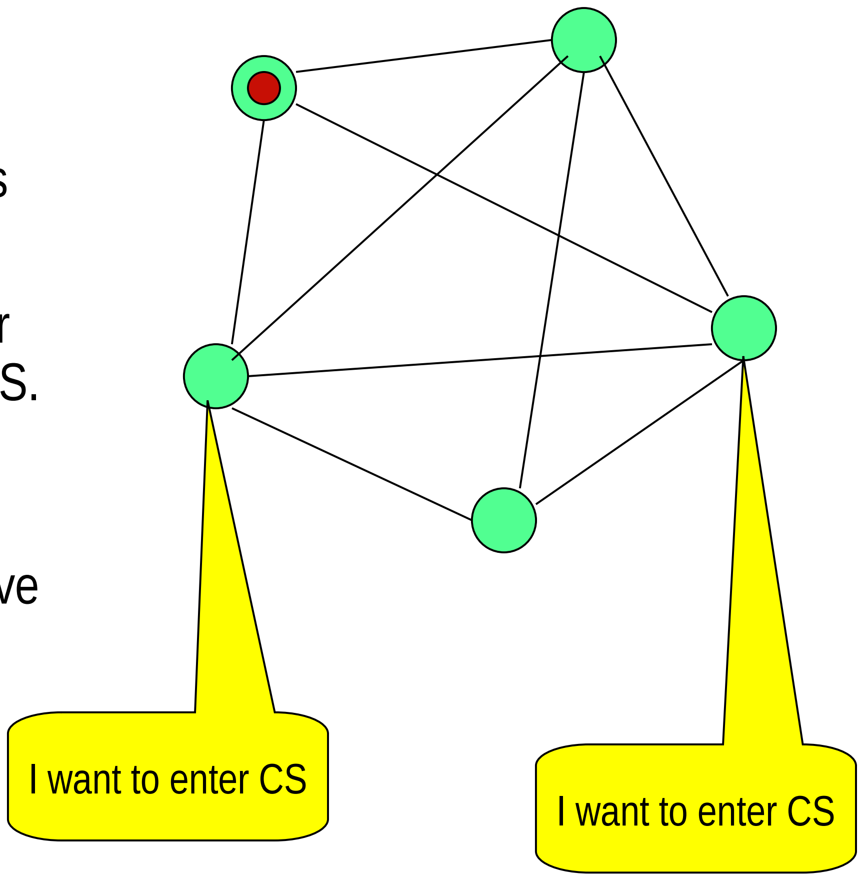
Token-passing Algorithms

Suzuki-Kasami algorithm
The Main idea

Completely connected network of processes

There is **one token** in the network. The holder of the token has the permission to enter CS.

Any other process trying to enter CS must acquire that token. Thus the token will move from one process to another based on demand.



Suzuki-Kasami Algorithm

Process i broadcasts **(i, num)**

Sequence number of the request

Each process maintains

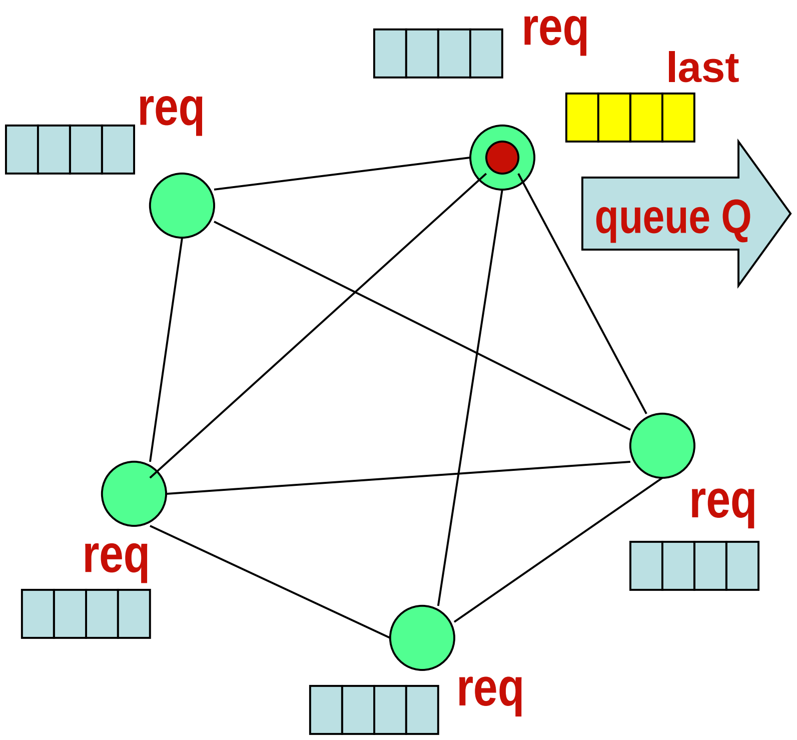
-an array **req**: **req[j]** denotes the sequence no of the ***latest request*** from process j

