Προγραμματισμός II (ECE116)

#20 υποδοχείς (sockets)

Σπύρος Λάλης
lalis@iuth.gr

Υποδοχείς

- Ειδικές **δομές** διαδιεργασιακής επικοινωνίας
- Προσπελάζονται μέσω **περιγραφέων** αρχείων
  - βλέπε `read`, `write`, `close` και ανακατεύθυνση Ε/Ε
- Υπάρχει δυνατότητα καθορισμού του επιθυμητού **τύπου και πρωτοκόλλου** επικοινωνίας

- **Unix sockets:** τοπική επικοινωνία ανάμεσα σε διεργασίες που εκτελούνται στον **ίδιο** Η/Υ
  - τοπικός χειρισμός από το λειτουργικό
- **Internet sockets:** (δια)δικτυακή επικοινωνία ανάμεσα σε διεργασίες σε **διαφορετικούς** Η/Υ
  - χρησιμοποιούνται συγκεκριμένα **πρωτόκολλα** επικοινωνίας που είναι **ανεξάρτητα** του εκάστοτε λειτουργικού συστήματος

Δημιουργία υποδοχέα

```
int socket(int domain, int type, int protocol);
```

- Το `domain` καθορίζει το επιθυμητό **πλαίσιο** / **πεδίο** της επικοινωνίας
  - `AF_UNIX` για Unix sockets
  - `AF_INET` για Internet sockets
- Το `type` καθορίζει τον επιθυμητό **τύπο** της επικοινωνίας
  - `SOCK_STREAM` για αξιόπιστη ροή δύο κατευθύνσεων
  - `SOCK_DGRAM` για αναξιόπιστη μετάδοση διακριτών μηνυμάτων
- Το `protocol` καθορίζει το **πρωτόκολλο** επικοινωνίας
  - μπορεί να αφεθεί ανοιχτό/απροσδιόριστο (τιμή `0`) οπότε επιλέγεται αυτόματα το προκαθορισμένο πρωτόκολλο για τον συνδυασμό που προκύπτει από το `domain` και το `type` που δίνονται ως παράμετροι

Βασικά είδη υποδοχέων

Datagram sockets

- Μετάδοση διακριτών **μηνυμάτων**
- **Δεν** δίνεται εγγύηση αξιοπιστίας/σειράς παράδοσης

Stream sockets

- Μετάδοση μιας **ροής δεδομένων**
- **Αξιόπιστα** και με σειρά **FIFO**
  - στο πνεύμα ενός FIFO pipe
- Η ροή είναι **διπλής** κατεύθυνσης (full duplex)
- Απαιτείται **σύνδεση** ανάμεσα σε δύο υποδοχείς

Συσχέτιση υποδοχέα με όνομα/διεύθυνση

```
int bind(int sockfd,
         struct sockaddr *addr,
         size_t addrlen)
```

- Συσχετίζει έναν υποδοχέα με στοιχεία επικοινωνίας (διεύθυνση), ώστε αυτός να μπορεί να **εντοπιστεί**
  - από άλλες διεργασίες

- Τα περιεχόμενα της δομής `struct sockaddr` **εξαρτώνται** από το `domain` του υποδοχέα
  - κάθε `domain` μπορεί να εισάγει **διαφορετική** δομή για τα απαιτούμενα στοιχεία επικοινωνίας

Διευθύνσεις Unix domain sockets

```
struct sockaddr_un {
  sa_family_t sun_family; /* AF_UNIX */
  char sun_path[108];     /* pathname */
};
```

- Το `sun_family` πρέπει να είναι `AF_UNIX`
- Το `sun_path` περιέχει ένα **όνομα/μονοπάτι** στον χώρο ονομάτων του συστήματος αρχείων
  - όπως για αρχεία και επώνυμους αγωγούς

- Μετά την χρήση, το όνομα πρέπει να απομακρυνθεί
  - όπως για αρχεία και επώνυμους αγωγούς

