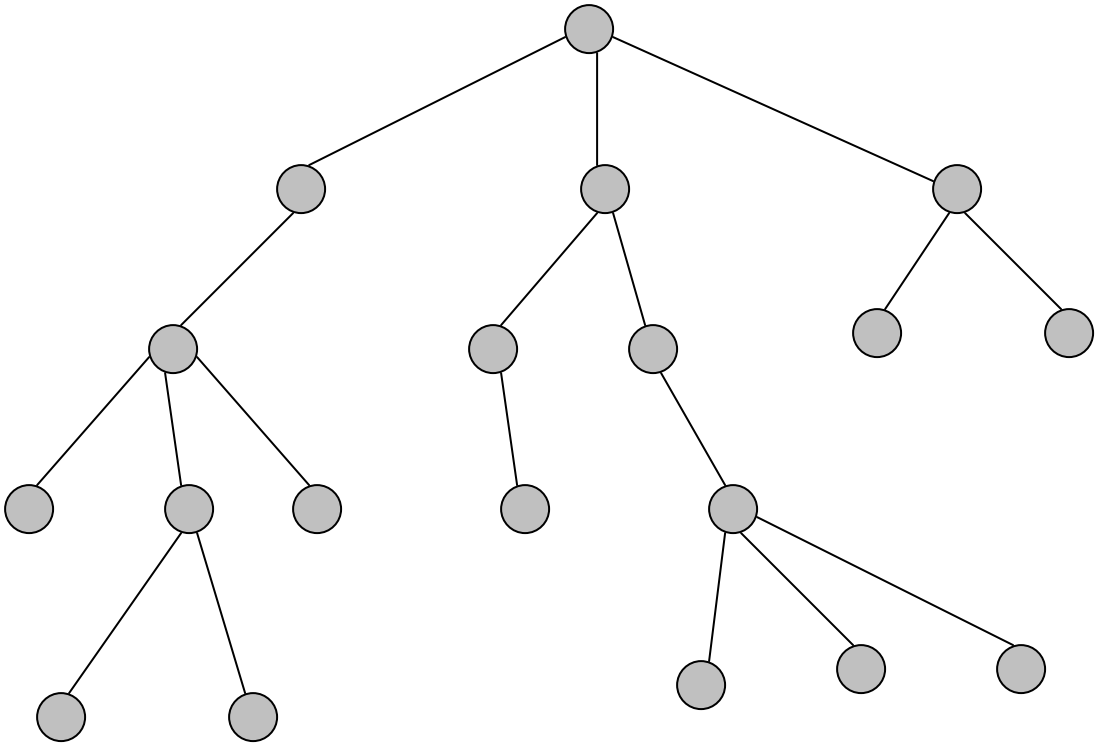
Near-Optimal Hot-Potato Routing on Trees

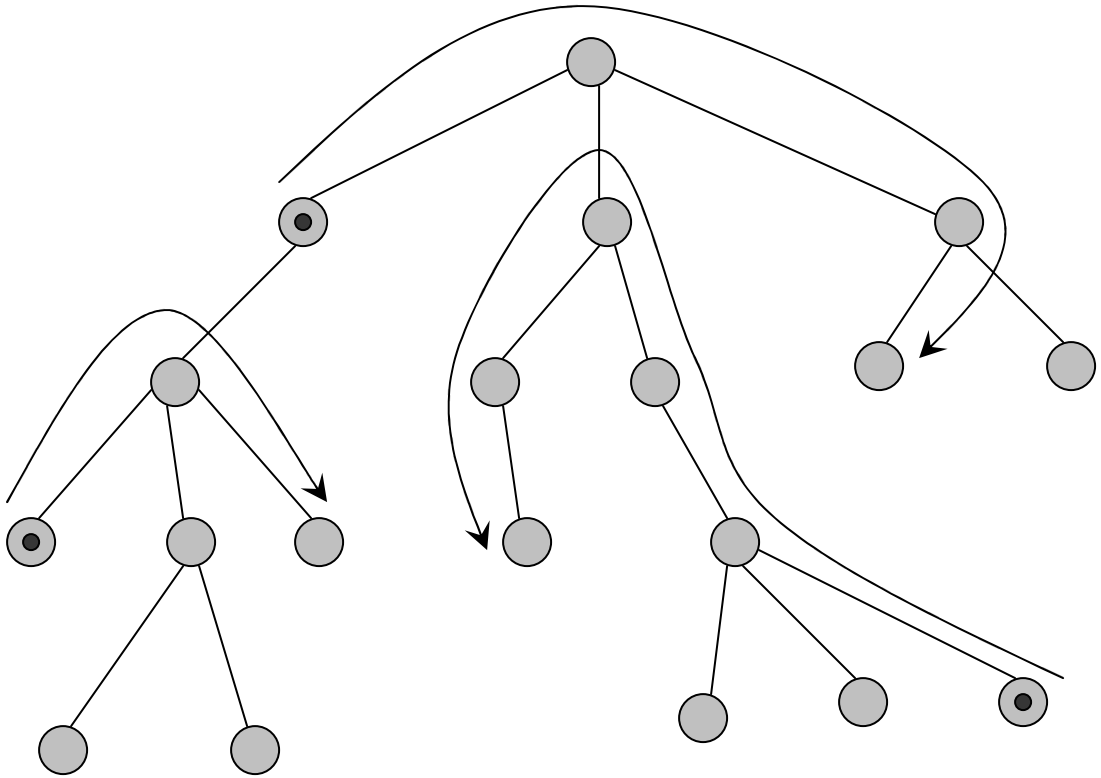
Costas Busch	Rensselaer Polytechnic Inst.
Malik Magdon Ismail	Rensselaer Polytechnic Inst.
Marios Mavronicolas	University of Cyprus
Roger wattenhofer	ETH Zurich

Trees



Trees are important in many networks
(i.e. spanning trees)

