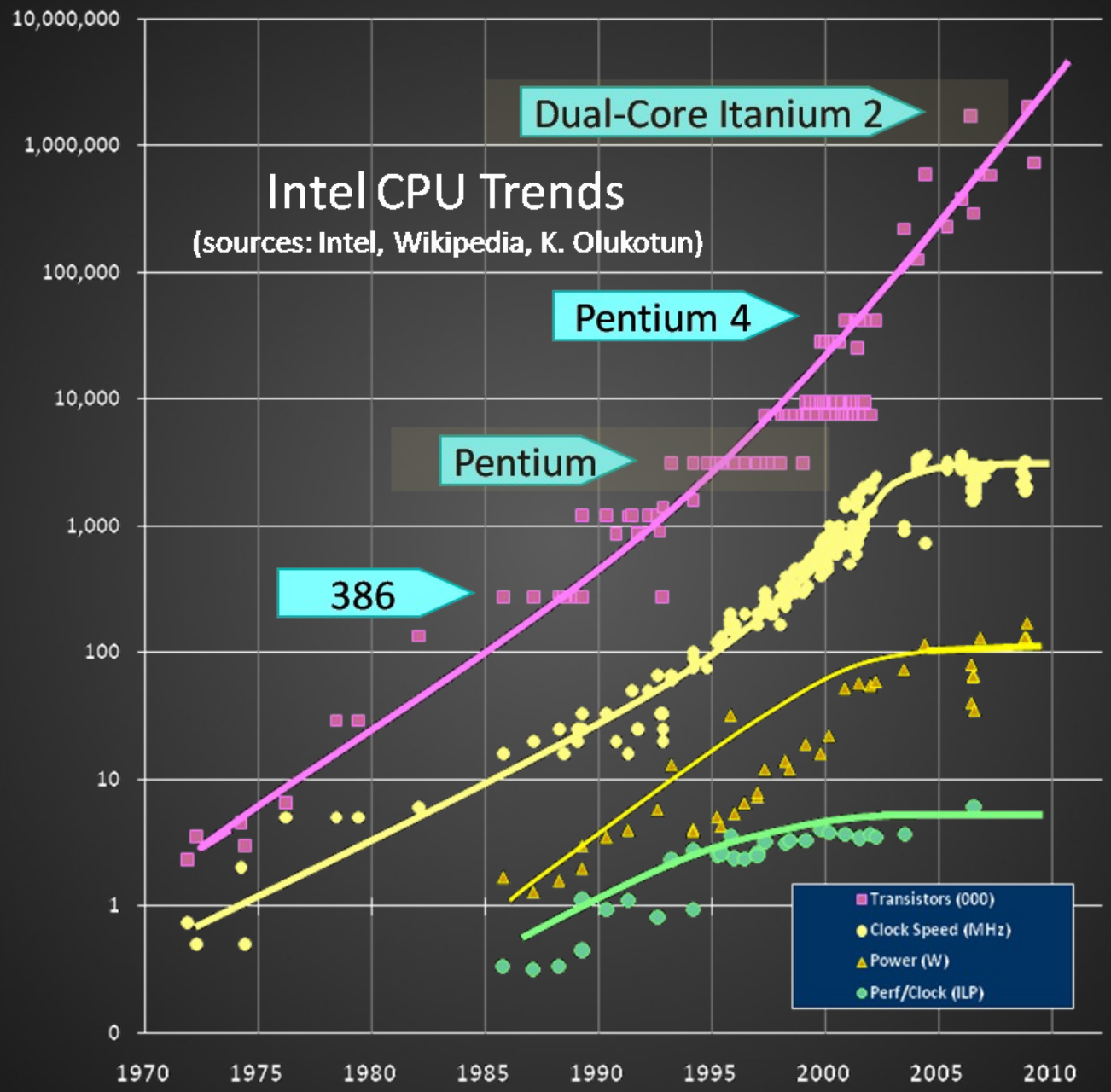
Διπλωματική Εργασία

Τροποποίηση της υλοποίησης διαδικτυακών πρωτοκόλλων λειτουργικού συστήματος για την εκμετάλλευση πολλαπλών επεξεργαστικών μονάδων


Intel CPU Trends
(sources: Intel, Wikipedia, K. Olukotun)
Dual-Core Itanium 2
Pentium 4
Pentium
386
10,000,000
1,000,000
100,000
10,000
1,000
100
10
1
0
1970
1975
1980
1985
1990
1995
2000
2005
2010
Transistors (000)
Clock Speed (MHz)
Power (W)
Perf/Clock (ILP)

Το πρώτο βήμα: η BGL

- Ένα μόνο νήμα στον πυρήνα κάθε φορά

Το πρώτο βήμα: η BGL

- Ένα μόνο νήμα στον πυρήνα κάθε φορά
- Κλιμάκωση μόνο για αμιγώς υπολογιστικές εφαρμογές

Το επόμενο βήμα: περισσότερα κλειδώματα

- 1Z Εντοπισμός συχνά διεκδικούμενου κλειδώματος
- 2. Αντικατάσταση με περισσότερα κλειδώματα

Προβλήματα

- LOR
- Κλιμάκωση προς τα κάτω
- Bus contention

Οργάνωση της στοίβας πρωτοκόλλων στο DragonFlyBSD

Η πρώτη ύλη

Η στοίβα πρωτοκόλλων στο BSD

- Αρκετά γρήγορο
- Αξιόπιστο
- Τεκμηριωμένο
- 20+ χρόνια ανάπτυξης (το τελευταίο δεν είναι απαραίτητα καλό)

