CSE 531: Distributed and Multiprocessor Operating Systems

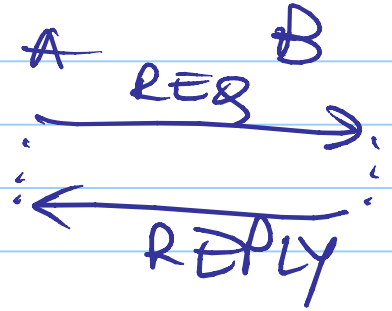
Class: 22 Date: 11/7

Fall 2017

Note Title

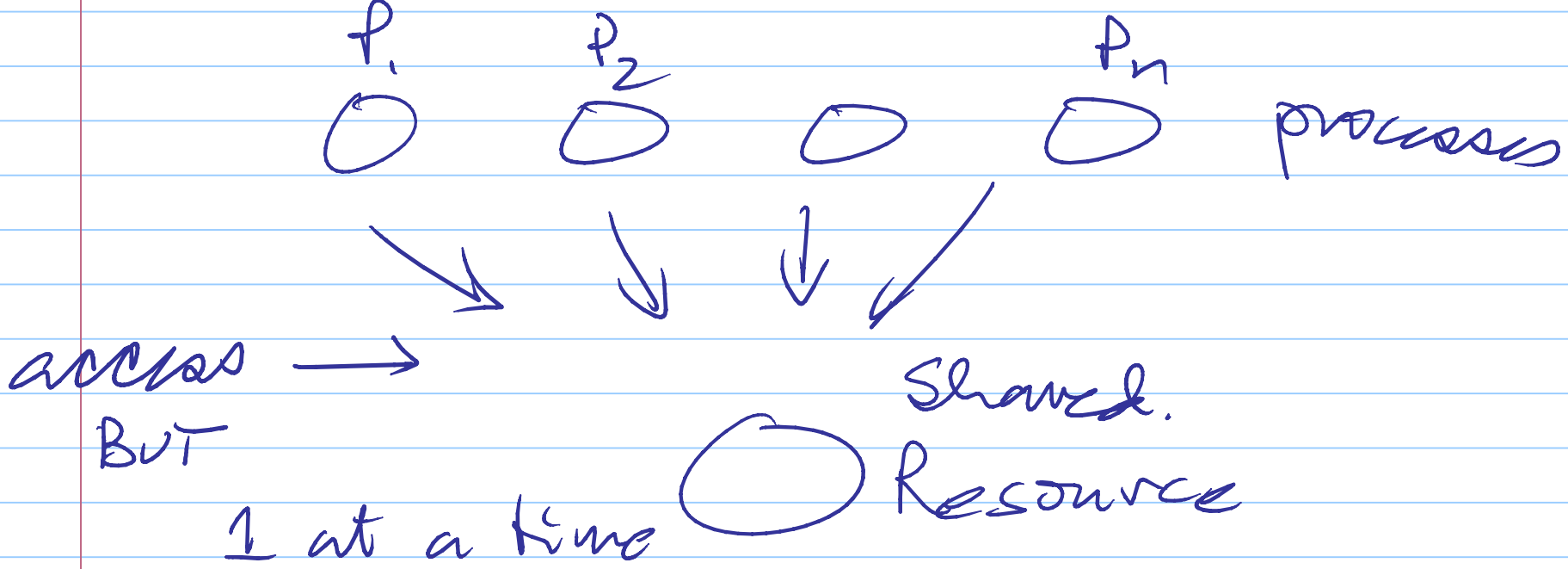
Message passing

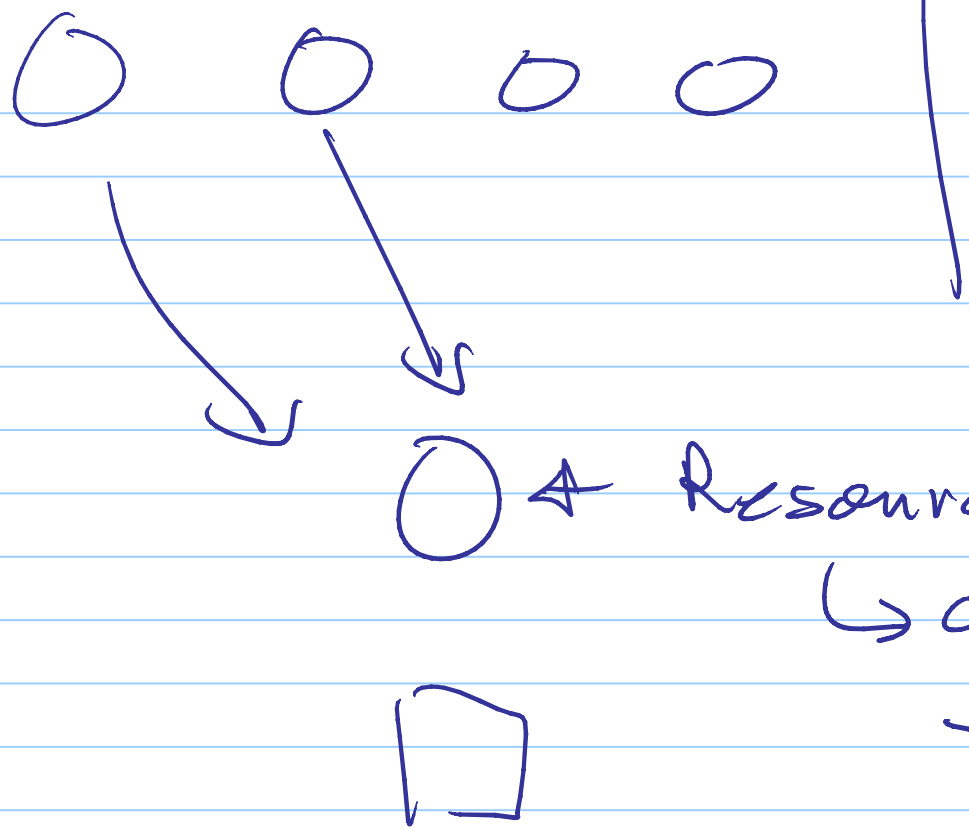
→ Transactions



→ Connectionless

Distributed Mutual Exclusion.



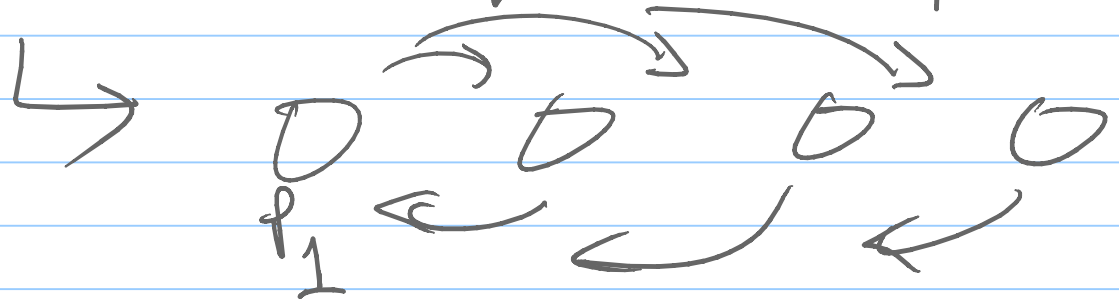


Centralized Solution to the dist mutex problem

Resource Mngr
↳ controls access
→ decides who gets access

Distributed solution to Distributed mutual Exclusion

- 1st - Lamport 1974



Distributed Locking

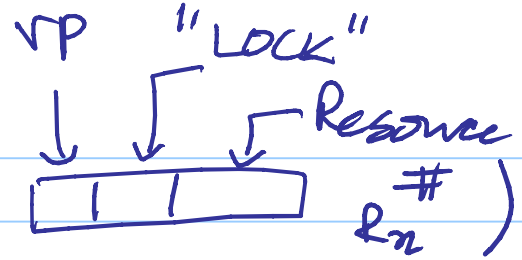
Resource mngr → Lock Server

↳ has n resources

↳ so n locks

L#, → 0, 1, ...(n-1)

API client



→ lock(Rx) { send(Lsp,)

rcv(rp, msg)