Datagram sockets

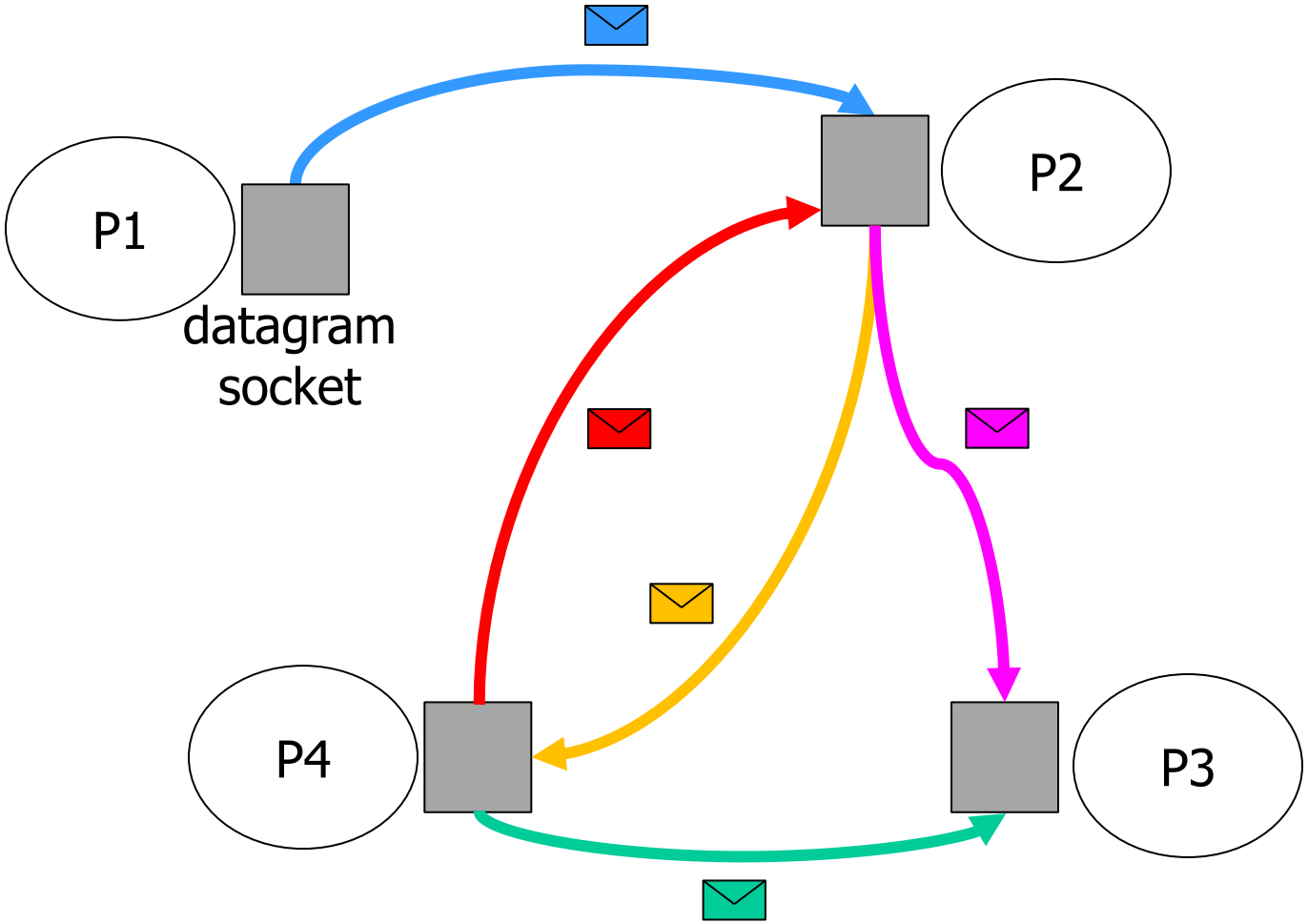
Λειτουργικότητα

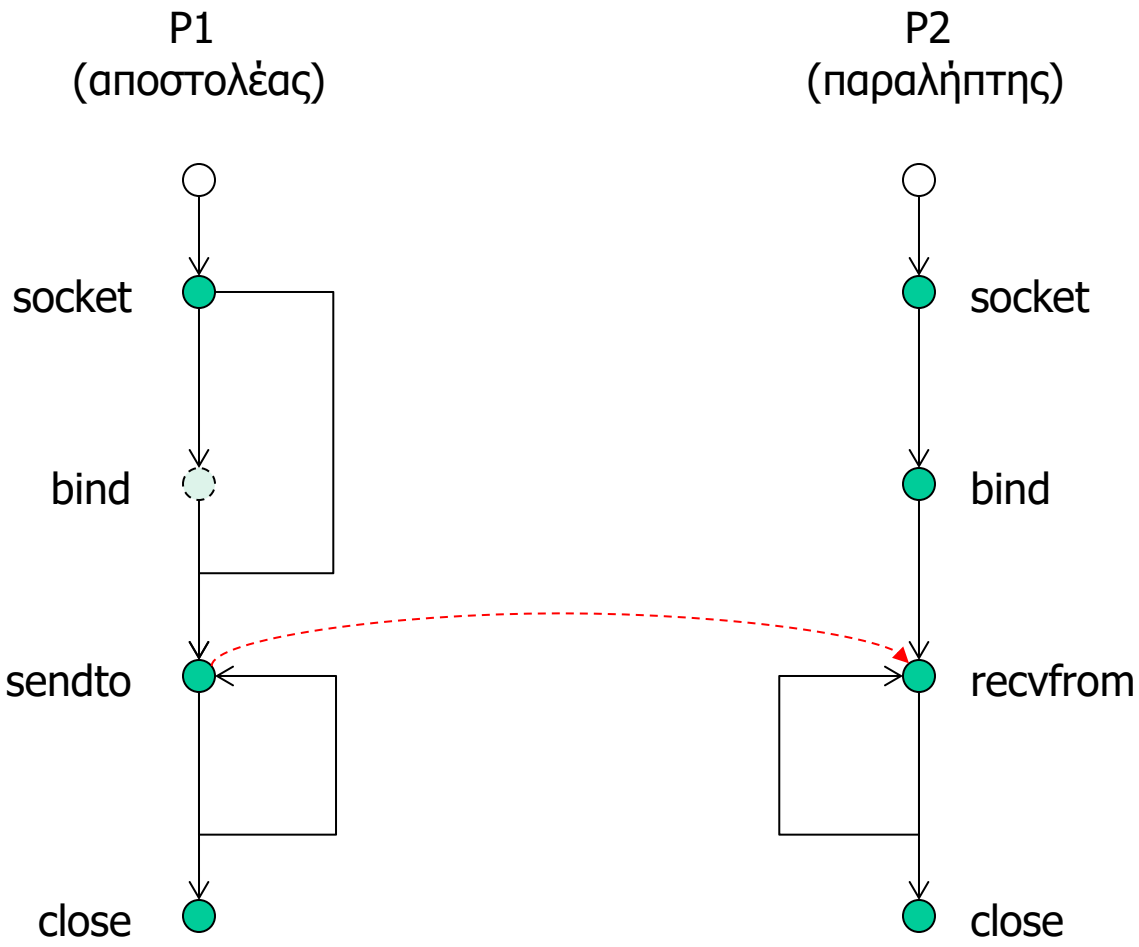
- **Αναξιόπιστη** μεταφορά **μηνυμάτων**
  - best effort