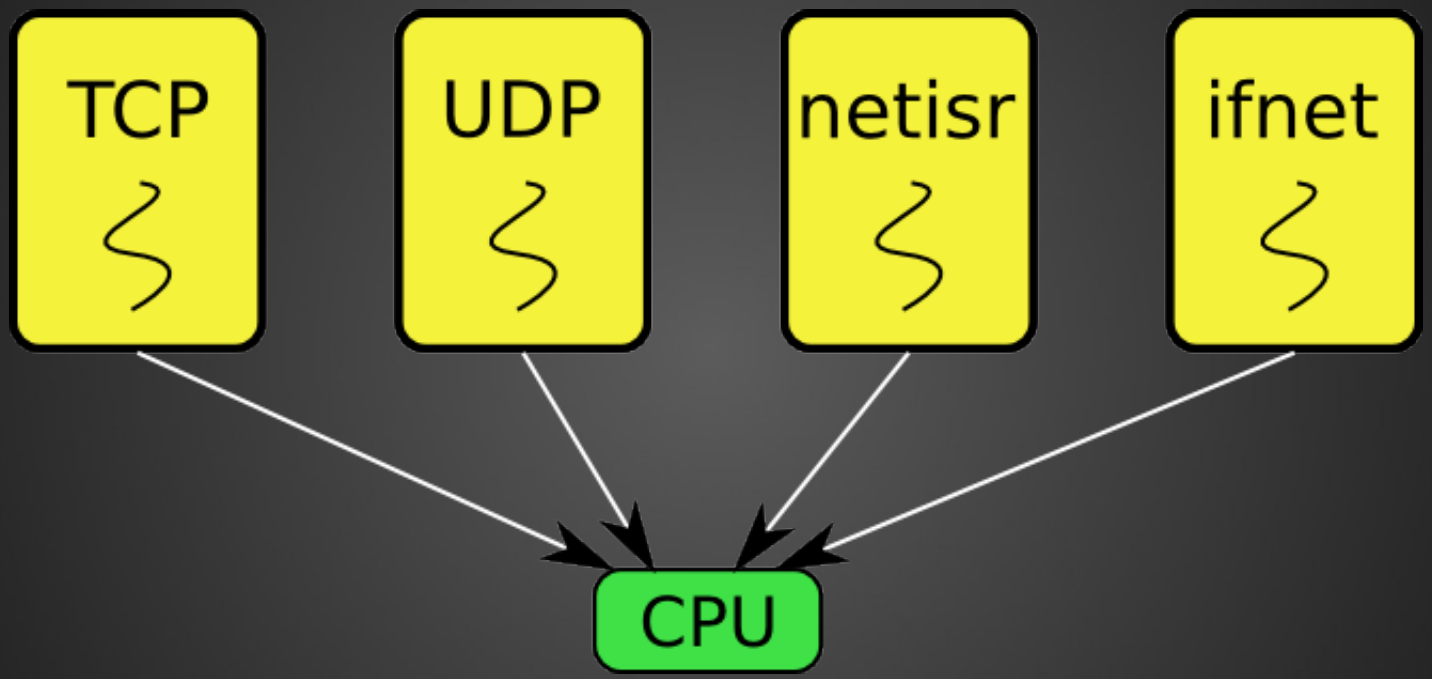
tcp_input()

- 4.4BSD (Net/2 Release): 1100 SLOC
- DragonFly 1.0: 1900+ SLOC
- 2η σε κυκλωματική πολυπλοκότητα σε όλο τον πυρήνα
- **Πολύ** εύθραυστη