↳ OK

↳ ~~not OK~~

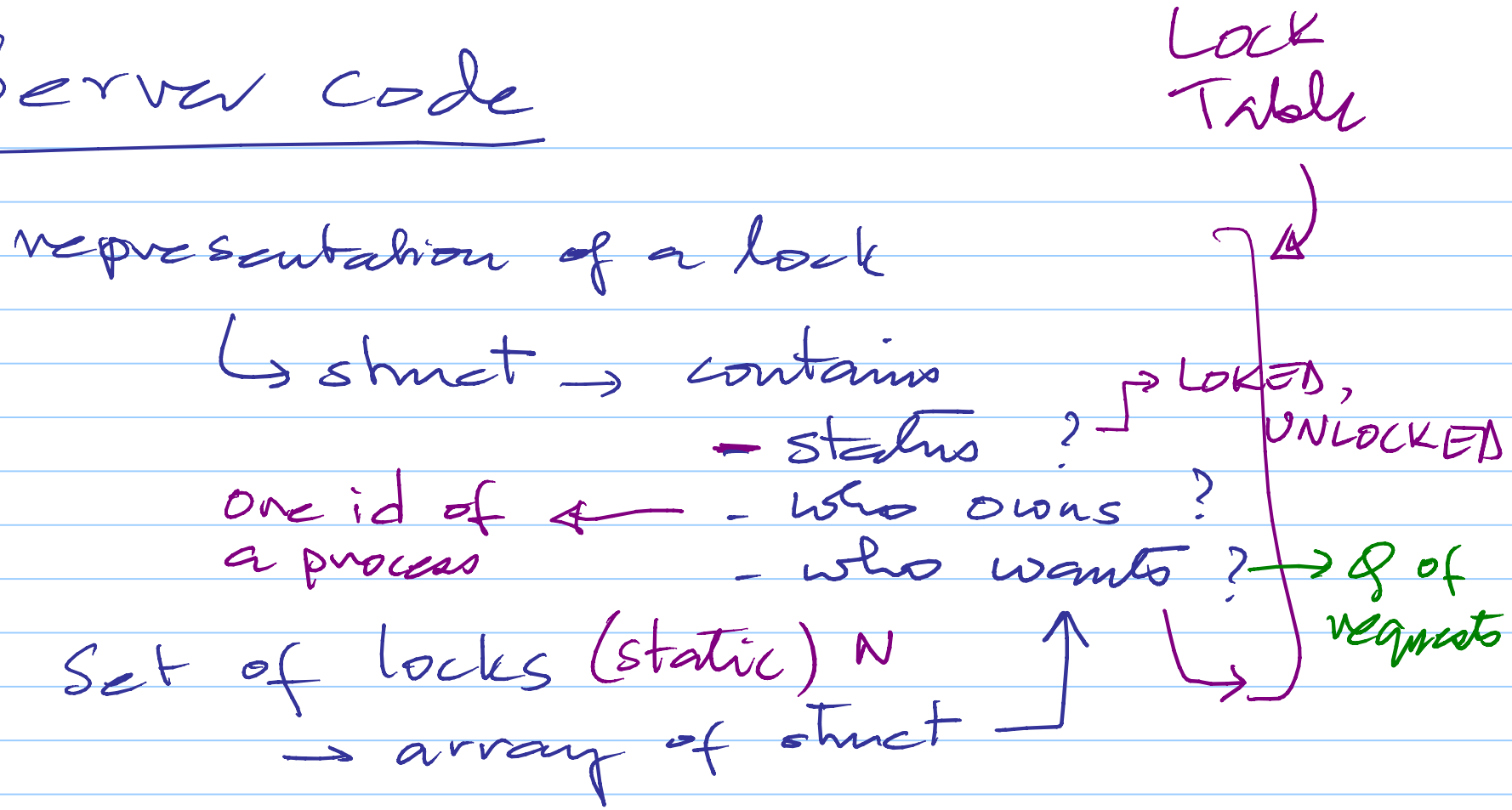
↳ error

→ unlock(Rx) {

need it! send(Lsp, [rp, UNLOK, R#])

rules → rcv(- - - .)

Server code


Lock Table
representation of a lock
↳ struct → contains
- status ? → LOKED, UNLOCKED
One id of a process ← - who owns ?
- who wants ? → # of requests
Set of locks (static) N
→ array of struct