- **Δεν** χρειάζεται (απαραίτητα) κάποια «σύνδεση» μεταξύ των υποδοχέων που χρησιμοποιούνται για αυτή την επικοινωνία

- Ο **ίδιος** υποδοχέας μπορεί να χρησιμοποιηθεί για την μετάδοση/παραλαβή μηνυμάτων προς/από διαφορετικούς υποδοχείς

- Κάθε μήνυμα είναι **διακριτό** από τα υπόλοιπα


P1
datagram
socket
P2
P3
P4


P1
(αποστολέας)
P2
(παραλήπτης)
socket
bind
sendto
close
socket
bind
recvfrom
close

Αποστολή μηνύματος

```
ssize_t sendto(int sockfd,
               const void *buf, size_t len,
               int flags,
               const struct sockaddr *addr,
               socklen_t addrlen);
```

- Στέλνει το μήνυμα `buf` μεγέθους `len` στην διεύθυνση `addr` που δίνεται με την ίδια μορφή όπως στην `bind`
- Αν ο υποδοχέας «συνδεθεί» (μέσω `connect`) σε μια συγκεκριμένη διεύθυνση, η `addr` μπορεί να είναι `NULL` ή χρησιμοποιείται η λειτουργία `send` (που δεν δέχεται ως παράμετρο κάποια διεύθυνση)

Παραλαβή μηνύματος

```
ssize_t recvfrom(int sockfd,
                 void *buf, size_t len,
                 int flags,
                 struct sockaddr *addr,
                 socklen_t *addrlen);
```

- Μπλοκάρει μέχρι να παραληφθεί το επόμενο μήνυμα, που αποθηκεύεται στο `buf` με μέγιστο μέγεθος `len`
- Η διεύθυνση του αποστολέα αποθηκεύεται στην `addr`
- Αν δεν είναι επιθυμητή η παραλαβή της διεύθυνσης, η `addr` μπορεί να είναι `NULL` ή χρησιμοποιείται η `recv` (που δεν επιστρέφει διεύθυνση)

client (sender)

```
int main(int argc, char *argv[]) {
  int s, n, count;
  struct sockaddr_un addr;

  count = atoi(argv[2]);

  s = socket(AF_UNIX, SOCK_DGRAM, 0);

  addr.sun_family = AF_UNIX;
  strcpy(addr.sun_path, argv[1]);

  do {
    printf("sending ...\n");
    n = sendto(s, &count, sizeof(int), 0,
                  (struct sockaddr *)&addr, sizeof(addr));
    if (n <= 0) { perror("send"); break; }
    printf("sent value %d (%d bytes)\n", count, n);
  } while (--count > 0);

  close(s);

  return(0);
}
```

server (receiver)

```
int main(int argc, char *argv[]) {
  int s, n, val;
  struct sockaddr_un addr;

  s = socket(AF_UNIX, SOCK_DGRAM, 0);

  addr.sun_family = AF_UNIX;
  strcpy(addr.sun_path, argv[1]);
  bind(s, (struct sockaddr *)&addr, sizeof(addr));

  do {
    printf("receiving ...\n");
    n = recvfrom(s, &val, sizeof(int), 0, NULL, NULL);
    if (n < 0) { perror("receive"); break; }
    printf("received value %d (%d bytes)\n", val, n);
  } while ((n > 0) && (val != -1));

  close(s);
  unlink(argv[1]);

  return(0);
}
```