(Some requests will be stale soon)

Additionally, the holder of the token maintains

-an array **last: last[j]** denotes the sequence number of *the latest visit* to CS from for process j.

- **a queue Q** of waiting processes



req: array[0..n-1] of integer

last: array [0..n-1] of integer

Suzuki-Kasami Algorithm

When a process **i** receives a request **(k, num)** from process **k,** it sets **req[k] to max(req[k], num).**

The holder of the token

--Completes its CS

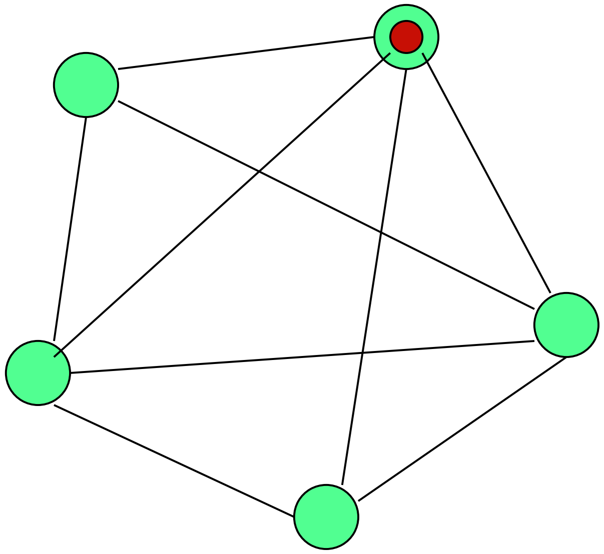
--Sets **last[i]:=** its own **num**

--Updates **Q** by retaining each process **k** only if **1+ last[k] = req[k]**

(*This guarantees the freshness of the request)*

--Sends the token to the ***head of Q***, along with the array **last** and the ***tail of Q***