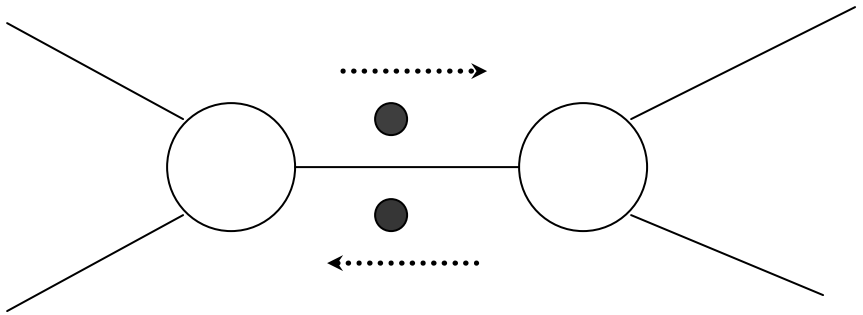
Routing on Trees



- Every node generates at most one packet
- Packets follow shortest paths

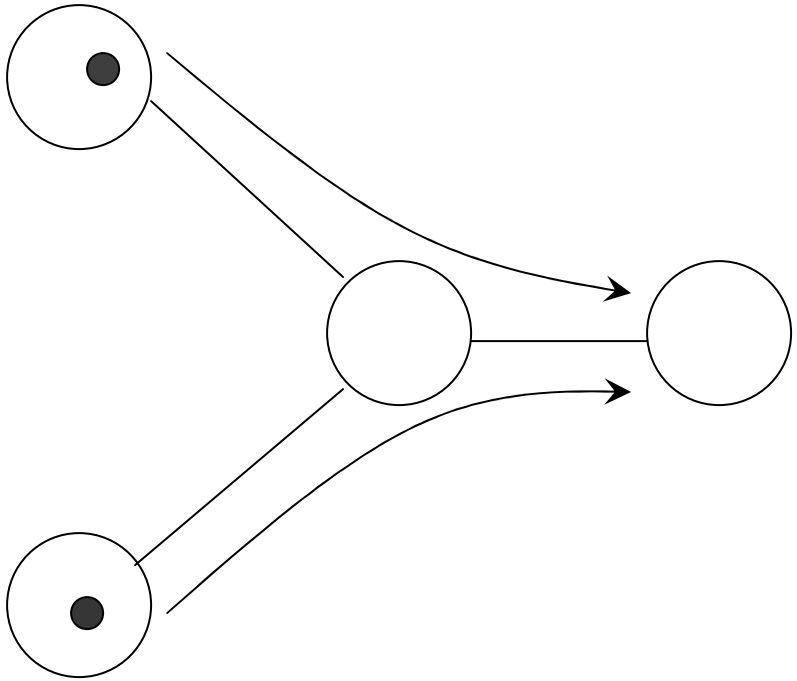
Network Model

- Synchronous network
- Bi-directional links
- One packet per time step



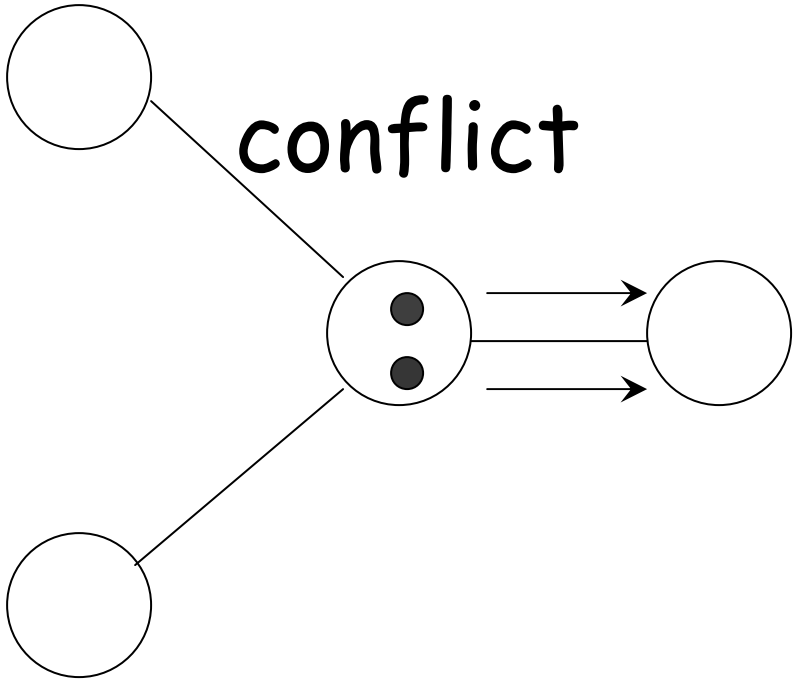
Hot-Potato Routing

Time 0

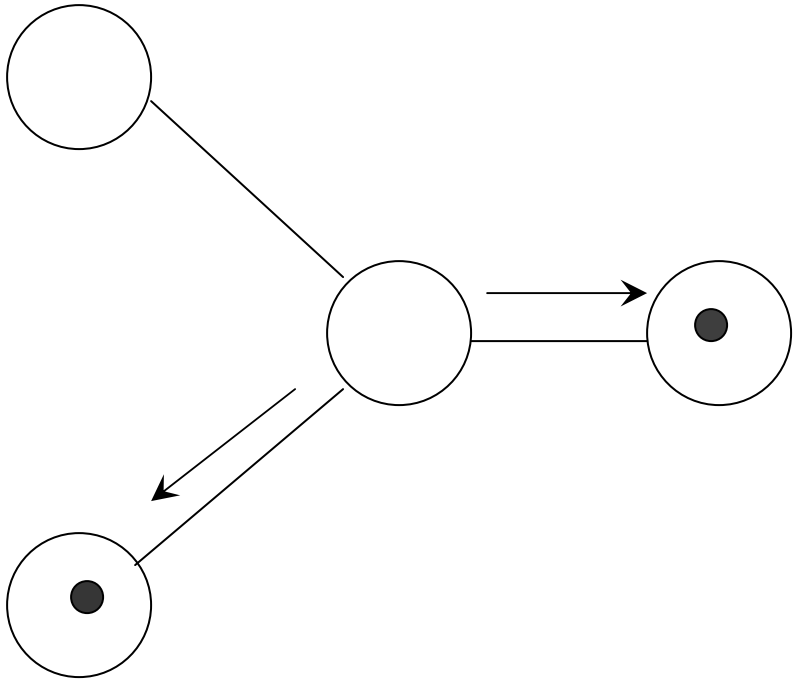


Buffer-less nodes

Time 1

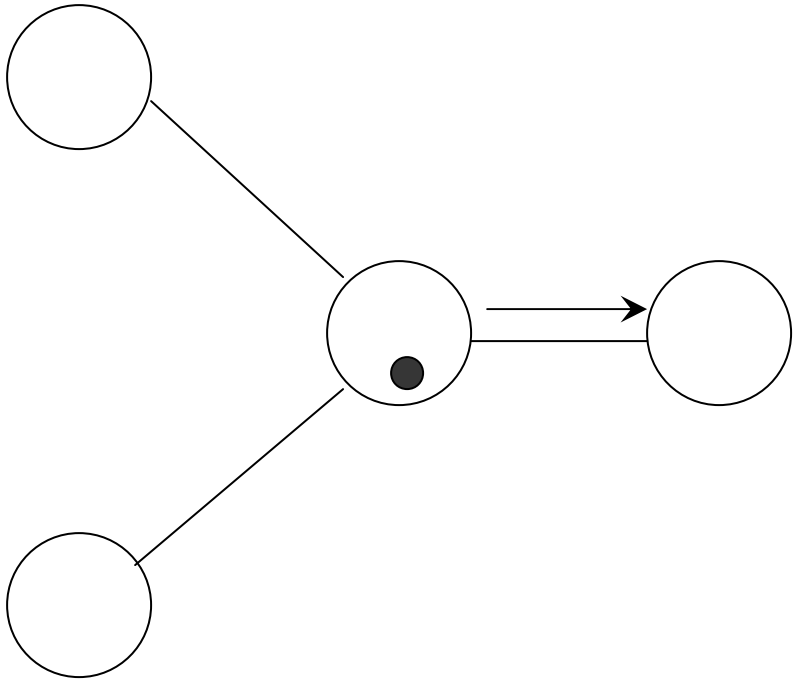


Time 2

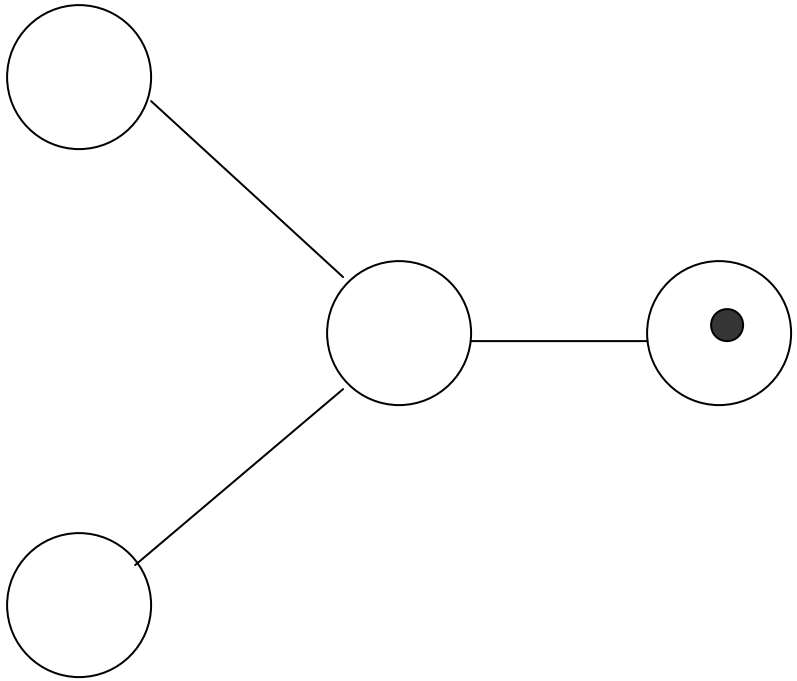


deflected

Time 3



Time 4



Hot-potato routing is interesting:

- Optical networks
- Simple hardware implementations
- Works well in practice:

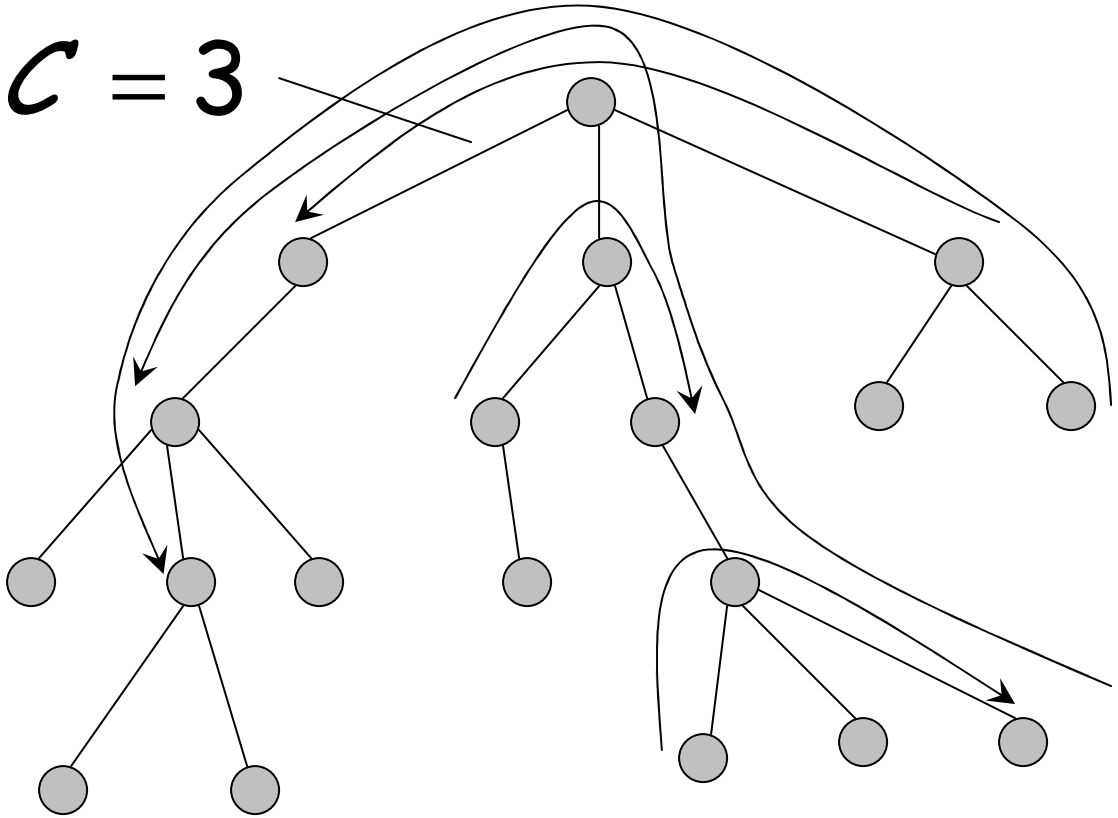
  Bartzis *et al.*: *EUROPAR 2000*

  Maxemchuck: *INFOCOM 1989*

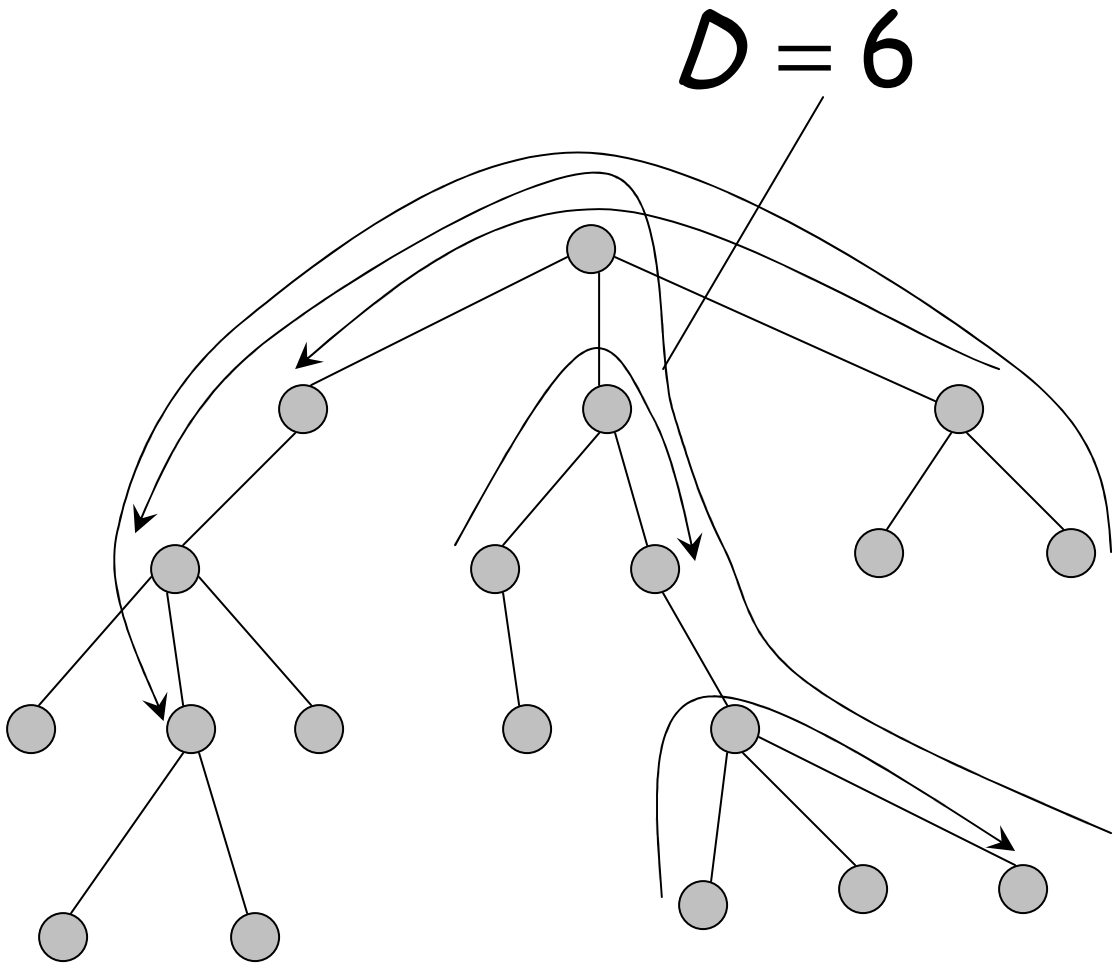
Objective: Find hot-potato algorithm which minimizes routing time

The time until the last packet is delivered to its destination

Congestion: Maximum numbers of packets that share an edge



Dilation: Maximum path length



A lower bound on Routing Time:

Congestion+Dilation

$$\Omega(C + D)$$

We want to find an algorithm close to this lower bound

Our contributions:

- Deterministic Algorithm: $O((\delta C + D)\log n)$

node degree

network size

- Randomized Algorithm: $O((C + D)\log^2 n)$

degree independent

Related Work for Trees

- Matching Routing [ACG94] [PRS97] [Z97]
- Direct Routing [AHLT98][BMMS04]
- Hot-Potato routing [RSW00]

Most results have routing time $O(n)$

(worst case bound for $O(C+D)$)

Presentation Outiline

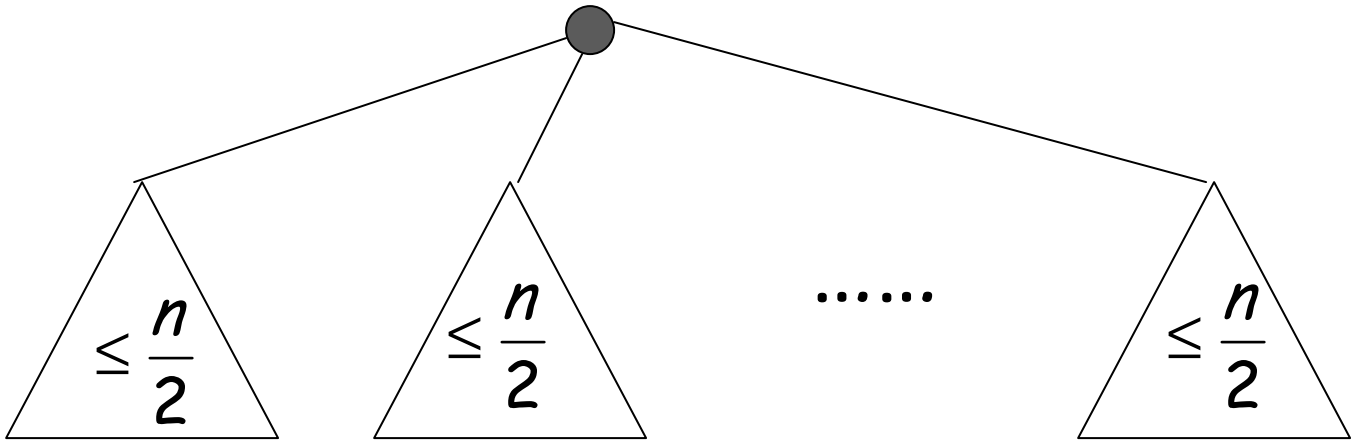
- Deterministic Algorithm
- Randomized Algorithm