Οργάνωση της στοίβας πρωτοκόλλων στο DragonFlyBSD

Αλλαγές στο DragonFly

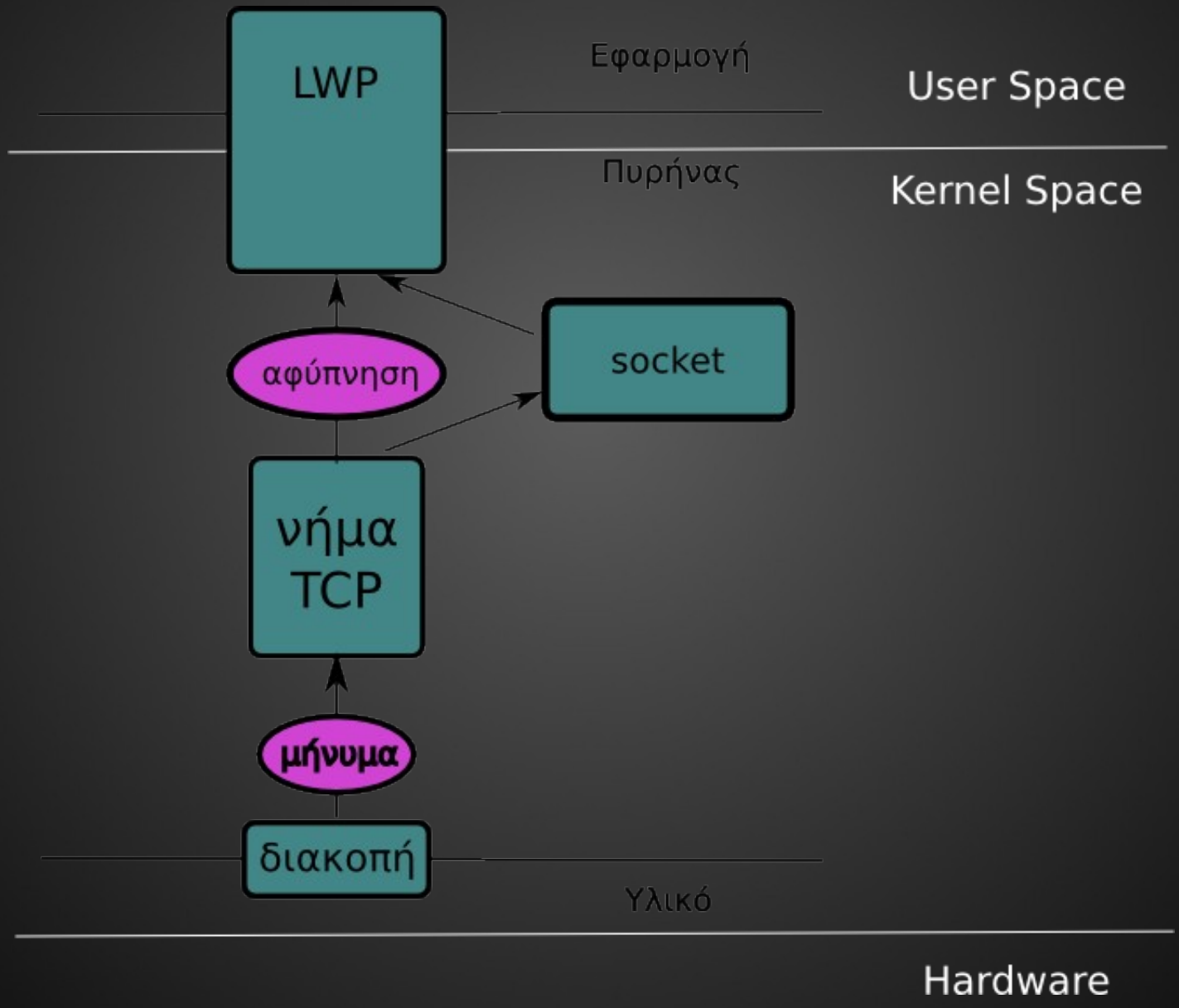
Τα νήματα



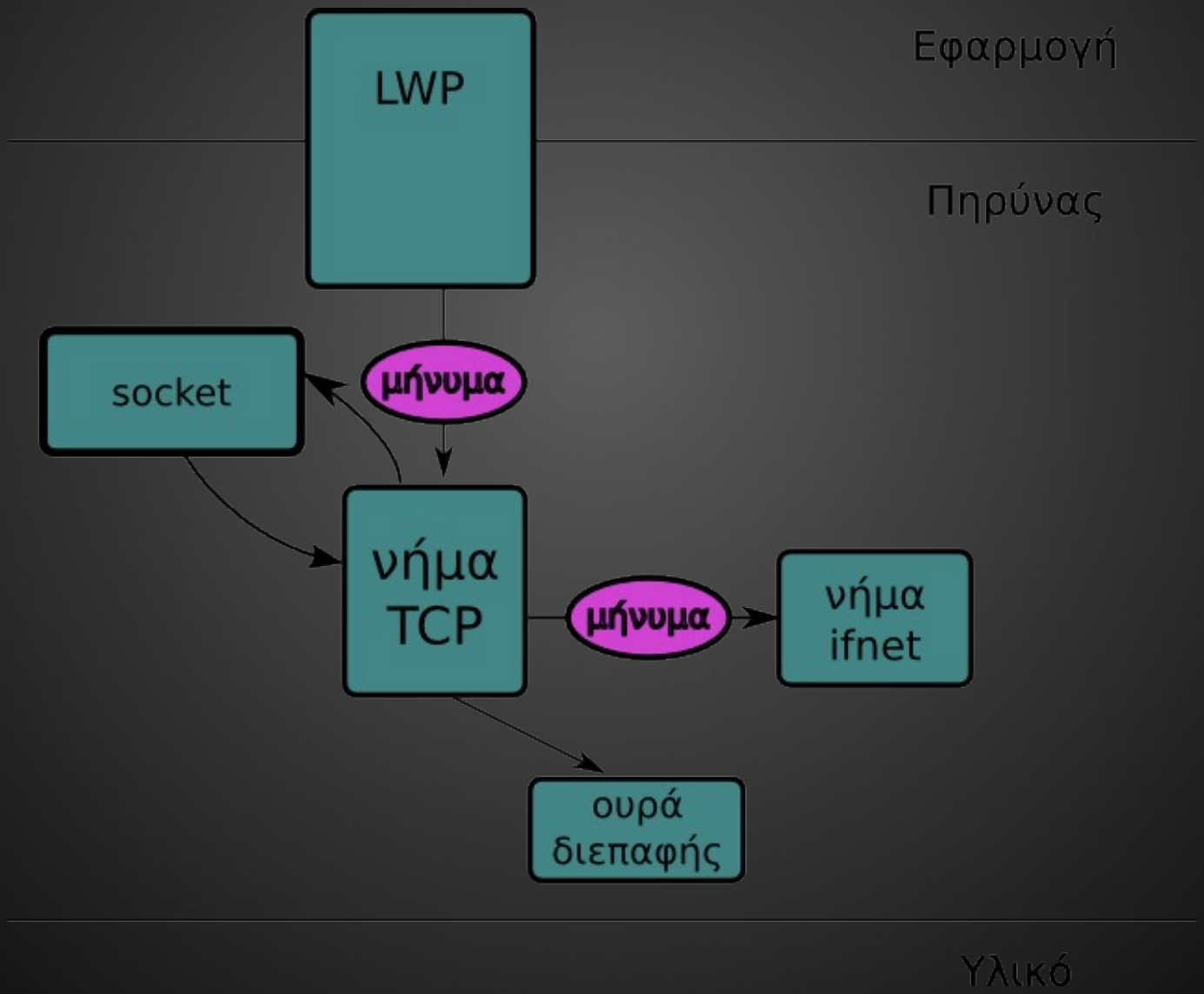
Επικοινωνία με τα νήματα πρωτοκόλλου

- abort
- accept
- attach
- bind
- connect
- connect2
- control
- detach
- disconnect
- listen
- peeraddr
- recvd
- recvoob
- send
- κλπ