Lockstruct

{ int status

int owner

&type &; }

lock Server

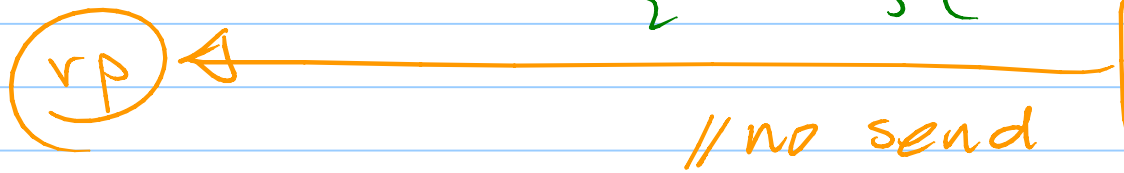
```
main ()
  { p = createport ()
    . register p as "lockserver" with nameserver
    . ∞ loop
        { rcv (p, request)
          ≡ ?
        }
```

```
rcv(p, req)

    req → rp, op, r#
        if (op == LOCK)
            Server_lock(rp, r#)

        else
            Server unlock(rp, r#)
```

```
Servelock (rp, r#)
  { if (Locktable[r#].status == unlocked)
        { change status to locked;
          change owner to rp;
          send (rp, OKmsg);
        }
    else { AddQ (Locktable[R#].Q ,
                 item )
                                    }
```



// no send

client behaviour

→ need to check?

① unlock a resource without locking

② locking locked resource

what is OK

option 1 → • no unlocking without locking
• locking again not OK

---

option 2 → unlock == no op
2nd lock == no op

---

option 3 → nested locking OK
locking N times needs
N unlocks to unlock

⋮

Assume well behaved clients

↳ no unlock, no lock twice

↳ how ① → server check & error return

② → client check

↳ lock() { check local lock table for existing lock → error

unlock (rp, r#)

{ check Q for waiters — YES

→ DelQ get one waiter rp (wrp) from Q

→ Send (wrp, OK)

Send (rp, OK)

change owner

NO → else change status → unlocked

Send (rp, OK) }

Deadlocks possible ?