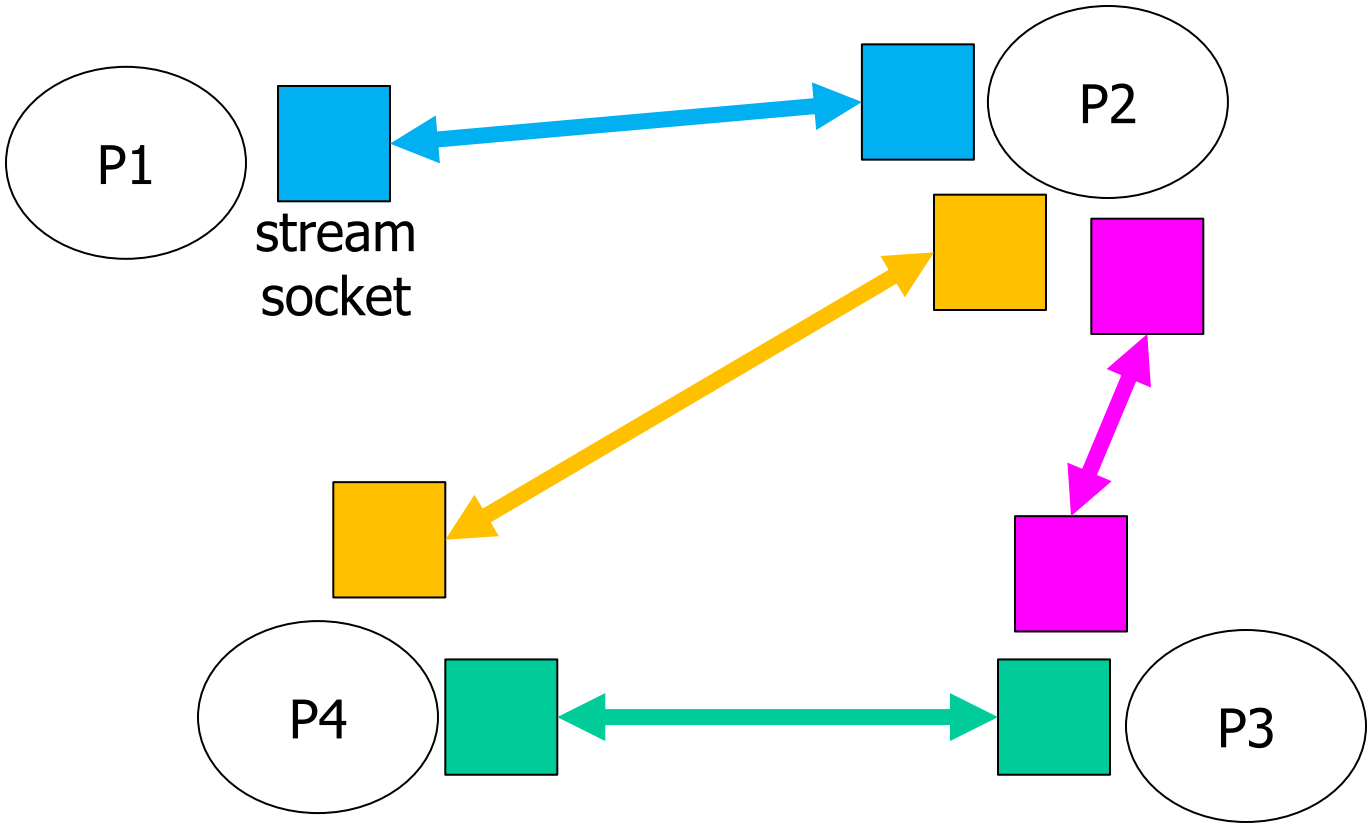
Stream sockets

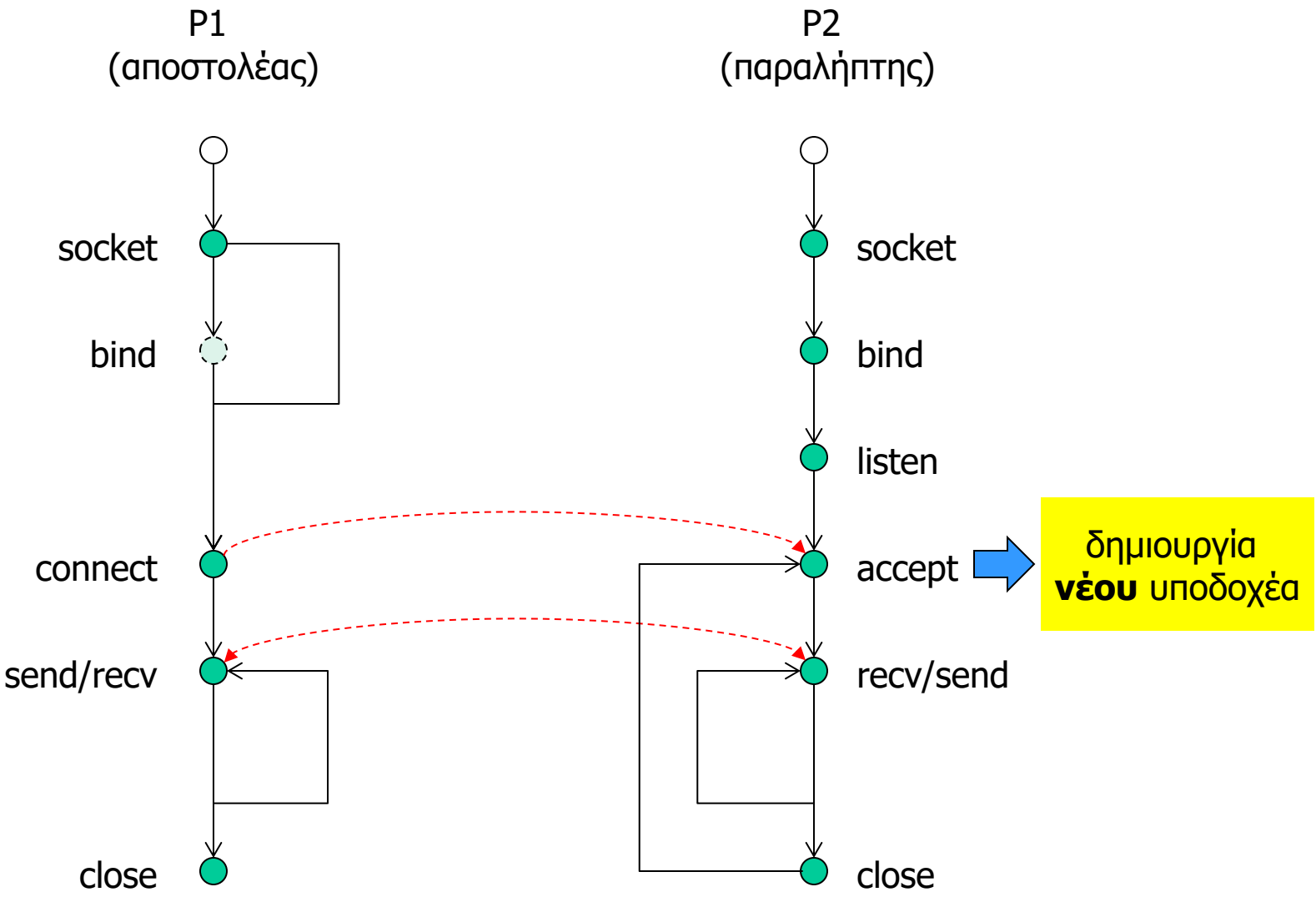
Λειτουργικότητα

- **Αξιόπιστη** μεταφορά **ροής δεδομένων**
  - μπορεί να είναι ταυτόχρονα διπλής κατεύθυνσης (full duplex)

- Απαιτείται **σύνδεση** ανάμεσα στους δύο υποδοχείς
  - η μια πλευρά **αναμένει/αποδέχεται** την σύνδεση
  - η άλλη πλευρά **αιτείται/δημιουργεί** την σύνδεση

- Στην πλευρά που αποδέχεται την σύνδεση δημιουργείται (αυτόματα) ένας **νέος** υποδοχέας
- Κάθε σύνδεση είναι **ανεξάρτητη** από τις υπόλοιπες
- Ένας συνδεδεμένος υποδοχέας μπορεί να χρησιμοποιηθεί για την μετάδοση/παραλαβή δεδομένων **μόνο** πάνω από τη συγκεκριμένη σύνδεση


P1
stream
socket
P2
P4
P3


P1
(αποστολέας)
P2
(παραλήπτης)
socket
bind
connect
send/recv
close
socket
bind
listen
accept
recv/send
close
δημιουργία
νέου υποδοχέα

Διαδικασία σύνδεσης

```
int listen(int sockfd, int backlog);
```

- **Πρόθεση** αποδοχής αιτήσεων για σύνδεση

```
int accept(int sockfd,
           const struct sockaddr *saddr,
           size_t *saddrlen) ;
```

- Αποδοχή (επόμενης) σύνδεσης