In fact**, token ≡ (Q, last)**



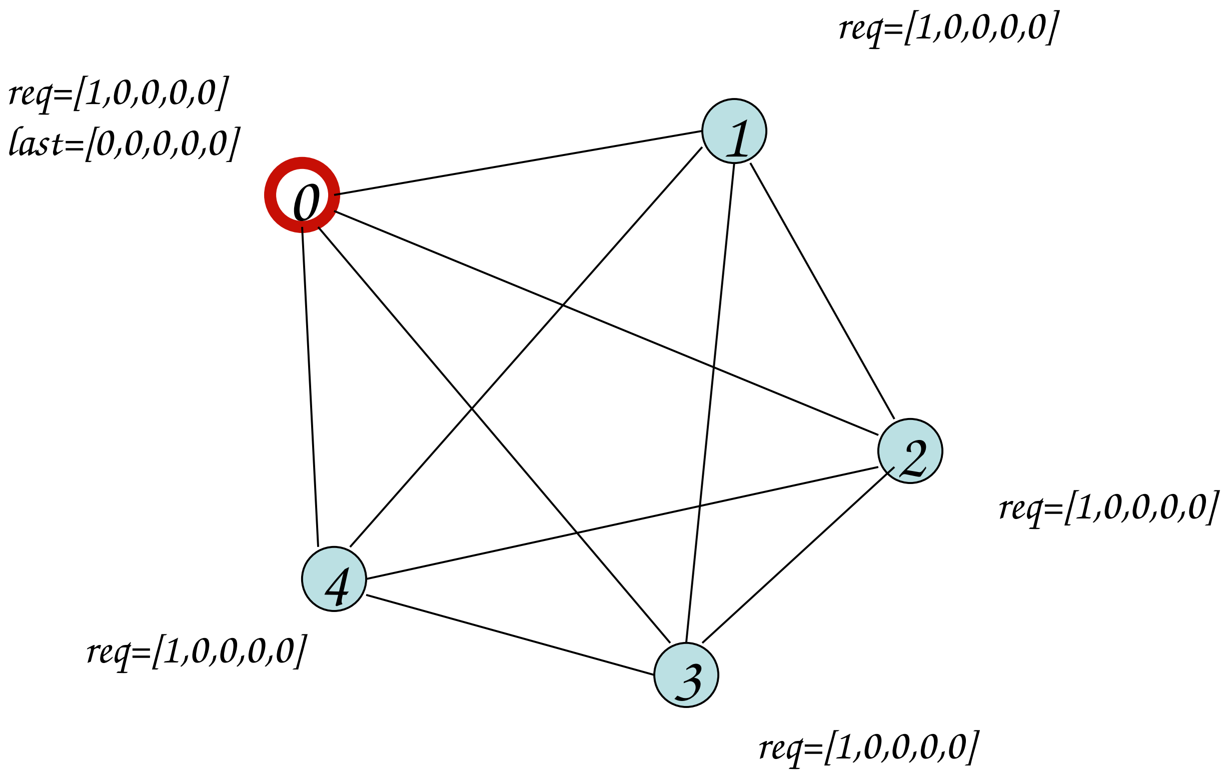
Req: array[0..n-1] of integer

Last: Array [0..n-1] of integer

Suzuki-Kasami's algorithm

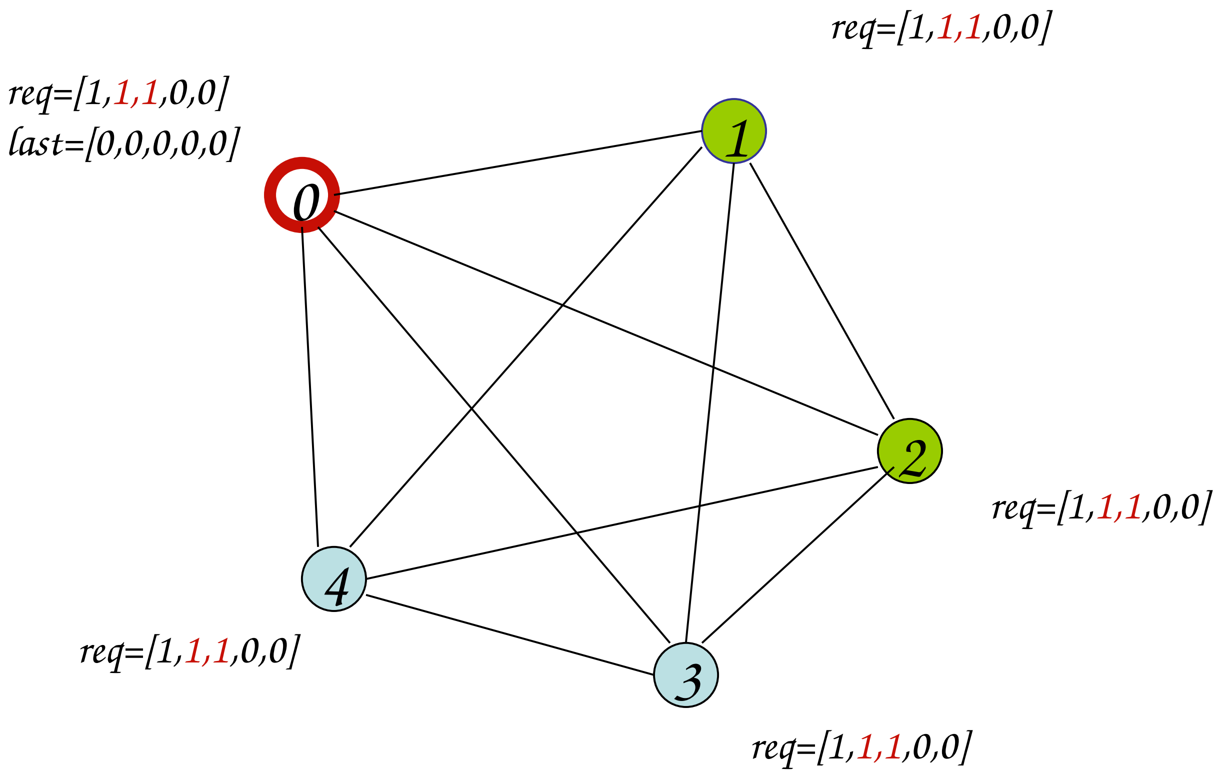
```
{Program of process j}
Initially, ∀i: req[i] = last[i] = 0
* Entry protocol *
        req[j] := req[j] + 1
        Send (j, req[j]) to all
        Wait until token (Q, last) arrives
        Critical Section
* Exit protocol *
        last[j] := req[j]
        ∀k ≠ j: k ∉ Q ^ req[k] = last[k] + 1 → append k to Q;
        if Q is not empty → send (tail-of-Q, last) to head-of-Q fi

* Upon receiving a request (k, num) *
        req[k] := max(req[k], num)
```