Deterministic Algorithm

1. Divide time into phases according to short nodes

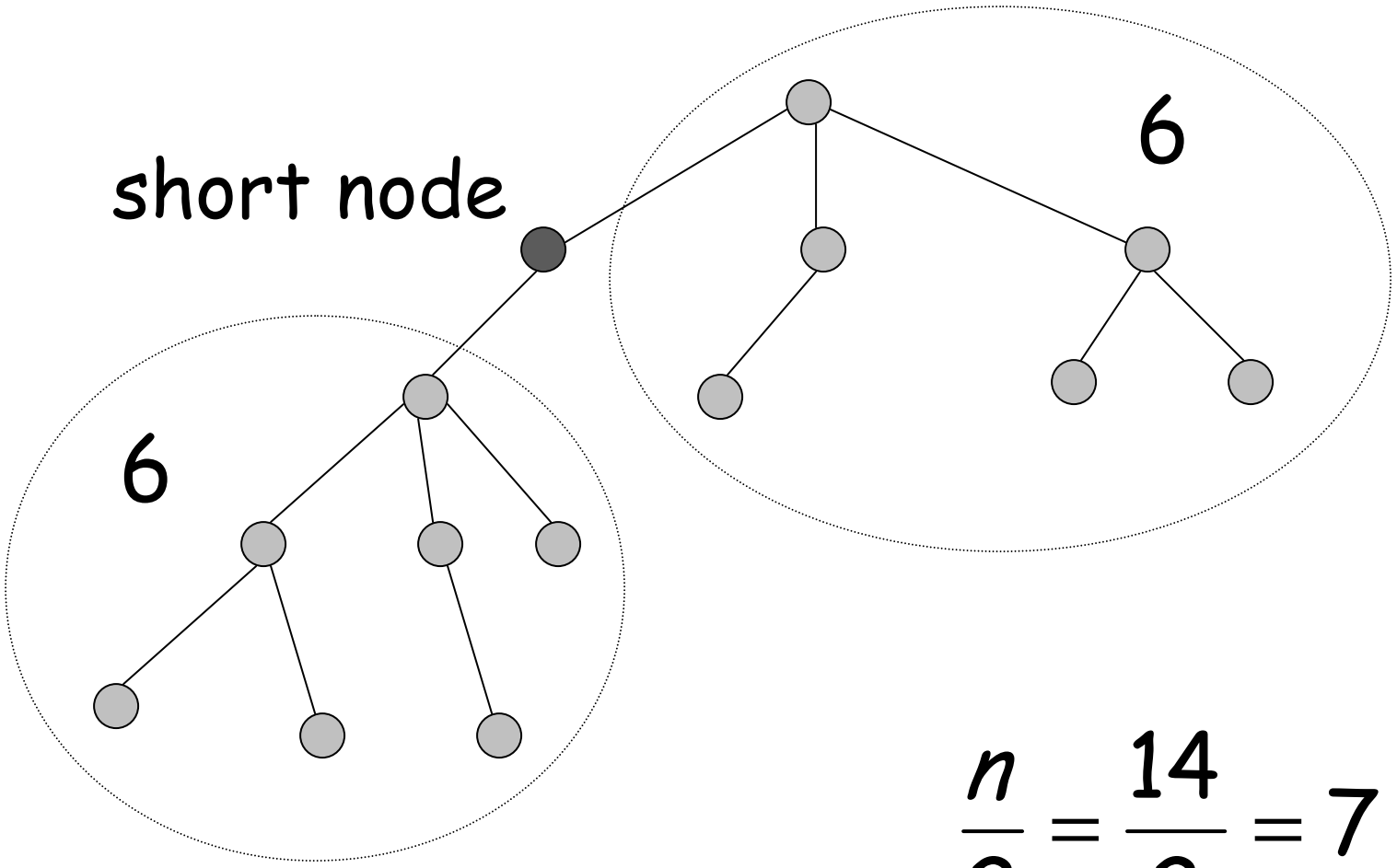
2. At each phase send packets to their destinations greedily

Short Node:



every subtree has at most $\frac{n}{2}$ nodes

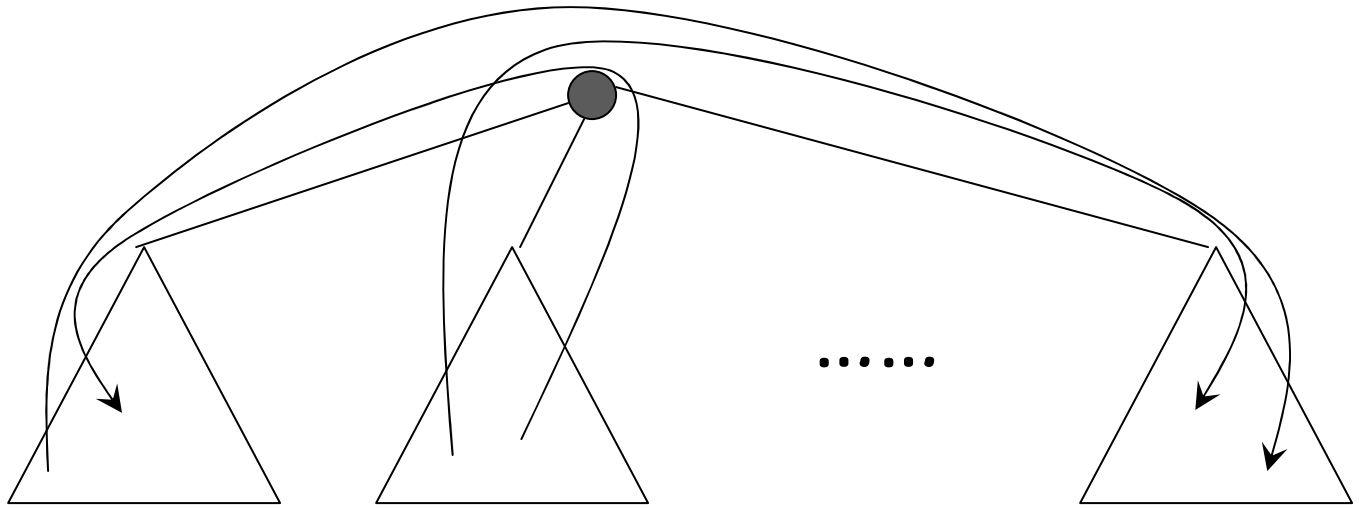
Example



$$\frac{n}{2} = \frac{14}{2} = 7$$

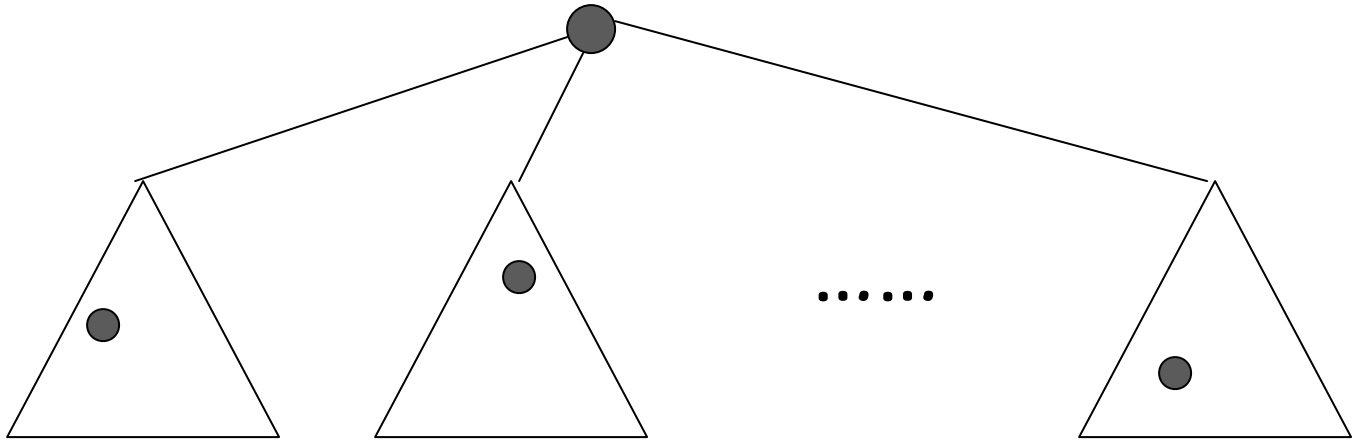
Phase 1:

Route packets that cross the short node



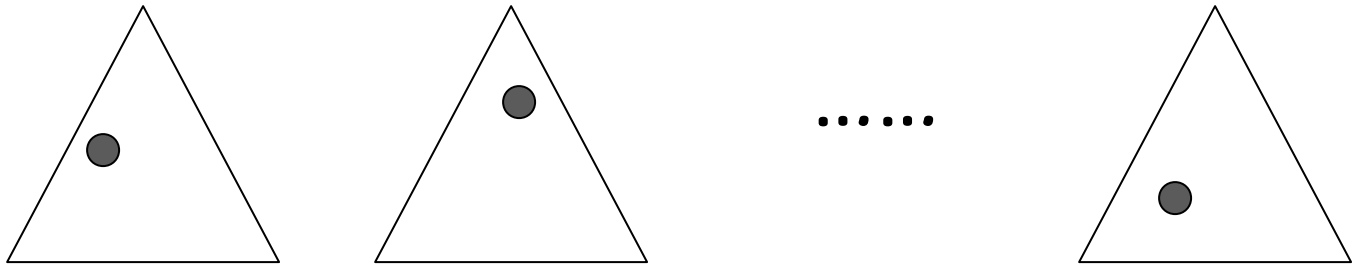
Phase 2:

In each subtree get the short nodes



Phase 2:

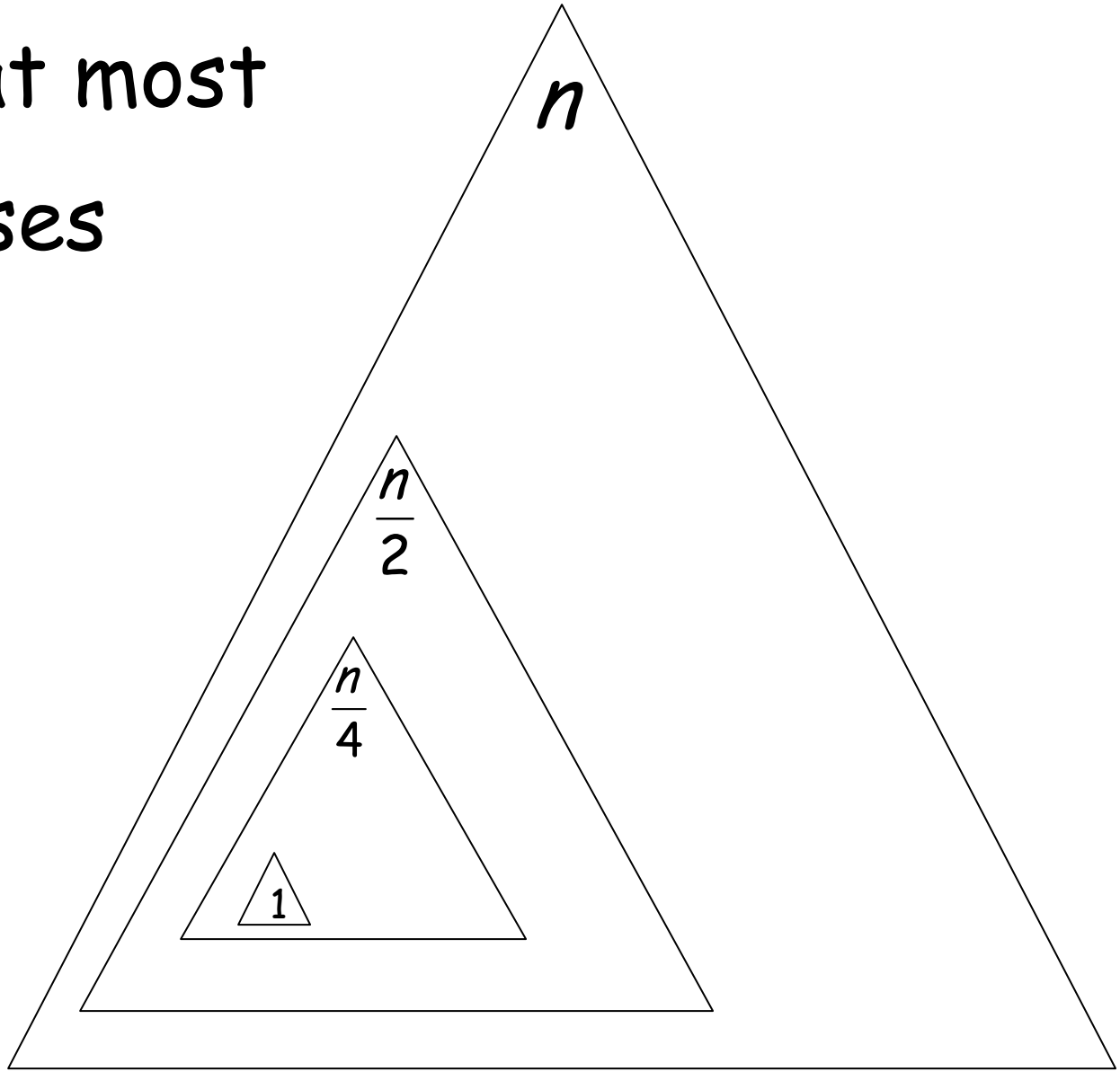
In each subtree get the short nodes



Phase 2:
Route packets that cross the short nodes

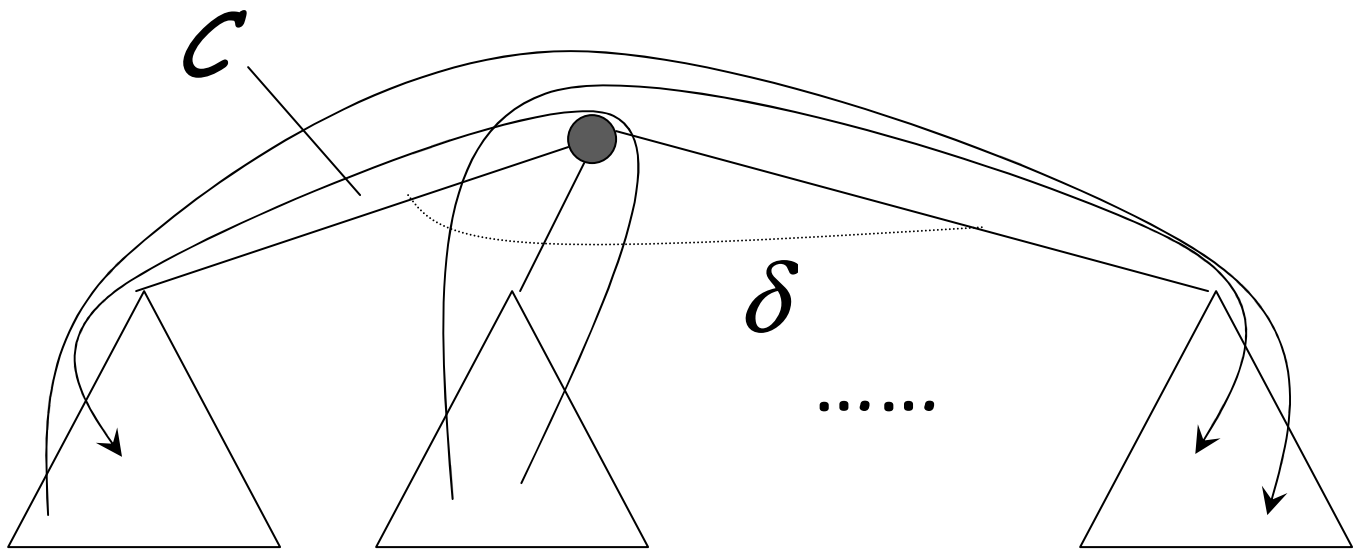
There are at most

$\log n$ phases



