

Programmation Réseau

# API C UDP



Jean-Baptiste.Yunes@univ-paris-diderot.fr

UFR Informatique

2013-2013

# Le mode paquet en C

- pour le mode paquet, c'est le type de la socket qui doit être modifié
  - `SOCK_DGRAM`
- pour les envois, il n'est pas nécessaire de connecter la socket via `connect()`
- il ne s'agit d'ailleurs que d'une pseudo-connexion
  - aucun protocole sous-jacent n'est mis en œuvre...

- pour l'émission on dispose des fonctions :
  - `send()`, uniquement si la socket est associée à une adresse (pseudo-connectée donc)
  - `sendto()` qui est recommandé
  - `sendmsg()` pour les messages fragmentés (scattered)

```
Terminal — less — 80x24

SEND(2)                                BSD System Calls Manual                                SEND(2)

NAME
    send, sendmsg, sendto -- send a message from a socket

SYNOPSIS
    #include <sys/socket.h>

    ssize_t
    send(int socket, const void *buffer, size_t length, int flags);

    ssize_t
    sendmsg(int socket, const struct msghdr *message, int flags);

    ssize_t
    sendto(int socket, const void *buffer, size_t length, int flags,
           const struct sockaddr *dest_addr, socklen_t dest_len);

DESCRIPTION
    Send(), sendto(), and sendmsg() are used to transmit a message to another
    socket. Send() may be used only when the socket is in a connected state,
    while sendto() and sendmsg() may be used at any time.

:
```

- pour la réception on dispose de :
  - `recv()` pour la réception sur une socket associée à une adresse
  - `recvfrom()`, recommandé
  - `recvmsg()` pour les réceptions à défragmenter (gathered)

```
Terminal — less — 80x24

RECV(2)                                BSD System Calls Manual                                RECV(2)

NAME
    recv, recvfrom, recvmsg -- receive a message from a socket

LIBRARY
    Standard C Library (libc, -lc)

SYNOPSIS
    #include <sys/socket.h>

    ssize_t
    recv(int socket, void *buffer, size_t length, int flags);

    ssize_t
    recvfrom(int socket, void *restrict buffer, size_t length, int flags,
             struct sockaddr *restrict address, socklen_t *restrict address_len);

    ssize_t
    recvmsg(int socket, struct msghdr *message, int flags);

DESCRIPTION
```

```
:
```

```
int main(int argc, char *argv[]) {
    struct sockaddr_in addr; int sock; char tampon[256]; struct hostent *hent;

    if (argc < 3) {
        fprintf(stderr, "usage: %s host message\n", argv[0]); exit(1);
    }
    hent = gethostbyname(argv[1]);
    if (hent == NULL) {
        fprintf(stderr, "%s: host %s unknown\n", argv[0], argv[1]); exit(1);
    }
    sock = socket(PF_INET, SOCK_DGRAM, 0); // Protocol family
    if (sock == -1) {
        perror("socket: "); exit(1);
    }
    addr.sin_family = AF_INET; // Address family
    addr.sin_port = htons(PORT);
    memcpy(&(addr.sin_addr.s_addr), hent->h_addr_list[0], hent->h_length);
    strcpy(tampon, argv[2]);
    if (sendto(sock, tampon, 256, 0, (struct sockaddr *)&addr, sizeof(addr)) == -1) {
        perror("sendto:"); close(sock); exit(1);
    }
    close(sock);
    return 0;
}
```

```
int main(int argc, char *argv[]) {
    struct sockaddr_in addr; int sock; char tampon[256];

    sock = socket(PF_INET, SOCK_DGRAM, 0); // Protocol family
    if (sock == -1) {
        perror("socket: "); exit(1);
    }
    addr.sin_family = AF_INET; // Address family
    addr.sin_port = htons(PORT);
    addr.sin_addr.s_addr = htonl(INADDR_ANY);
    if (bind(sock, (struct sockaddr *)&addr, sizeof(addr)) == -1) {
        perror("bind: "); close(sock); exit(1);
    }
    while (1) {
        if (recv(sock, tampon, 256, 0) == -1) {
            perror("recv:"); close(sock); exit(1);
        }
        printf("Recu : %s\n", tampon);
        if (!strcmp(tampon, "quit")) break;
    }
    close(sock);
    return 0;
}
```



## Un exemple : echo

- L'implémentation d'un service d'écho nécessite
  - l'association, côté client, à une adresse et un port pour recevoir les réponses
  - l'utilisation de `recvfrom()`, côté serveur, afin de récupérer l'adresse du client afin de lui renvoyer la réponse

```
int main(int argc, char *argv[]) {
    struct sockaddr_in addrto, addrfrom; int sock; char tampon[256]; struct hostent *hent;

    hent = gethostbyname(argv[1]);
    if (hent==NULL) { fprintf(stderr, "%s: host %s unknown\n", argv[0], argv[1]); exit(1); }
    sock = socket(PF_INET, SOCK_DGRAM, 0); // Protocol family
    if (sock==-1) { perror("socket: "); exit(1); }
    addrfrom.sin_family = AF_INET; // Address family
    addrfrom.sin_port = htons(0); // Any port
    addrfrom.sin_addr.s_addr = htonl(INADDR_ANY); // Any address
    if (bind(sock, (struct sockaddr *)&addrfrom, sizeof(addrfrom))==-1) {
        perror("bind: "); close(sock); exit(1);
    }
    addrto.sin_family = hent->h_addrtype; // destination from db
    memcpy(&(addrto.sin_addr.s_addr), hent->h_addr_list[0], hent->h_length);
    addrto.sin_port = htons(PORT); // fixed port
    strcpy(tampon, argv[2]);
    if (sendto(sock, tampon, 256, 0, (struct sockaddr *)&addrto, sizeof(addrto))==-1) {
        perror("sendto:"); close(sock); exit(1);
    }
    if (strcmp(tampon, "quit")) {
        if (recv(sock, tampon, 256, 0)==-1) {
            perror("recv:"); close(sock); exit(1);
        }
        printf("Recu : %s\n", tampon);
    }
    close(sock);
    return 0;
}
```

```
int main(int argc, char *argv[]) {
    struct sockaddr_in addr, from; int sock; char tampon[256]; socklen_t lg;

    sock = socket(PF_INET, SOCK_DGRAM, 0); // Protocol family
    if (sock == -1) { perror("socket: "); exit(1); }
    addr.sin_family = AF_INET;           // Address family
    addr.sin_port = htons(PORT);
    addr.sin_addr.s_addr = htonl(INADDR_ANY);
    if (bind(sock, (struct sockaddr *)&addr, sizeof(addr)) == -1) {
        perror("bind: "); close(sock); exit(1);
    }
    while (1) {
        lg = sizeof(from);
        if (recvfrom(sock, tampon, 256, 0, (struct sockaddr *)&from, &lg) == -1) {
            perror("recv:"); close(sock); exit(1);
        }
        printf("Recu : %s\n", tampon);
        if (!strcmp(tampon, "quit")) break;
        if (sendto(sock, tampon, 256, 0, (struct sockaddr *)&from, lg) == -1) {
            perror("sendto:"); close(sock); exit(1);
        }
        printf("Renvoye : %s\n", tampon);
    }
    close(sock);
    return 0;
}
```