Example



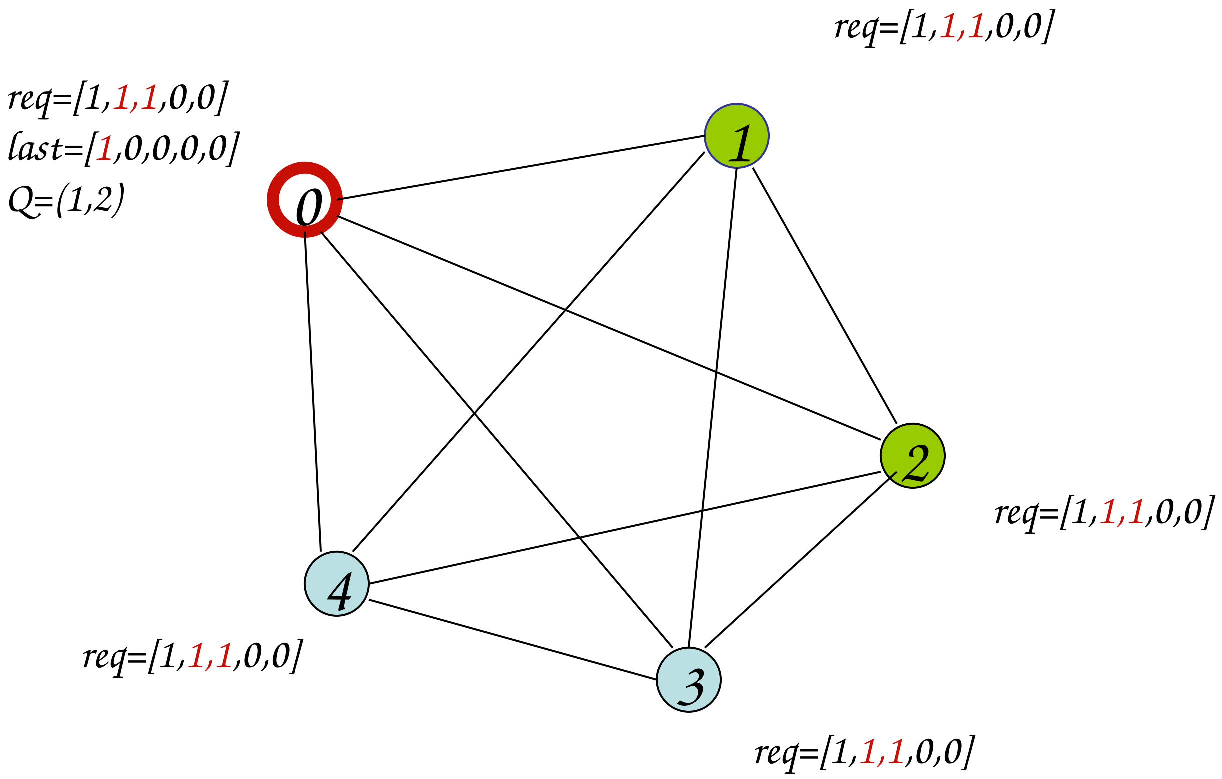
initial state

Example



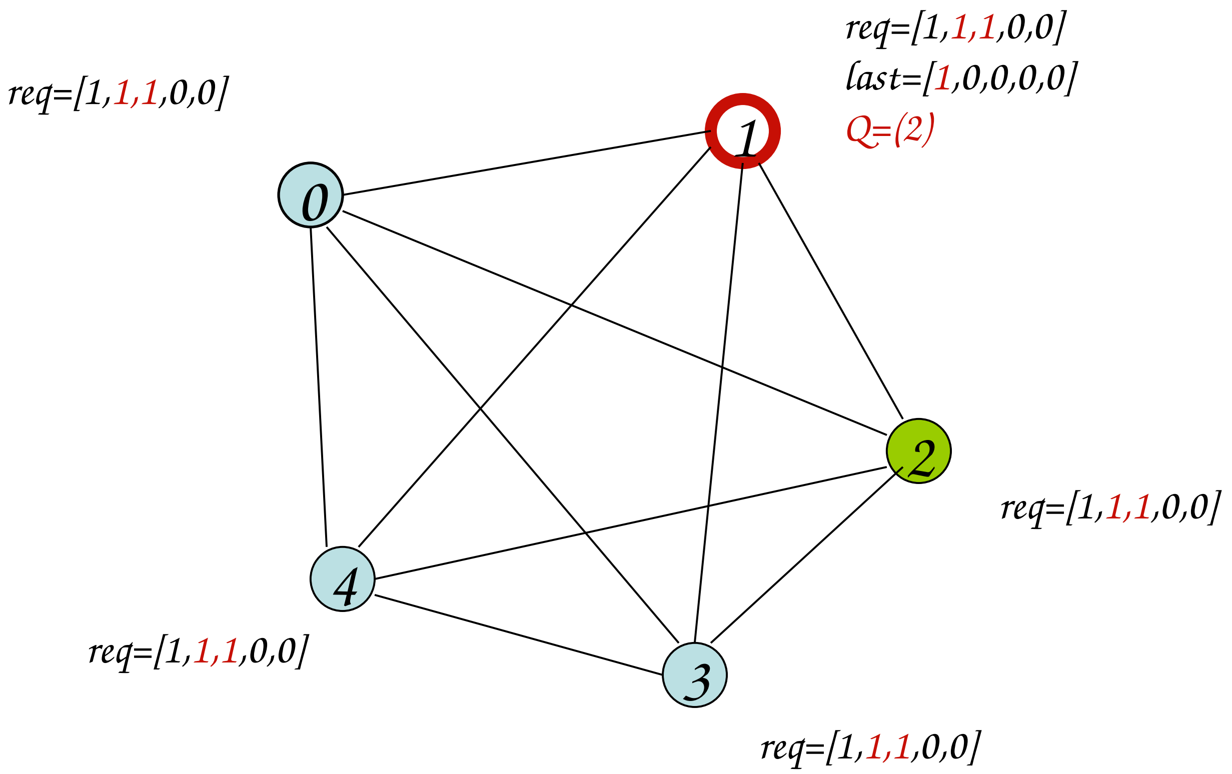
