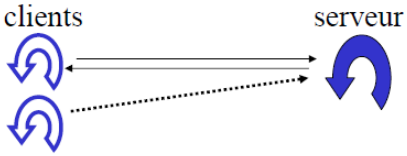
# Modèles de serveurs

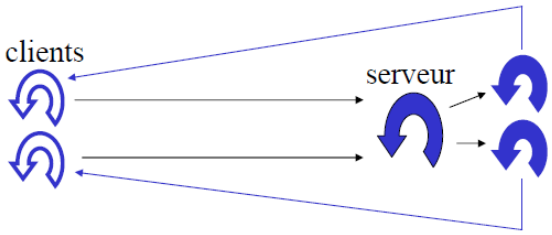
Deux comportements de serveurs et deux protocoles de transport combinés génèrent quatre modèles de serveurs :

| | |
|---|---|
| Itératif Data–gramme | Itératif Connecté |
| Concourant Data–gramme | Concourant Connecté |

- Serveur itératif
  - Le serveur ne traite qu'une demande à la fois




- Serveur parallèle ou concourant
  - Le serveur traite plusieurs demandes simultanément




La terminologie ``**tâche esclave**'' employée dans les algorithmes qui suivent se veut neutre quant au choix technologique retenu pour les implémenter. Ce qui importe c'est leur nature concourante avec la ``**tâche maître**'' qui les engendre.

**Algorithme itératif - Mode datagramme (protocoles IP, UDP) :**

1. Créer une socket, lui attribuer un port connu des clients.
2. Répéter :
   - Lire une requête d'un client,
   - Formuler la réponse,
   - Envoyer la réponse, conformément au protocole d'application.

**Critique :**Cette forme de serveur est la plus simple, elle n'est pas pour autant inutile. Elle est adaptée quand il y a un tout petit volume d'information à échanger et en tout cas sans temps de calcul pour l'élaboration de la réponse. Les serveurs de date ou de temps en sont d'excellents exemples.

**Algorithme Itératif - Mode connecté (protocole TCP) :**

1. Créer une socket, lui attribuer un port connu des clients.
2. Mettre la socket à l'écoute du réseau, en mode passif.
3. Accepter la connexion entrante, obtenir une socket pour la traiter.
4. Entamer le dialogue avec le client, conformément au protocole de l'application.
5. Quand le dialogue est terminé, fermer la connexion et aller en 3).

**Critique :** Ce type de serveur est peu utilisé. Son usage pourrait être dédié à des relations clients/serveurs mettant en jeu de petits volumes d'informations avec la nécessité d'en assurer à coup sûr le transport. Le temps d'élaboration de la réponse doit rester rapide.

Le temps d'établissement de la connexion n'est pas négligeable par rapport au temps de réponse du serveur, ce qui le rend peu attractif.

**Algorithme concourant - Mode datagramme (protocoles IP, UDP) :**

**Maître :**

1. Créer une socket, lui attribuer un port connu des clients.
2. Répéter :
   - Lire une requête d'un client
   - Créer une tâche esclave pour élaborer la réponse.

**Esclave :**

1. Recevoir la demande du client,
2. Élaborer la réponse,
3. Envoyer la réponse au client, conformément au protocole de l'application,
4. Terminer le processus.

**Critique :** Si le temps d'élaboration de la réponse est rendu indifférent pour cause de création de processus esclave, par contre le coût de création de ce processus fils est prohibitif par rapport à son usage : formuler une seule réponse et l'envoyer. Cet inconvénient l'emporte généralement sur l'avantage apporté par le ``parallélisme".

Néanmoins, dans le cas d'un temps d'élaboration de la réponse long par rapport au temps de création du processus esclave, cette solution se justifie.

**Algorithme concourant - Mode connecté (protocoles TCP) :**

**Maître :**

1. Créer une socket, lui attribuer un port connu des clients.
2. Mettre la socket à l'écoute du réseau, en mode passif.
3. Répéter :
   - Accepter la connexion entrante, obtenir une socket pour la traiter,
   - Créer une tâche esclave pour traiter la réponse.

**Esclave :**

1. Recevoir la demande du client,
2. Amorcer le dialogue avec le client, conformément au protocole de l'application,
3. Terminer la connexion et le processus.

**Critique :**

C'est le type le plus général de serveur parce qu'il offre les meilleurs caractéristiques de transport et de souplesse d'utilisation pour le client. Il est surdimensionné pour les ``petits" services et sa programmation soignée n'est pas toujours à la portée du programmeur débutant.