Λήψη δεδομένων



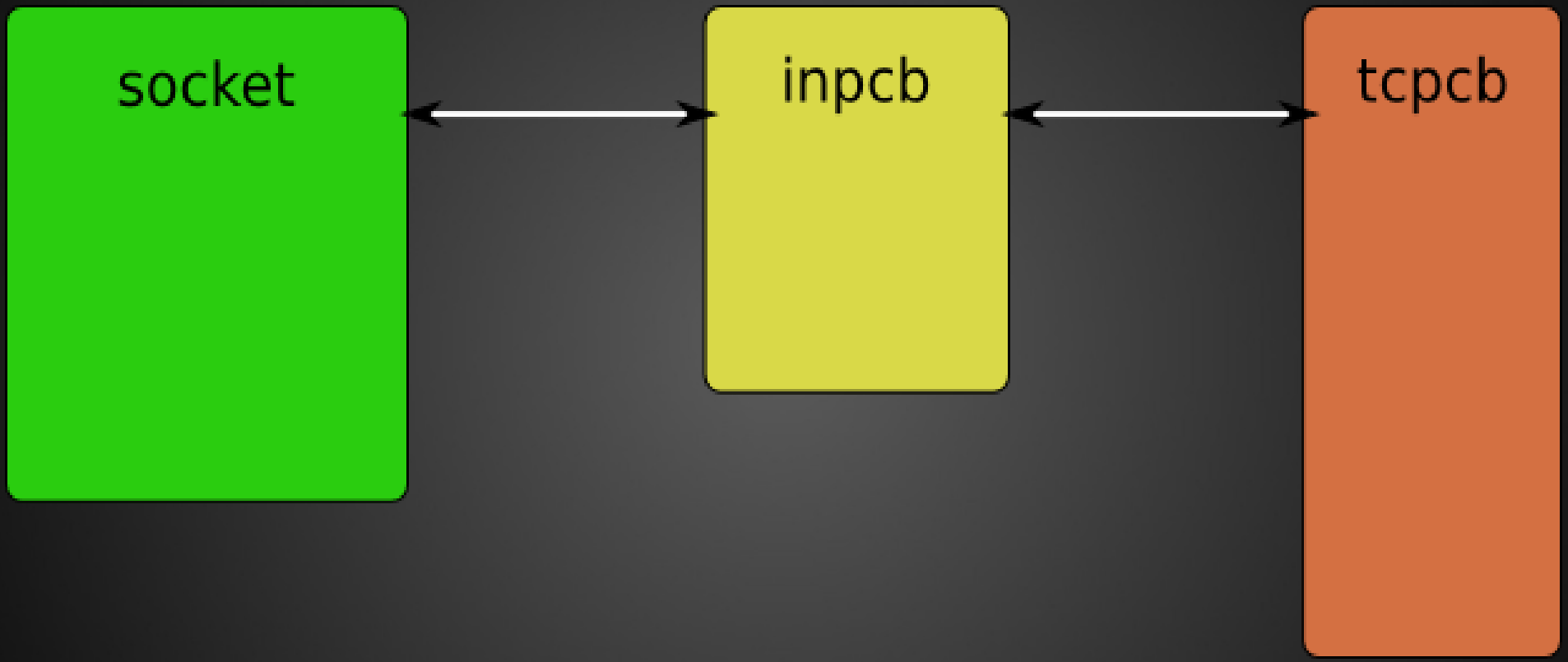
Αποστολή δεδομένων



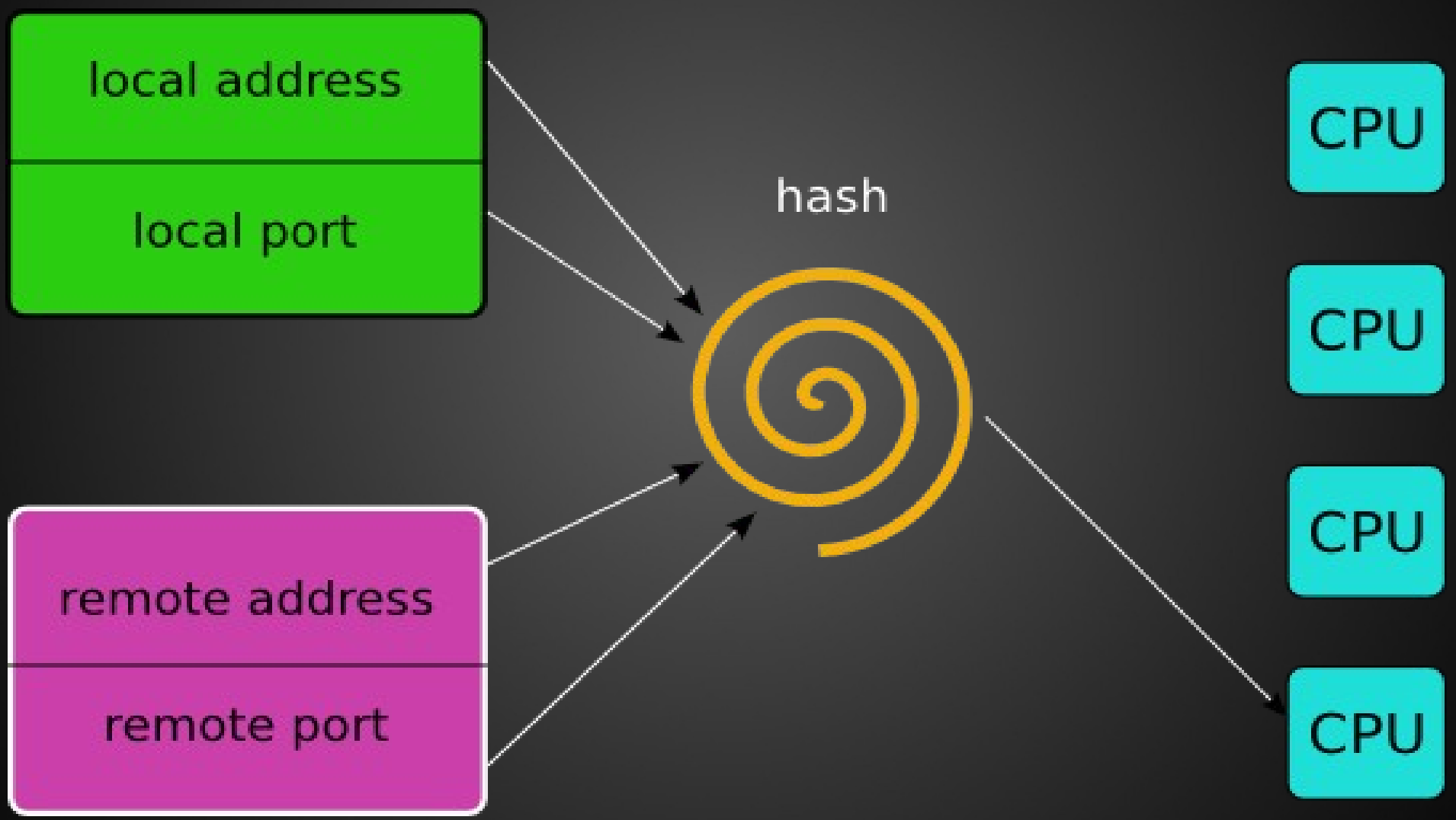
Ζητήματα (1)

Το PCB

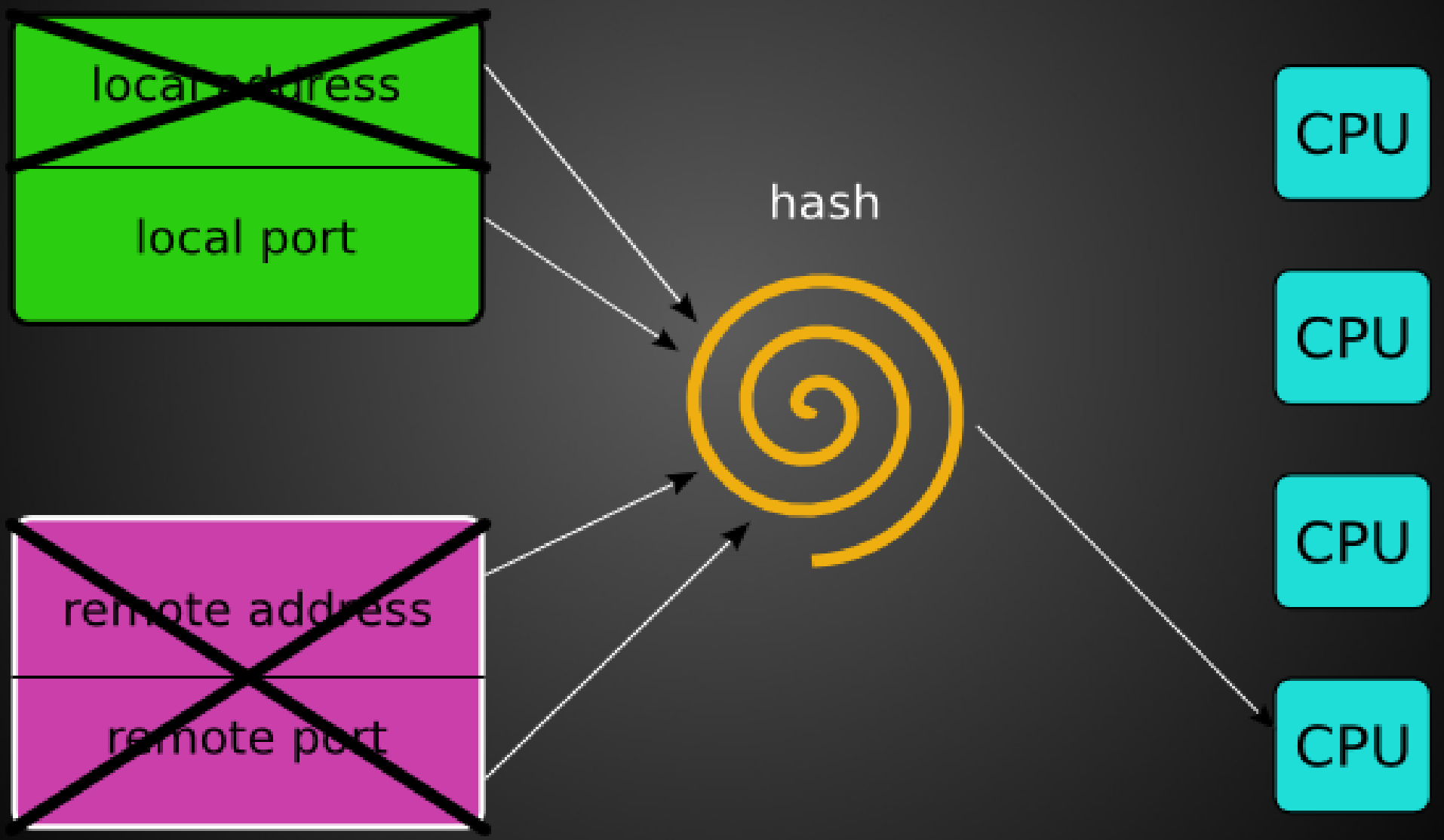
Protocol Control Block



TCP hash


local address
local port
remote address
remote port
hash
CPU
CPU
CPU
CPU

UDP hash


local address
local port
remote address
remote port
hash
CPU
CPU
CPU
CPU

Λύσεις για το UDP

- Χρήση μόνο της τοπικής πόρτας
- Migration + tombstones
- Inpcb replication
  - NxN?