1 & 2 send requests

Example



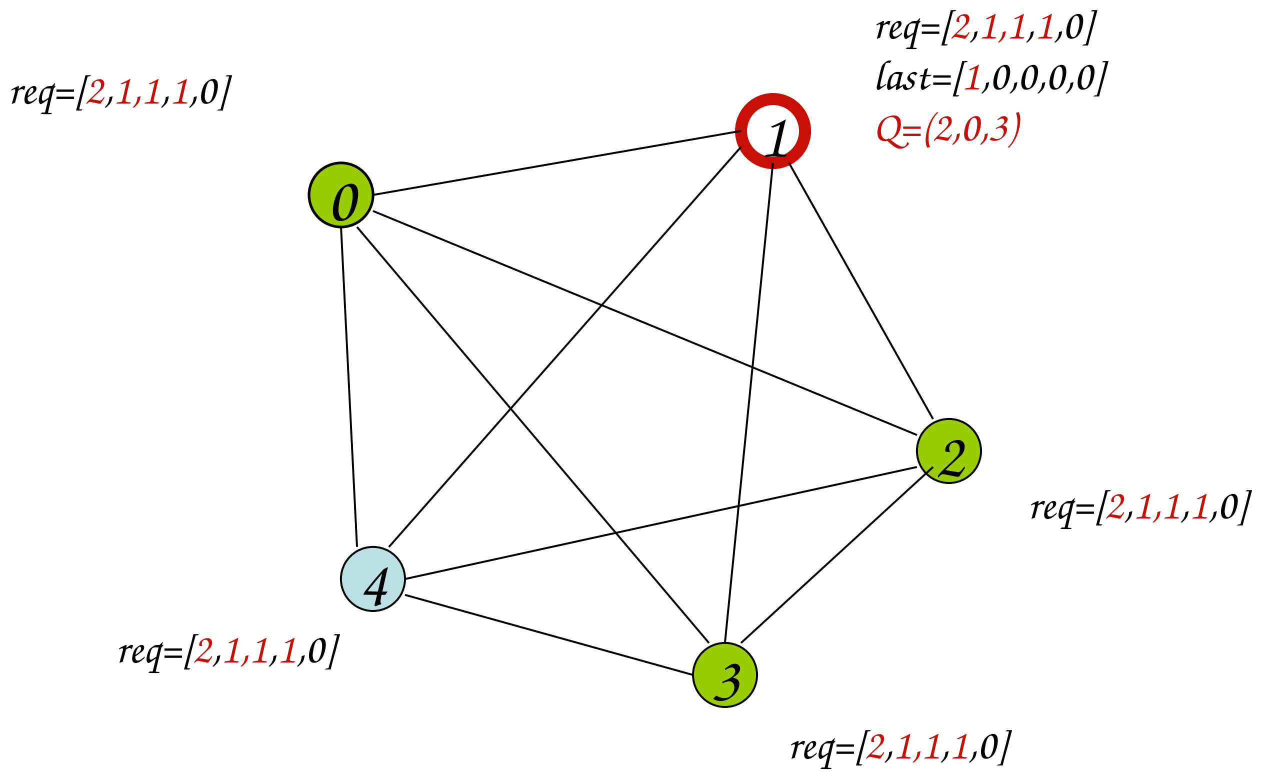
0 prepares to exit CS

Example



