PROGRAMMATION RÉSEAU

Arnaud Sangnier
sangnier@irif.fr

API TCP C - IV

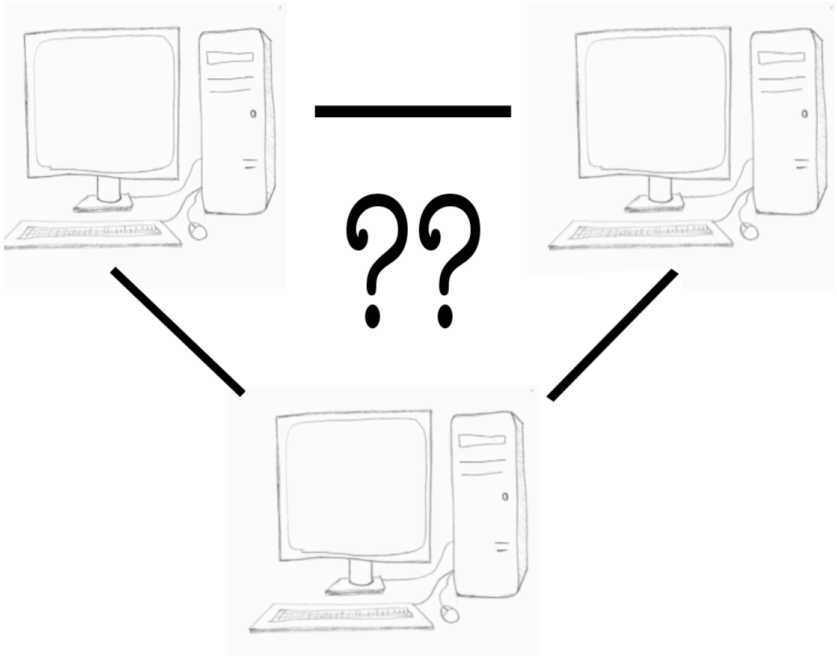
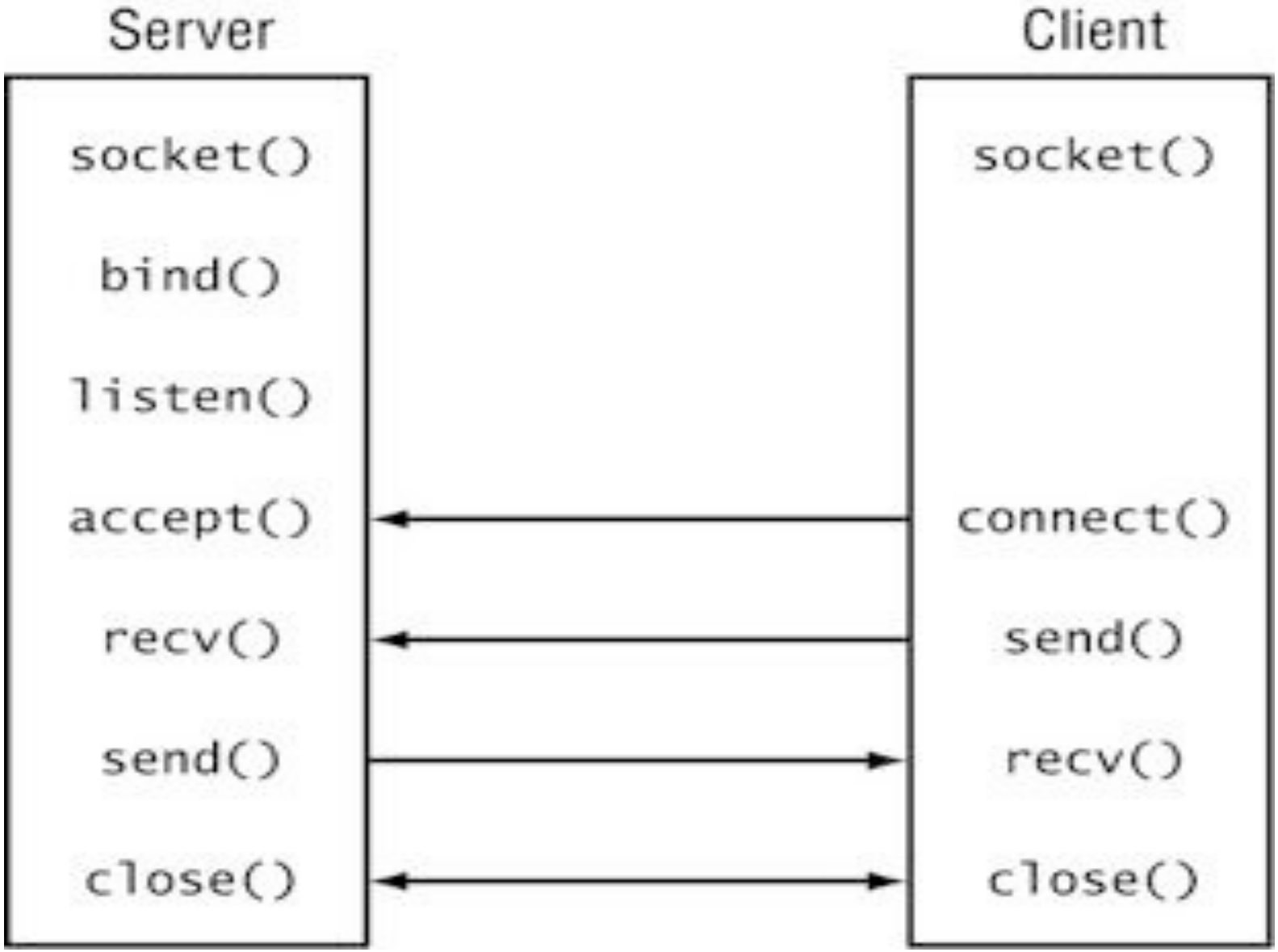