- Αποθήκευση διεύθυνσης της απέναντι πλευράς
- Επιστρέφει **νέο υποδοχέα** για την σύνδεση

```
int connect(int sockfd,
            const struct sockaddr *addr,
            socklen_t addrlen);
```

- **Σύνδεση** με μια συγκεκριμένη διεύθυνση

Αποστολή και παραλαβή δεδομένων

- Αποστολή: `send` (`write`)
- Παραλαβή: `recv` (`read`)

- Υφίσταται «μόνιμη» σύνδεση ανάμεσα στους δύο υποδοχείς
- Ο αποστολέας **δεν** χρειάζεται να προσδιορίζει κάθε φορά (εκ νέου) την διεύθυνση του παραλήπτη
- Ο παραλήπτης **δεν** χρειάζεται να παραλαμβάνει κάθε φορά (εκ νέου) την διεύθυνση του αποστολέα

client (sender)

```
int main(int argc, char *argv[]) {
  int s, n, count;
  struct sockaddr_un addr;

  count = atoi(argv[2]);

  s = socket(AF_UNIX, SOCK_STREAM, 0);

  addr.sun_family = AF_UNIX;
  strcpy(addr.sun_path, argv[1]);

  connect(s, (struct sockaddr *)&addr, sizeof(addr));

  do {
    printf("sending ...\n");
    n = send(s, &count, sizeof(int), 0);
    if (n <= 0) { perror("send"); break; }
    printf("sent value %d (%d bytes)\n", count, n);
  } while (--count > 0);

  close(s);

  return(0);
}
```