Phase k:

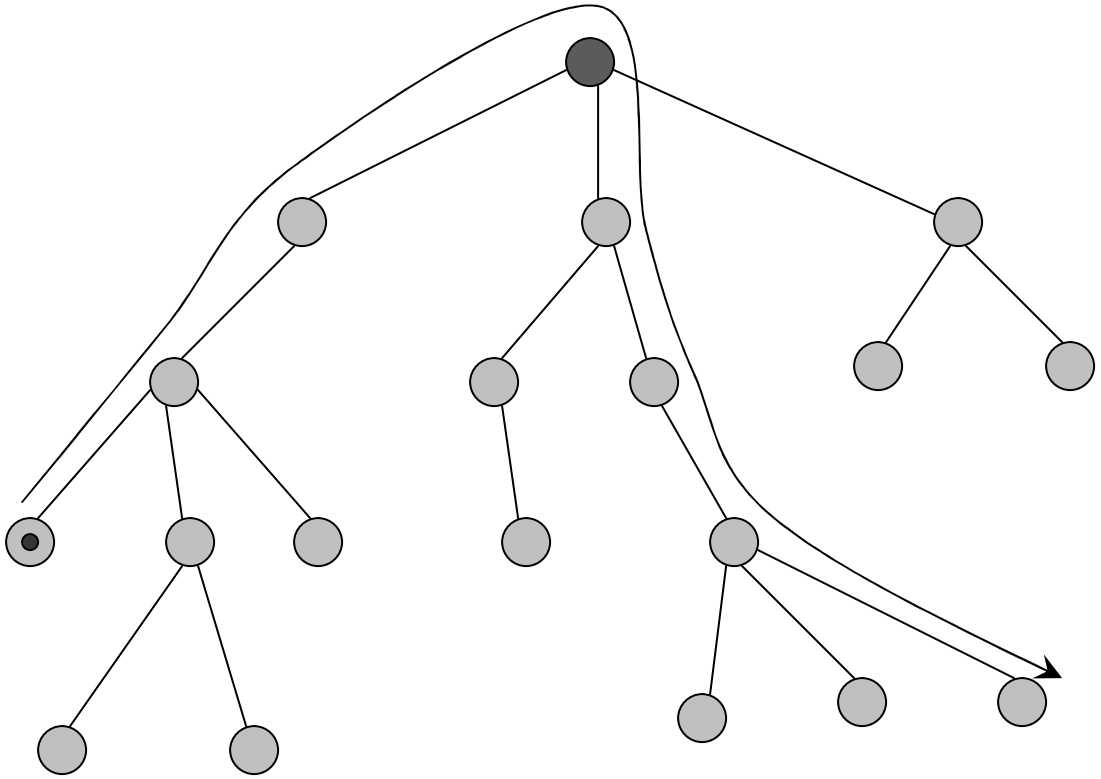
Route packets that cross the short node



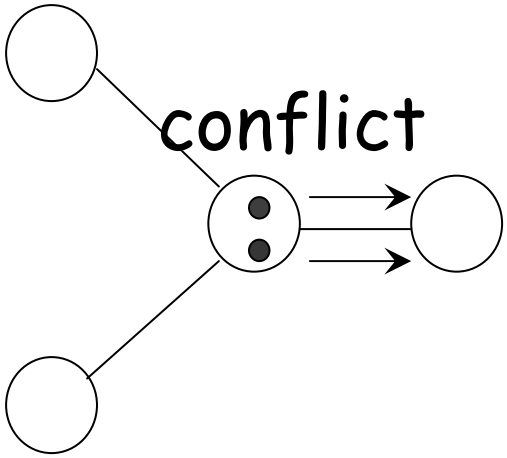
Bound on number of packets: δC

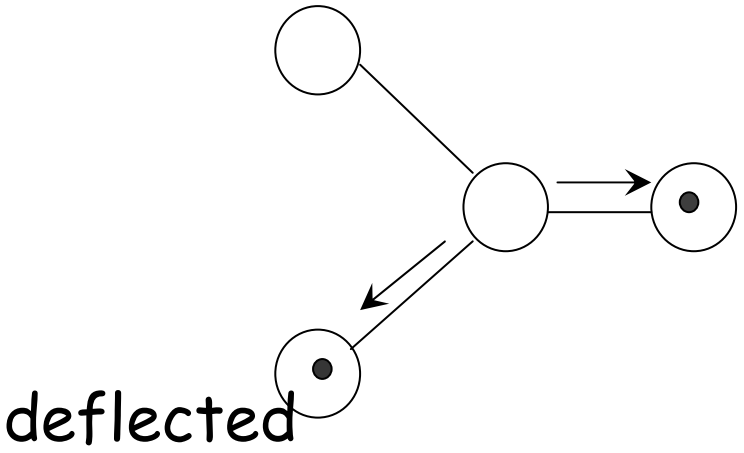
Phase k:

