}$



rcv. msg. by time $t-1$

recv. these msg. by time $\leq t$

✓

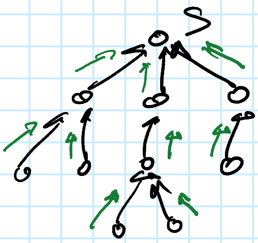
Convergecast

opposite of broadcast

assume rooted tree

broadcast gives a rooted tree

flooding — node from which msg. is received first is parent



Echo:

1. leaves send msg. to parents
2. inner node:
   after recv. msg. from all leaves, send msg. to parent

---

often combined with flooding — flooding/echo

→ use it for term. detection

→ time complexity of convergecast: height of tree

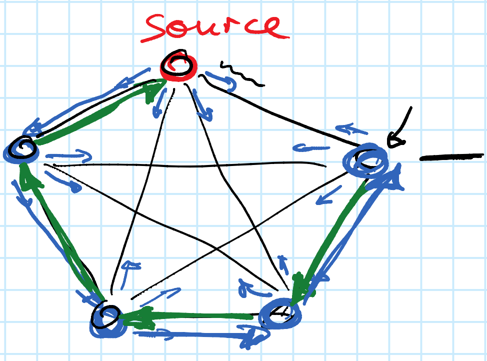
→ msg. compl.: $n-1$

→ use it to compute functions such as sum

Time Complexity of Flooding/Echo

flooding : $\leq D$

echo : height of tree



synchr:

BFS tree

breadth first search

shortest path tr.

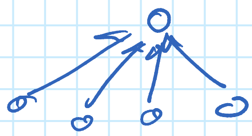
height $\leq D$

asynchronous:

height $\leq n-1$

best possible guarantee, even if $D=1$

K_n



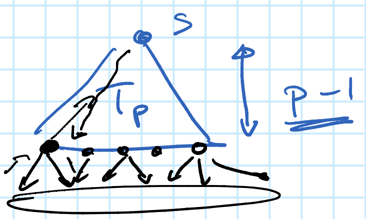
BFS tree

BFS Tree Construction

Algorithm 3.3 Dijkstra BFS

The algorithm proceeds in phases. In phase p the nodes with distance p to the root are detected. Let T_p be the tree in phase p. We start with T_1 which is the root plus all direct neighbors of the root. We start with phase $p = 1$:
repeat
    The root starts phase p by broadcasting "start p" within T_p.
    When receiving "start p" a leaf node u of T_p (that is, a node that was newly discovered in the last phase) sends a "join $p + 1$" message to all quiet neighbors. (A neighbor v is quiet if u has not yet "talked" to v.)
    A node v receiving the first "join p+1" message replies with "ACK" and becomes a leaf of the tree T_{p+1}.
    A node v receiving any further "join" message replies with "NACK".
    The leaves of T_p collect all the answers of their neighbors; then the leaves start an echo algorithm back to the root.
    When the echo process terminates at the root, the root increments the phase
until there was no new node detected

Idea: construct BFS tree level by level



time compl. to build level p

$p-1+2+p-1 = 2p$

time compl:

$TC \le 2\sum_{p=1}^{D} p = O(D^2)$

$$TC = 2\sum_{p=1}^{D} p = 2\,\frac{D(D+1)}{2} = O(D^2)$$

time: $O(D^2)$

message compl:

phase p: bcast/c.cast: $O(n)$

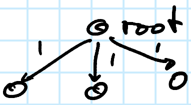
each edge ≤ 1 join msg / ≤ 1 ACK/NACK msg (in each dir.)

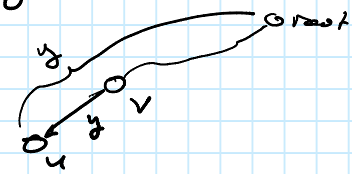
$MC = O(m + D \cdot n)$

Algorithm 3.4 Bellman-Ford BFS

Each node u stores an integer d_u which corresponds to the distance from u to the root. Initially $d_{\text{root}} = 0$, and $d_u = \infty$ for every other node u.
The root starts the algorithm by sending "1" to all neighbors.
if a node u receives a message "y" with $y < d_u$ from a neighbor v **then**
  node u sets $d_u := y$; v becomes the parent
  node u sends "$y + 1$" to all neighbors (except v)
end if

alg. used for computing routing tables (in Internet)



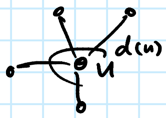


Time Compl: $\leq D$

by time t, nodes at dist. $\leq t$ from root learn their dist. to root

Message Compl: $O(m \cdot n)$

#msg. sent by node u of d(u)



$d(u) \cdot \underbrace{\text{"\#updates at node u"}}_{\leq n-1}$

$\leq 2(n-1)$ msg per edge

tight

trade-off between msg. & time compl.

best known:

time: $O(D \log^3 n)$

msg: $O(m + n \cdot \log^3 n)$