server (receiver)

```
int main(int argc, char *argv[]) {
  int s, s2, n, val;
  struct sockaddr_un addr;

  s2 = socket(AF_UNIX, SOCK_STREAM, 0);

  addr.sun_family = AF_UNIX;
  strcpy(addr.sun_path, argv[1]);
  bind(s2, (struct sockaddr *)&addr, sizeof(addr));

  listen(s2, 0);

  do {
    s = accept(s2, NULL, NULL);
    printf("new connection ...\n");
    do {
      printf("receiving ...\n");
      n = recv(s, &val, sizeof(int), 0);
      if (n < 0) { perror("receive"); break; }
      printf("received value %d (%d bytes)\n", val, n);
    } while ((n > 0) && (val != -1));
    close(s);
  } while (val != -1);

  close(s2);
  unlink(argv[1]);

  return(0);
}
```

Επικοινωνία πάνω από δίκτυο με UDP/IP και TCP/IP sockets

Δίκτυα υπολογιστών

Local area networks

- Σχετικά μικρές αποστάσεις
- Ethernet, WiFi

Metropolitan / wide area networks

- Μέτριες αποστάσεις (πόλης)
- FDDI, WiMax

Global networks

- Πλανητική κλίμακα
- Internet, 4/5G

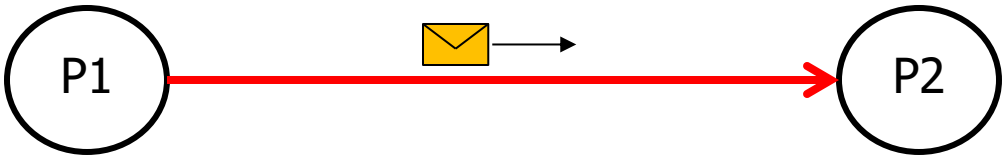
Επικοινωνία πάνω από το δίκτυο

- Οι περισσότερες τεχνολογίες δικτύων υπολογιστών υποστηρίζουν την αποστολή/παραλαβή **πακέτων**
  - τα πακέτα στέλνονται σε συγκεκριμένες **διευθύνσεις**
  - κάθε υπολογιστής έχει **διαφορετική** διεύθυνση (τουλάχιστον, σε επίπεδο τοπικού δικτύου)

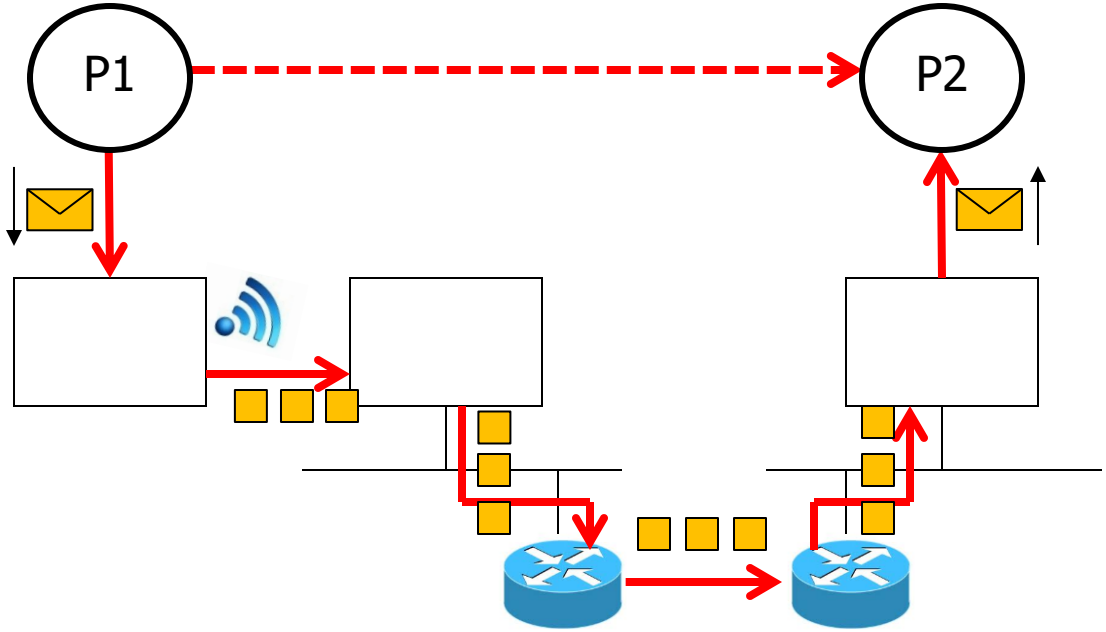
- Τα πακέτα μπορεί να **χαθούν**
  - προβλήματα μετάδοσης πάνω από το φυσικό μέσο
  - προβλήματα δρομολόγησης (για διαδίκτυα)
  - προβλήματα αποθήκευσης σε λειτουργικό

- Ένα μήνυμα μπορεί να σπάσει σε πακέτα, και να επανασυναρμολογηθεί αργότερα (στον προορισμό)