Schéma Client-Serveur en C



Côté Serveur - Lier la socket à un port

- Il faut associer la socket à un port donné
- **int bind(int sockfd, struct sockaddr *my_addr, int addrlen);**
- Comme pour connect, en IPv4, le deuxième argument sera souvent de type **struct sockaddr_in** et le troisième sera **sizeof(struct sockaddr_in)**
- Comme on est sur le serveur, on n'a pas besoin de spécifier l'adresse de la machine dans la plupart des cas, donc on mettra comme adresse en remplissant la valeur **htonl(INADDR_ANY)**
- Le numéro de port est fourni en remplissant la structure du deuxième argument

Côté Serveur - exemple pour bind

```
int sock=socket(PF_INET,SOCK_STREAM,0);
struct sockaddr_in address_sock;
address_sock.sin_family=AF_INET;
address_sock.sin_port=htons(4242);
address_sock.sin_addr.s_addr=htonl(INADDR_ANY);
int r=bind(sock,(struct sockaddr *)&address_sock,sizeof(struct
sockaddr_in));
```

- La ligne **address_sock.sin_addr.s_addr=htonl(INADDR_ANY);** sert à préciser que l'on peut prendre n'importe quelle adresse dans la structure
  - On remplit le champ **s_addr** de la structure **struct in_addr**

Côté Serveur - Écouter sur le port

- Une fois associée à un port, il faut faire de la socket et une socket serveur
  - On fait en sorte que le système autorise les demandes de connexion entrantes
  - On peut aussi préciser le nombre de demandes en attente possibles
- La fonction qui fait cela
  - **int listen(int sockfd, int backlog);**
- **backlog** précise le nombre de demandes en attente autorisé
- En général, on le met à 0 pour laisser le système décidé

```
r=listen(sock,0) ;
```