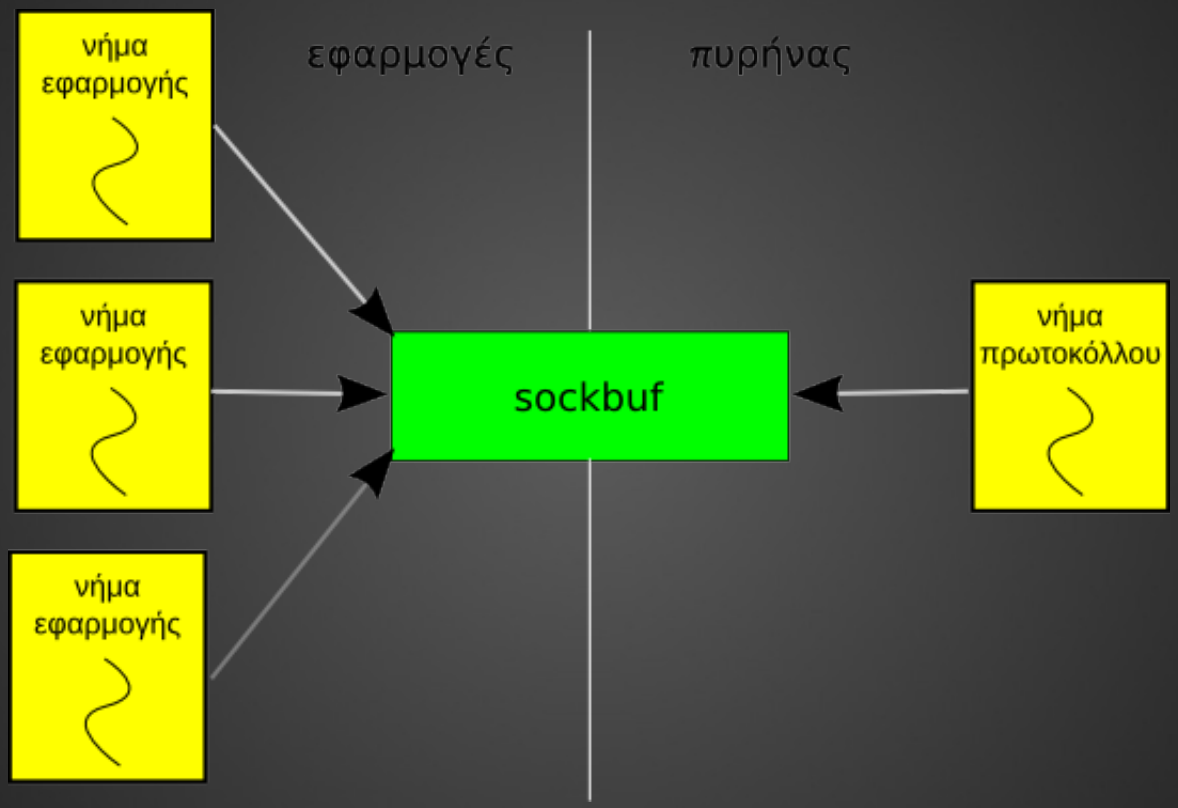
Ζητήματα (2)