Αφαίρεση



Πραγματικότητα



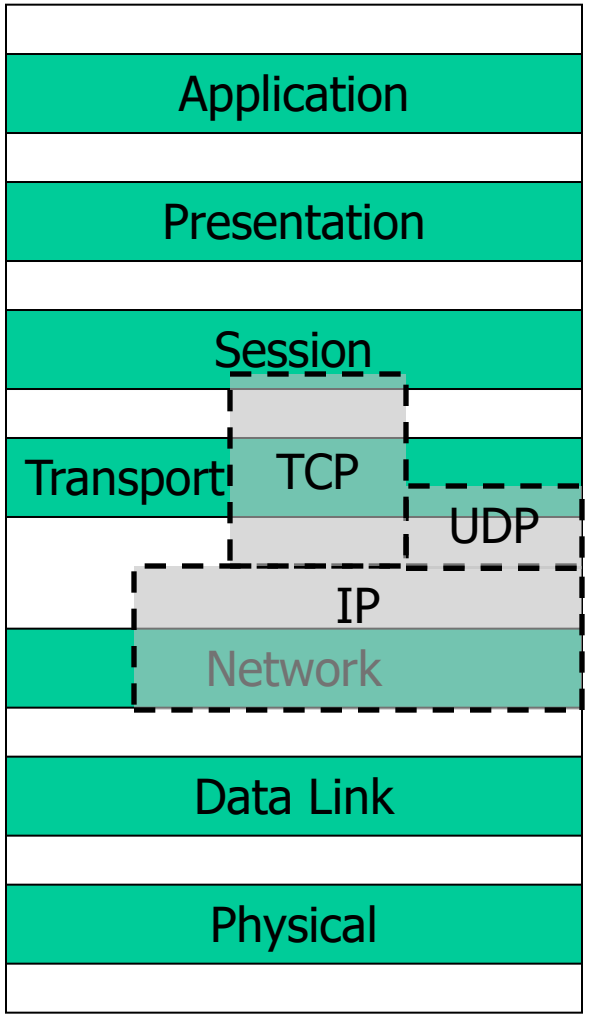
Στοίβες πρωτοκόλλων

- Για λόγους δομημένης ανάπτυξης, η λειτουργικότητα επικοινωνίας υλοποιείται συνήθως σε **στρώματα**

- Κάθε στρώμα εστιάζει στην επίλυση / διαχείριση συγκεκριμένων προβλημάτων επικοινωνίας
  - με βάση την λειτουργικότητα των πιο χαμηλών στρωμάτων

- Το στρώμα N βασίζεται στο «κάτω» στρώμα N-1
- Παρέχει λειτουργικότητα στο «πάνω» στρώμα N+1
- Μιλάμε για μια **στοίβα πρωτοκόλλων**
  - ISO-OSI, Internet protocols, κτλ