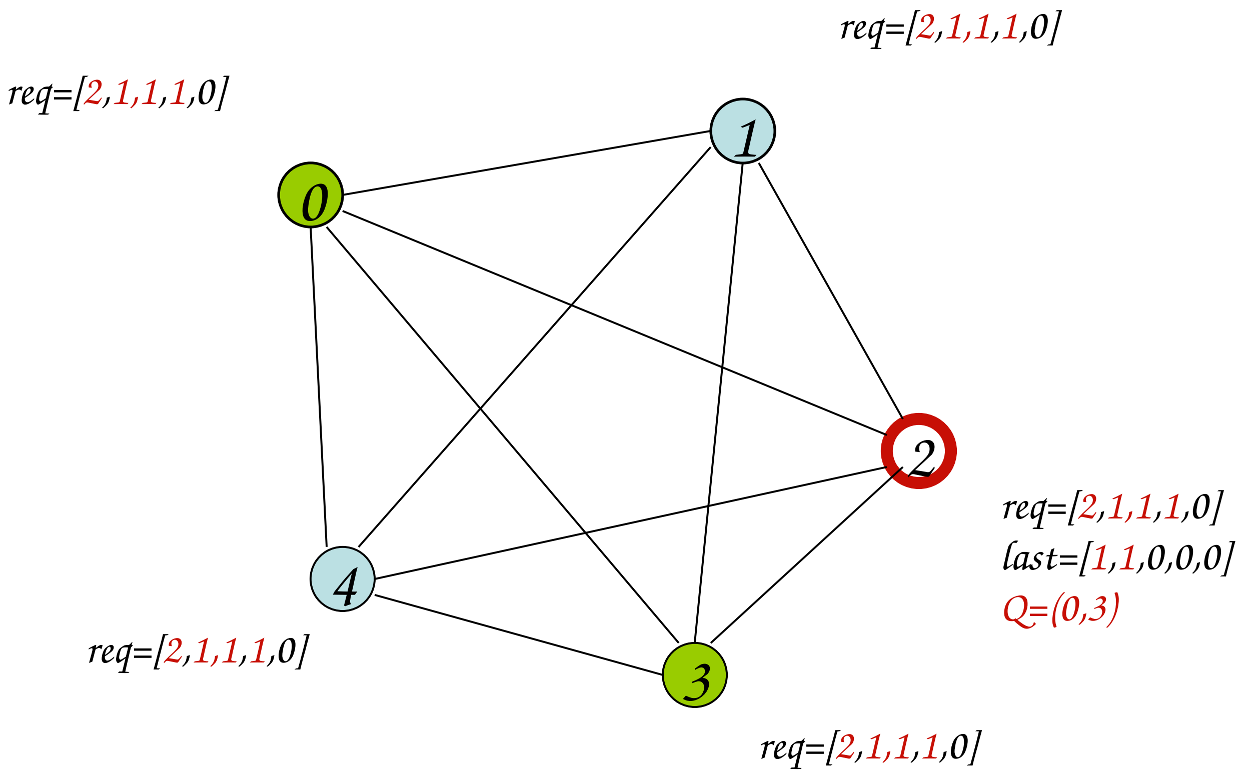
0 passes token (Q and last) to 1

Example



0 and 3 send requests

Example



1 sends token to 2