Côté Serveur - Accepter une connexion

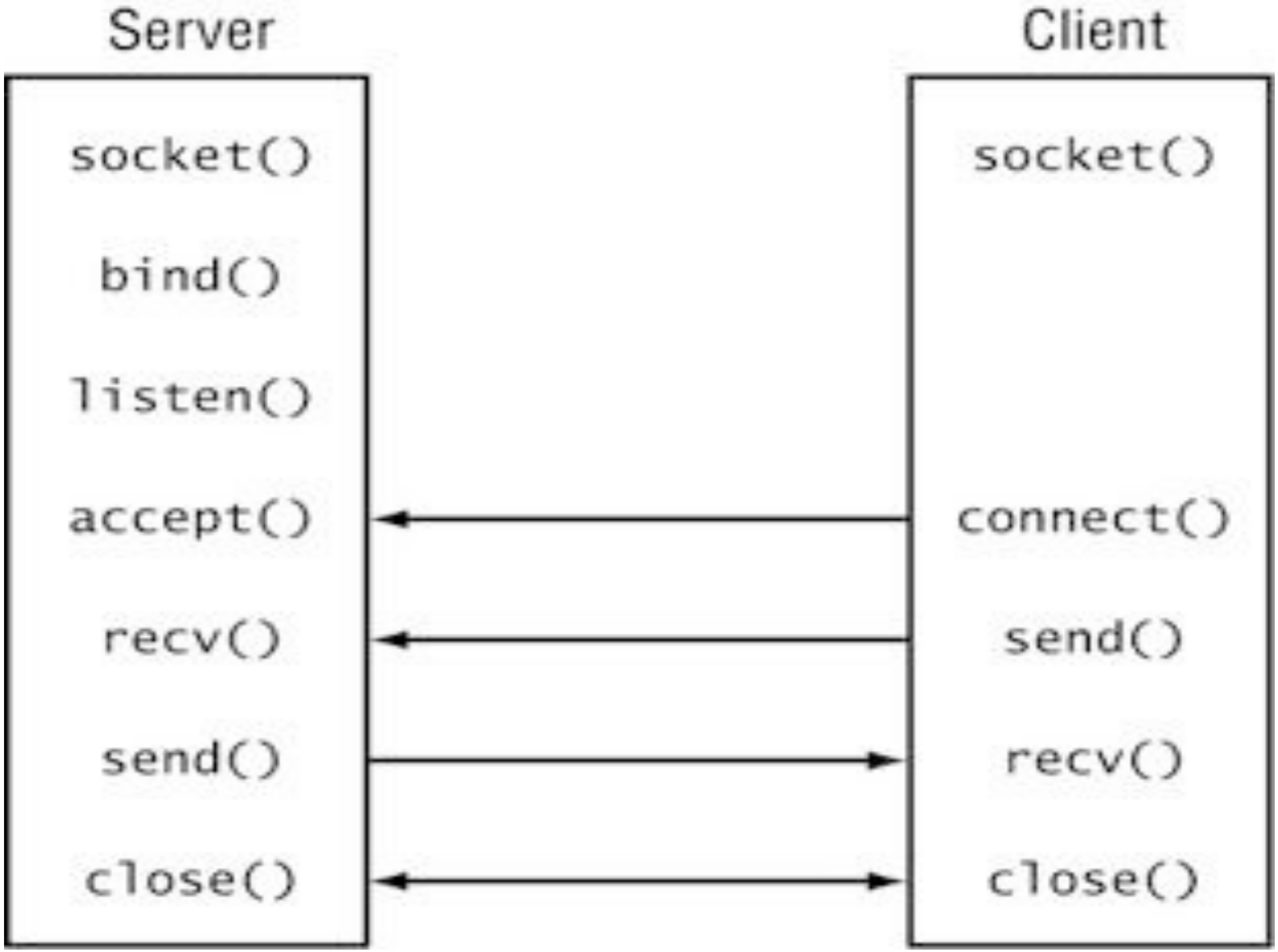
- L'acceptation d'une demande de connexion se fait grâce :
  - **int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);**
- Attend qu'une demande de connexion arrive si la file d'attente est vide
- On verra plus tard comment passer en mode non-bloquant
- Extrait une demande de la file d'attente
- **Renvoie un descripteur de la socket créé pour communiquer**
- De plus, cette fonction remplit les champs addr et addrlen avec des infos sur qui s'est connecté
- On pourra en particulier savoir quel hôte s'est connecté sur quel port
- **Erreur classique : communiquer sur sockfd !!!!!!!**

Côté Serveur - Utilisation d'accept

```
struct sockaddr_in caller;
socklen_t size=sizeof(caller);
int sock2=accept(sock,(struct sockaddr *)&caller,&size);
```

- Quand une connexion est acceptée, le programme remplit la structure caller avec les informations sur qui se connecte
- On communique ensuite sur **sock2**
- Ne pas oublier de fermer cette socket (et pas **sock**) avant d'accepter une nouvelle communication

Schéma Client-Serveur en C



Exemple

```
int main() {
  int sock=socket(PF_INET,SOCK_STREAM,0);
  struct sockaddr_in address_sock;
  address_sock.sin_family=AF_INET;
  address_sock.sin_port=htons(4242);
  address_sock.sin_addr.s_addr=htonl(INADDR_ANY);
  int r=bind(sock,(struct sockaddr *)&address_sock,sizeof(struct sockaddr_in));
  if(r==0){
    r=listen(sock,0);
    while(1){
      struct sockaddr_in caller;
      socklen_t size=sizeof(caller);
      int sock2=accept(sock,(struct sockaddr *)&caller,&size);
      if(sock2>=0){
        char *mess="Yeah!\n";
        send(sock2,mess,strlen(mess)*sizeof(char),0);
        char buff[100];
        int recu=recv(sock2,buff,99*sizeof(char),0);
        buff[recu]='\0';
        printf("Message recu : %s\n",buff);
      }
      close(sock2);
    }
  }
  return 0;
}
```

Récupération d'informations

```
int main() {
  int sock=socket(PF_INET,SOCK_STREAM,0);
  struct sockaddr_in address_sock;
  address_sock.sin_family=AF_INET;
  address_sock.sin_port=htons(4242);
  address_sock.sin_addr.s_addr=htonl(INADDR_ANY);
  int r=bind(sock,(struct sockaddr *)&address_sock,sizeof(struct
sockaddr_in));
  if(r==0){
    r=listen(sock,0);
    while(1){
      struct sockaddr_in caller;
      socklen_t size=sizeof(caller);
      int sock2=accept(sock,(struct sockaddr *)&caller,&size);
      if(sock2>=0){
        printf("Port de l'appelant: %d\n",ntohs(caller.sin_port));
        printf("Adresse de l'appelant: %s\n",inet_ntoa(caller.sin_addr));
      }
      close(sock2);
    }
  }
  return 0;
}
```