Η στοίβα του OSI & πρωτοκόλλων Internet

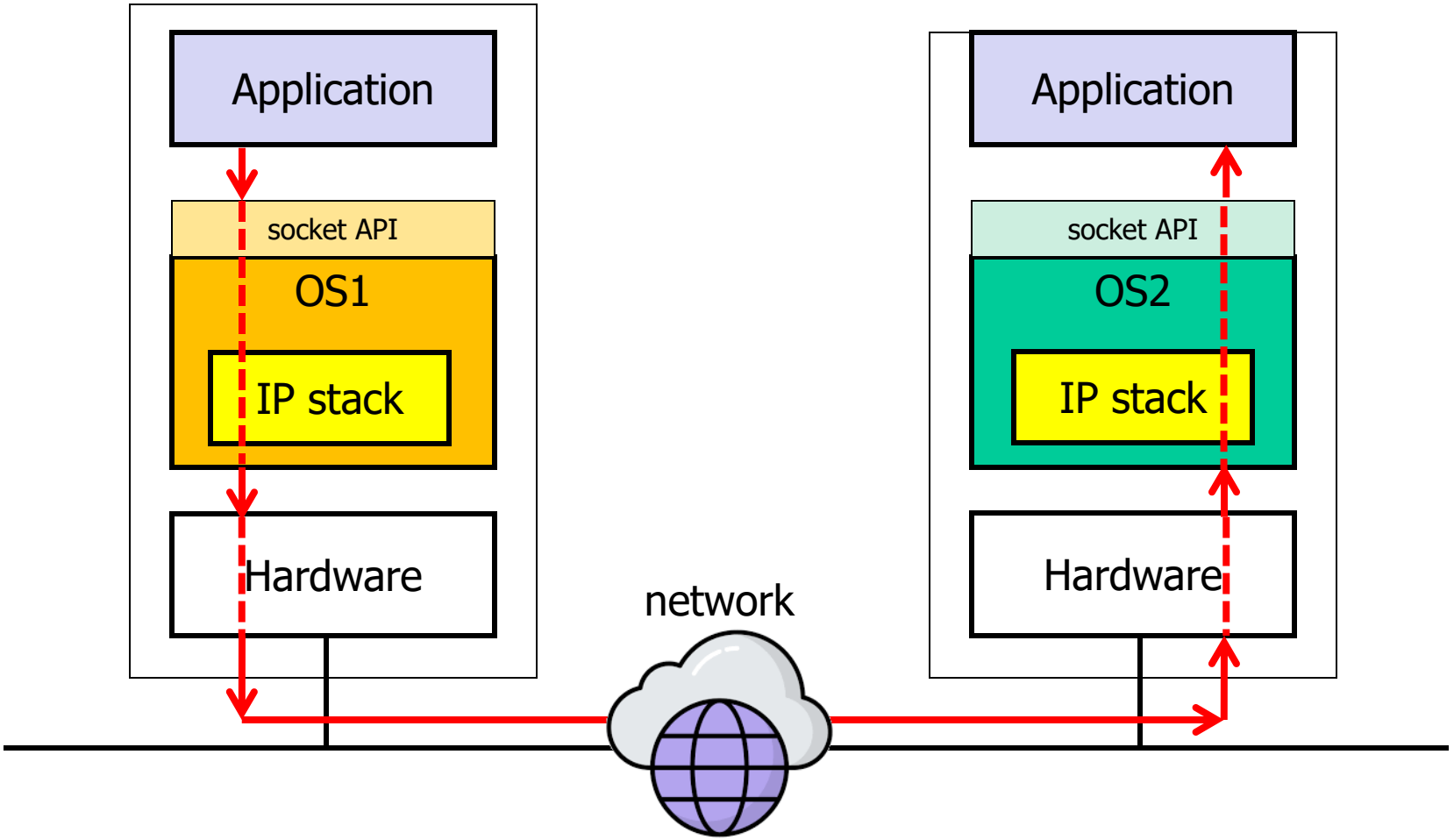


TCP:
- port-based local addressing
- reliable byte stream
- bidirectional data transfer
- error checking via checksums
- sequencing, flow control, buffering

UDP:
- port-based local addressing
- unreliable messages, up to 64K

IP:
- machine addressing, routing
- unreliable messages, up to 64K
- network technology independence
- packet fragmentation / re-assembly
- protocol (de)multiplexing

Υποστήριξη UDP/IP και TCP/IP



UDP/IP & TCP/IP sockets

- Για Internet sockets το `domain` είναι `AF_INET`

- Το προκαθορισμένο πρωτόκολλο για υποδοχείς τύπου `SOCK_DGRAM` είναι το **UDP/IP**
- Το προκαθορισμένο πρωτόκολλο για υποδοχείς τύπου `SOCK_STREAM` είναι το **TCP/IP**

- Οι διευθύνσεις επικοινωνίας είναι σύνθετες
  - **IP address:** διεύθυνση μηχανήματος στο διαδίκτυο
  - **port number:** τοπικός αριθμός θύρας/πόρτας
- Οι αριθμοί θύρας επιτρέπουν την **πολυπλεξία** επικοινωνίας στον ίδιο Η/Υ
  - για προγράμματα που εκτελούνται τοπικά ταυτόχρονα

Διευθύνσεις UDP/TCP sockets

```
struct sockaddr_in {
  sa_family_t sin_family;  /* AF_INET */
  in_port_t sin_port;      /* port number */
  struct in_addr sin_addr; /* IP address */
};
struct in_addr {
  uint32_t s_addr; /* encoded IP address */
};
```

- Το `sin_family` πρέπει να είναι `AF_INET`
- Το `sin_port` περιέχει τον **αριθμό πόρτας**
- Το `sin_addr` περιέχει την **διεύθυνση** στο δίκτυο
- Τα bytes των αριθμών πόρτας και διεύθυνσης πρέπει να δίνονται σε **network byte order**

client (sender)

```
int main(int argc, char *argv[]) {
  int s, n, count;
  struct sockaddr_in addr;

  count = atoi(argv[3]);

  s = socket(AF_INET, SOCK_DGRAM, 0);

  addr.sin_family = AF_INET;
  inet_aton(argv[1], &addr.sin_addr);
  addr.sin_port = htons(atoi(argv[2]));

  do {
    printf("sending ...\n");
    n = sendto(s, &count, sizeof(int), 0,
                  (struct sockaddr *)&addr, sizeof(addr));
    if (n <= 0) { perror("send"); break; }
    printf("sent value %d (%d bytes)\n", count, n);
  } while (--count > 0);

  close(s);

  return(0);
}
```

server (receiver)

```
int main(int argc, char *argv[]) {
  int s, n, val;
  struct sockaddr_in addr;

  s = socket(AF_INET, SOCK_DGRAM, 0);

  addr.sin_family = AF_INET;
  addr.sin_addr.s_addr = htonl(INADDR_ANY);
  addr.sin_port = htons(atoi(argv[1]));
  bind(s, (struct sockaddr *)&addr, sizeof(addr));

  do {
    printf("receiving ...\n");
    n = recvfrom(s, &val, sizeof(int), 0, NULL, NULL);
    if (n < 0) { perror("receive"); break; }
    printf("received value %d (%d bytes)\n", val, n);
  } while ((n > 0) && (val != -1));

  close(s);

  return(0);
}
```