— YES → every time

→ when a lock req happens

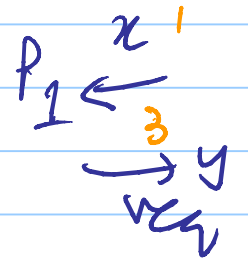
↳ check for deadlock

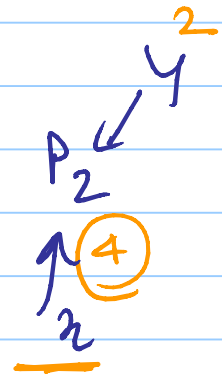
SERVER →

↓ maintain WFG

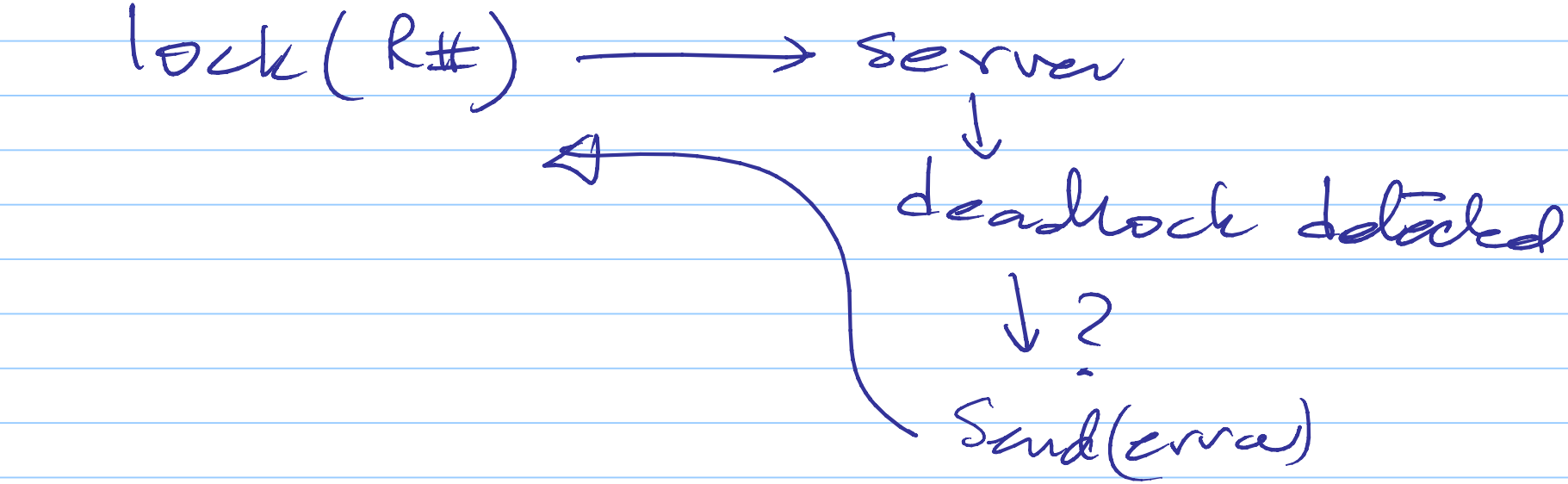
add edge

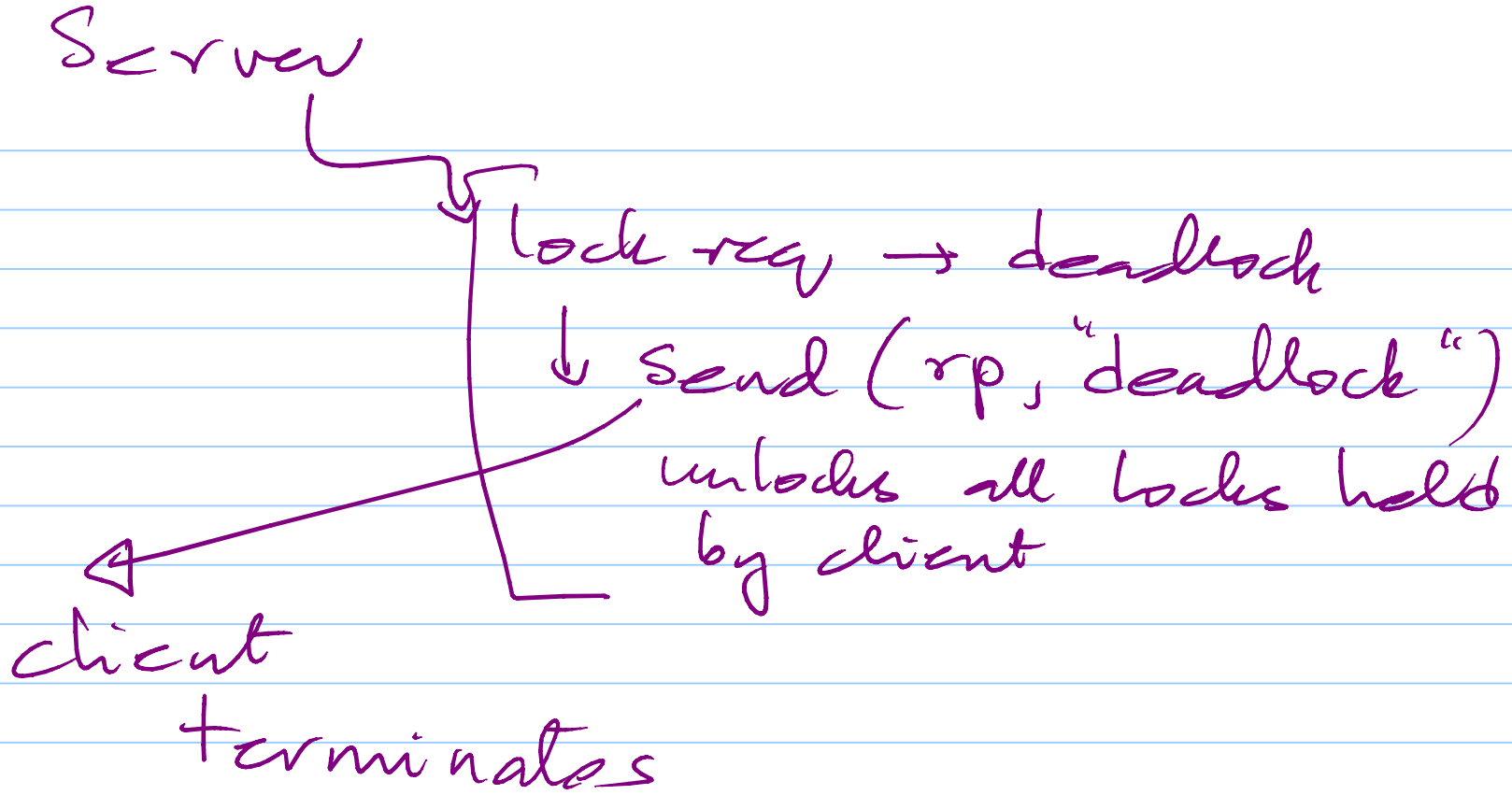
cycle detect





Deadlock resolution




Server
lock req → deadlock
send (rp, "deadlock")
unlocks all locks held by client
client terminates

- Shared & exclusive locks

resource ⟶ read locks → multiple
write locks → single

readlock → yes is no w waiting

Can a process holding a R lock, ask for a W-lock?

└YES ?