A packet follows its path greedily



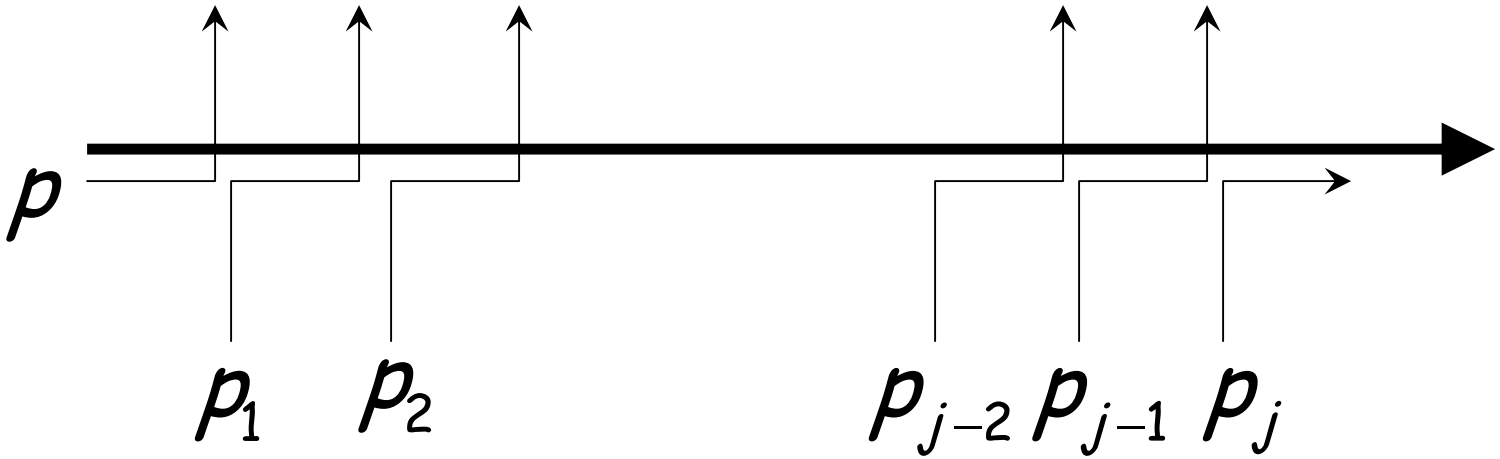
However, packets can conflict and get deflected





Deflection Sequence

[Borodin, Rabani, Schieber 1997]



If a packet p is deflected then some other packet p_j reaches its destination

Since there are at most δC packets, there are at most δC deflections

Worst Routing Time for a packet:

$$2\delta C + D$$

deflections

Initial distance

Total Routing Time

$$(2\delta C + D) \cdot \log n$$

Packet time
In a phase

Number
of Phases

Presentation Outiline

- Deterministic Algorithm
- Randomized Algorithm

Randomized Algorithm

Same with deterministic algorithm,
with only difference:

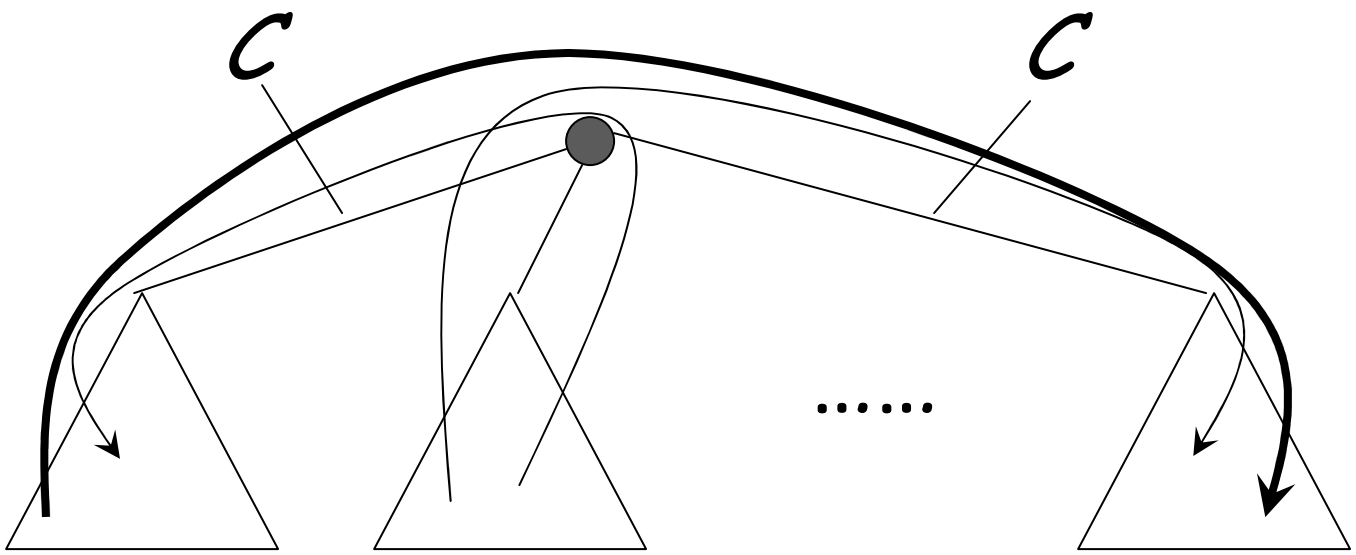
Packet conflicts are resolved
according to random packet priorities

Packet Priorities:

Low: each packet starts with a low priority

High: when a packet is deflected it increases its priority with probability $\frac{1}{4(C+D)}$

A high priority packet can conflict with at most $2C$ packets



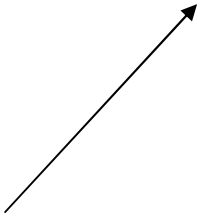
From those packets, $O(1)$ are expected to be in high priority

A high priority packet successfully reaches its destination with probability $\frac{1}{2}$

Thus $O(\log n)$ attempts to become high priority are enough.

Total Routing Time

$$\kappa(C + D)\log n \cdot \log n$$



Packet time
In a phase

Number
of Phases

Discussion

We presented two near-optimal hot-potato algorithms for trees

(within logarithmic factors from optimal)

Open problem:

Remove the logarithmic factors