Sockbuf

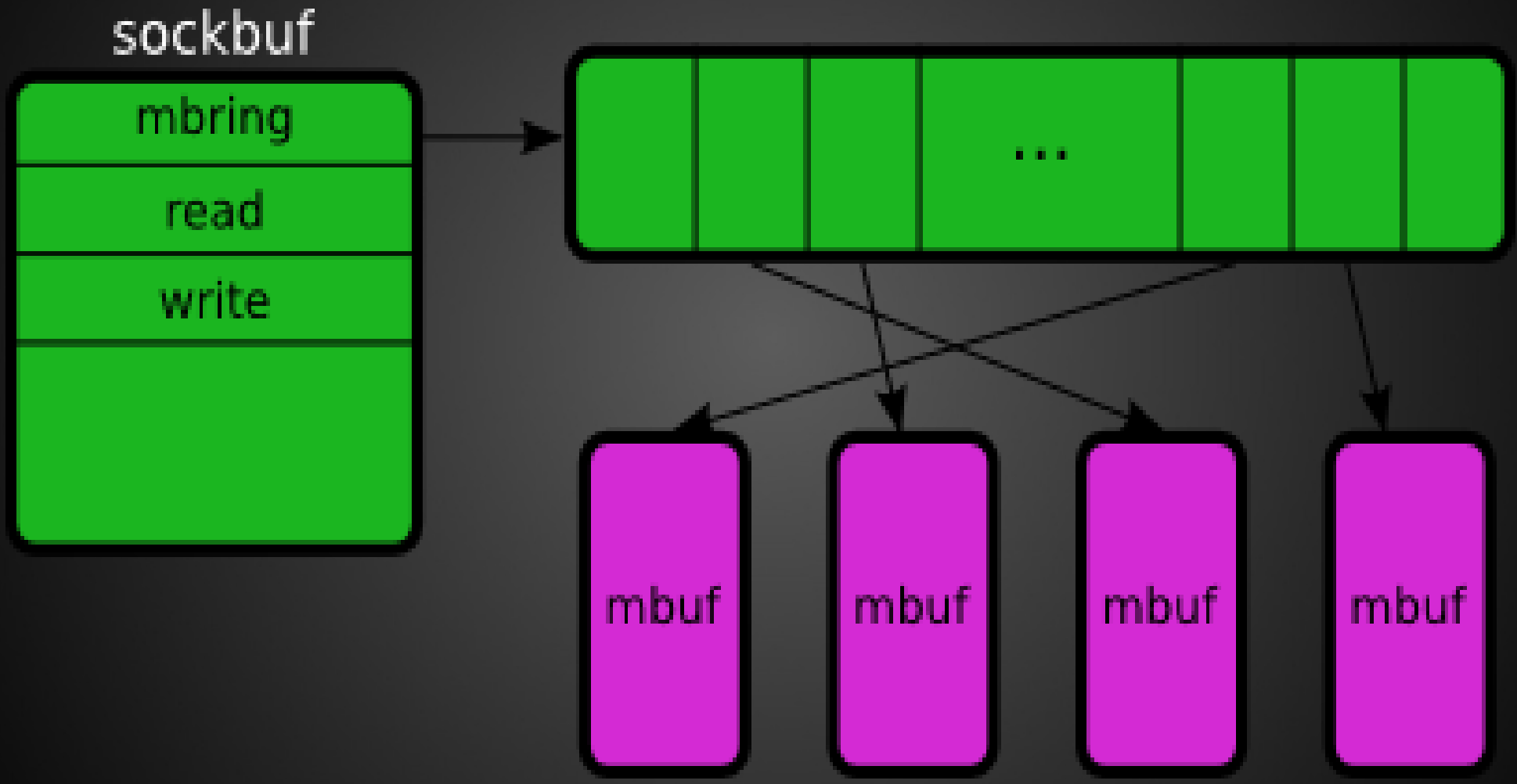
Sockbuf

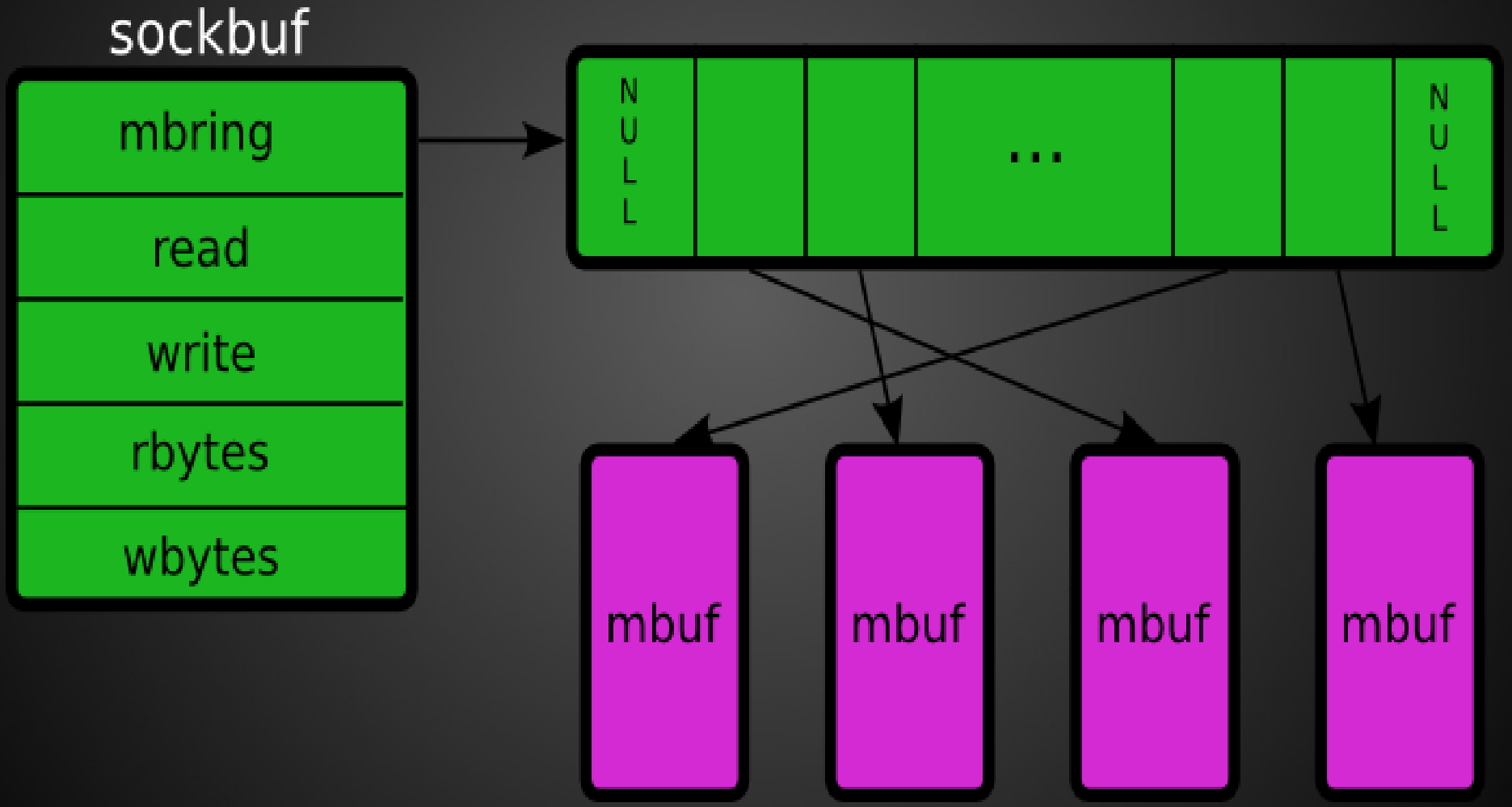
- Ενταμιευτής αποστολής (MPSC)
- Ενταμιευτής λήψης (SPMC)

Draw me a picture



Lamport

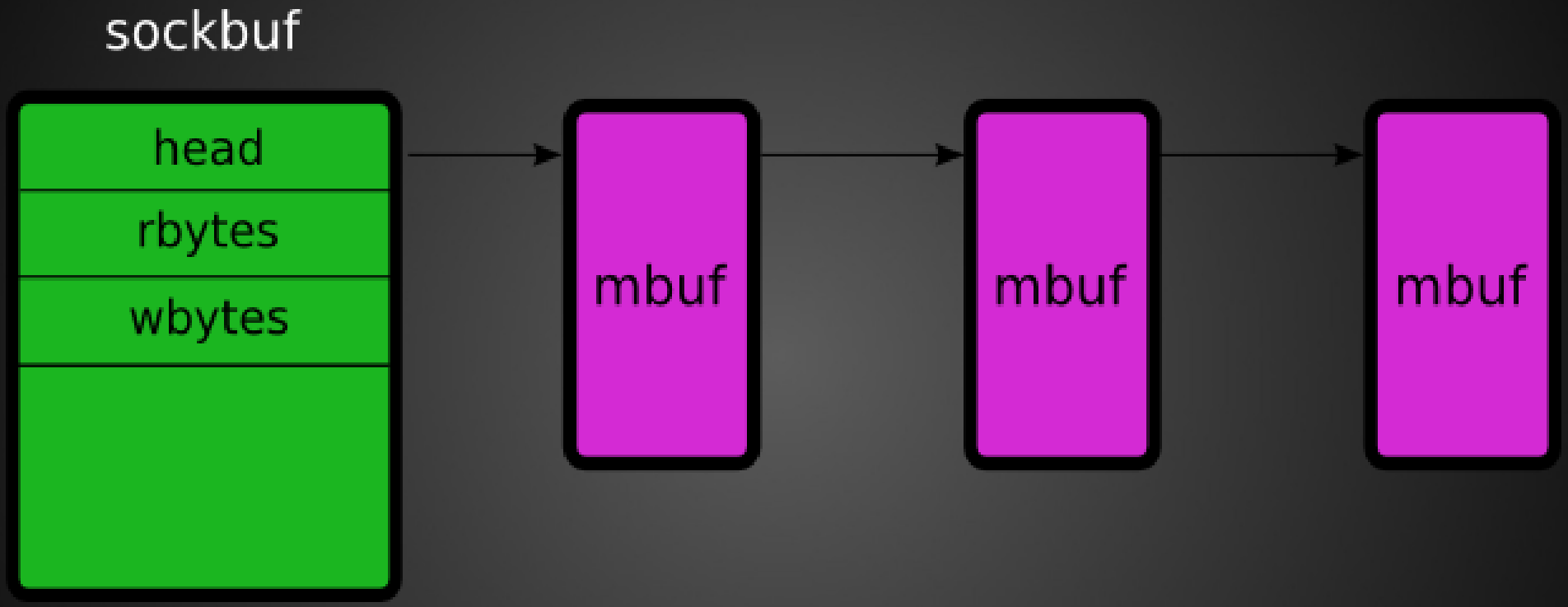



sockbuf
mbring
read
write
rbytes
wbytes
NULL
...
NULL
mbuf
mbuf
mbuf
mbuf

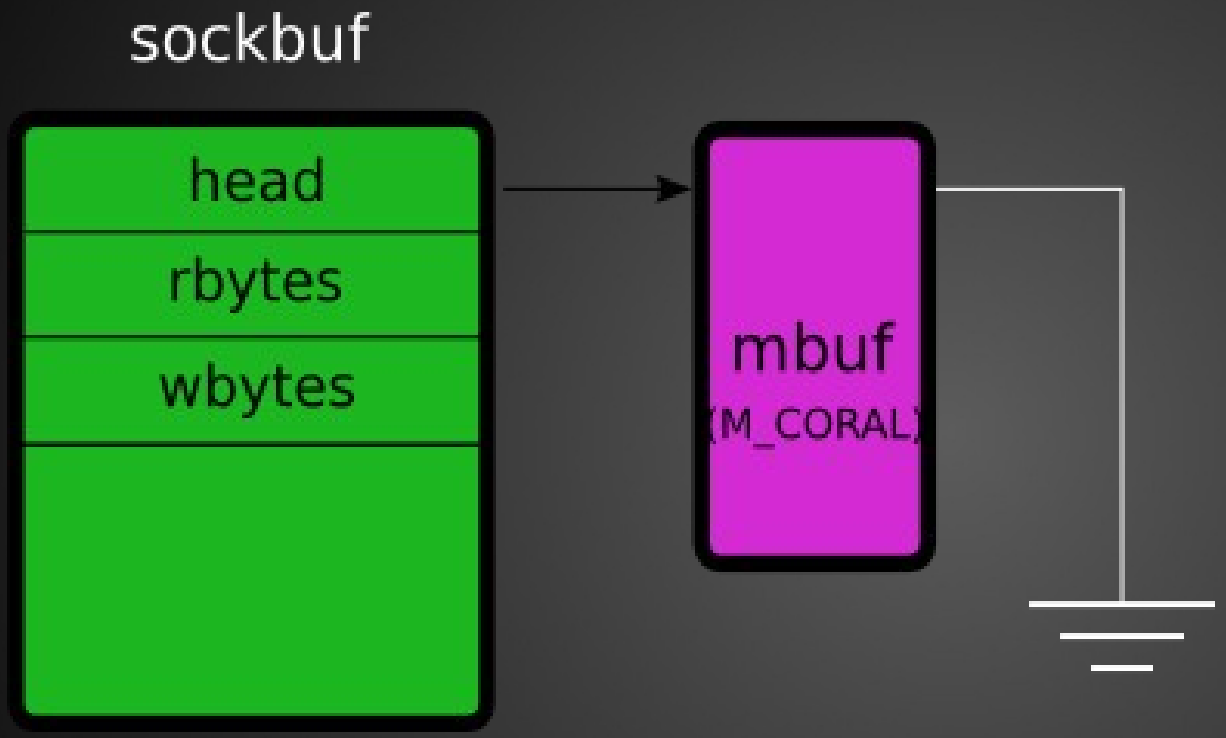
+/-Κυκλικού Ενταμιευτή

- Υπέρ
  - Ελάχιστο cacheline thrashing
  - Απλή υλοποίηση
- Κατά
  - Σπαταλά μνήμη
- Προοπτική
  - Μόνο με δυναμική αλλαγή μεγέθους

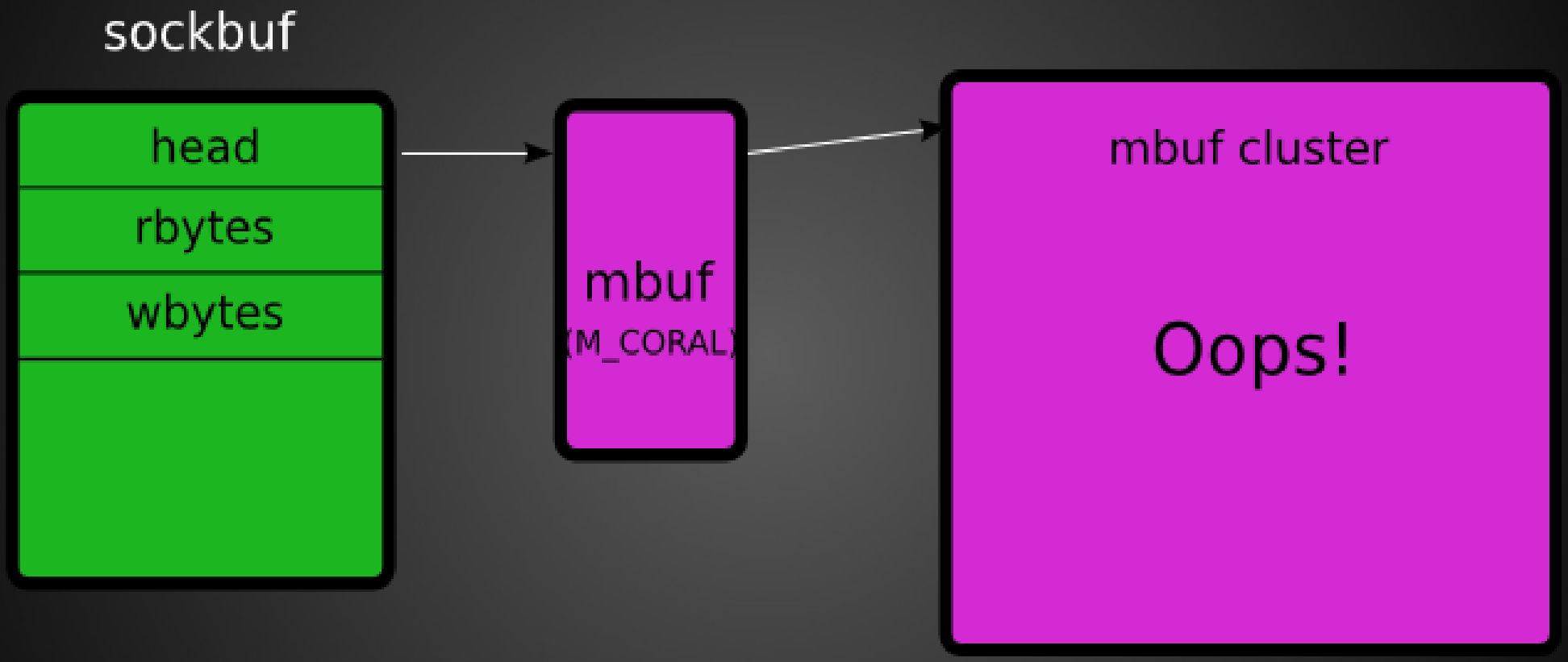
M_CORAL (1)



M_CORAL (2)



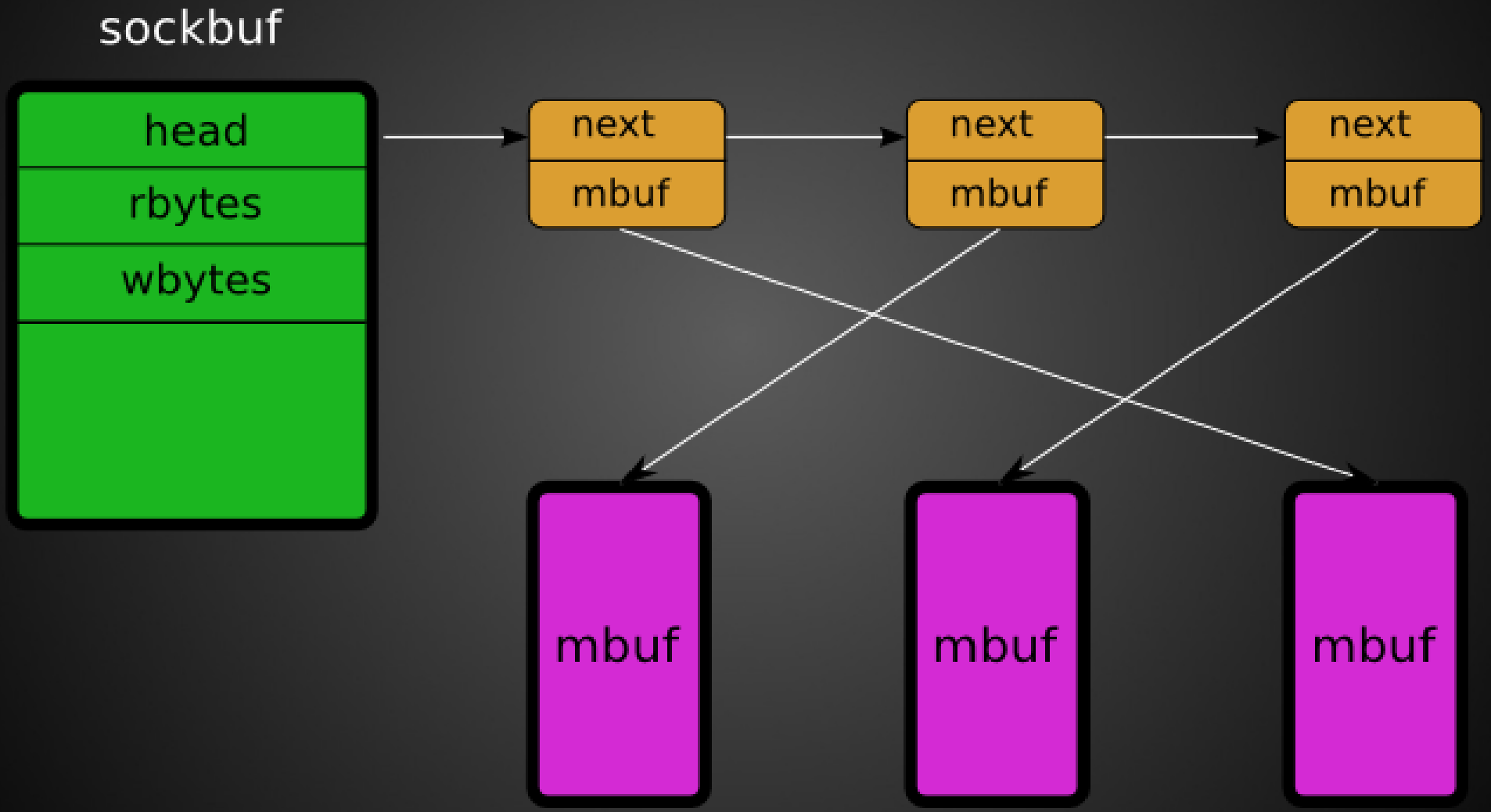
M_CORAL (3)



+/- M_CORAL

- Υπέρ
  - Δυναμικό μέγεθος
  - Δεν σπαταλά μνήμη για ανενεργά sockbuf
  - Cache-friendly
- Κατά
  - Τα mbuf clusters είναι μεγάλα (2K)
  - Περίπλοκο CFG
- Προοπτική
  - Θα χρησιμοποιηθεί μόλις λυθεί το πρόβλημα με τα mbuf clusters

cupholders


sockbuf
head
rbytes
wbytes
next
mbuf
next
mbuf
next
mbuf
mbuf
mbuf
mbuf

+/- cupholders

- Υπέρ
  - Δυναμικό μέγεθος
  - Απλός κώδικας
- Κατά
  - Larger cache footprint
  - Δέσμευση μνήμης
- Προοπτική
  - Θα μείνουν για λίγο ακόμα

Ανιχνευσιμότητα των δεδομένων

- «Πόσα bytes περιέχονται στο sockbuf;»
- Δεν μπορούμε να δώσουμε ένα νούμερο, αλλά μόνο ένα άνω ή κάτω όριο

Ζητήματα (3)

Races

Races

- Κάποια races τα αφήνουμε να εξελιχθούν
  - π.χ socket options
- Κάποια άλλα είναι επικίνδυνα
  - π.χ SS_CANTRCVMORE

Χαμένη αφύπνιση

```
get_mplock();
[...]
again:
  if (have_data)
      break;
  if (exception)
      break;
  sleep_on_sockbuf();
  goto again;
```

```
again:
    if (have_data)
        break;
    if (exception)
        break;
    syncmsg(notify_me);
    goto again;
```

Μετρήσεις

- Θεωρητικά εύκολο να δείξουμε την κλιμάκωση αλλά...
- Ιδέες:
  - Πολλές συνδέσεις με μικρή ροή δεδομένων
  - loopback + hacks
  - Χρήση 10G

10G

- mxge(4)
  - + HW csum, vlan tagging
  - - RSS, TSO, jumbo frames

Αποτελέσματα