

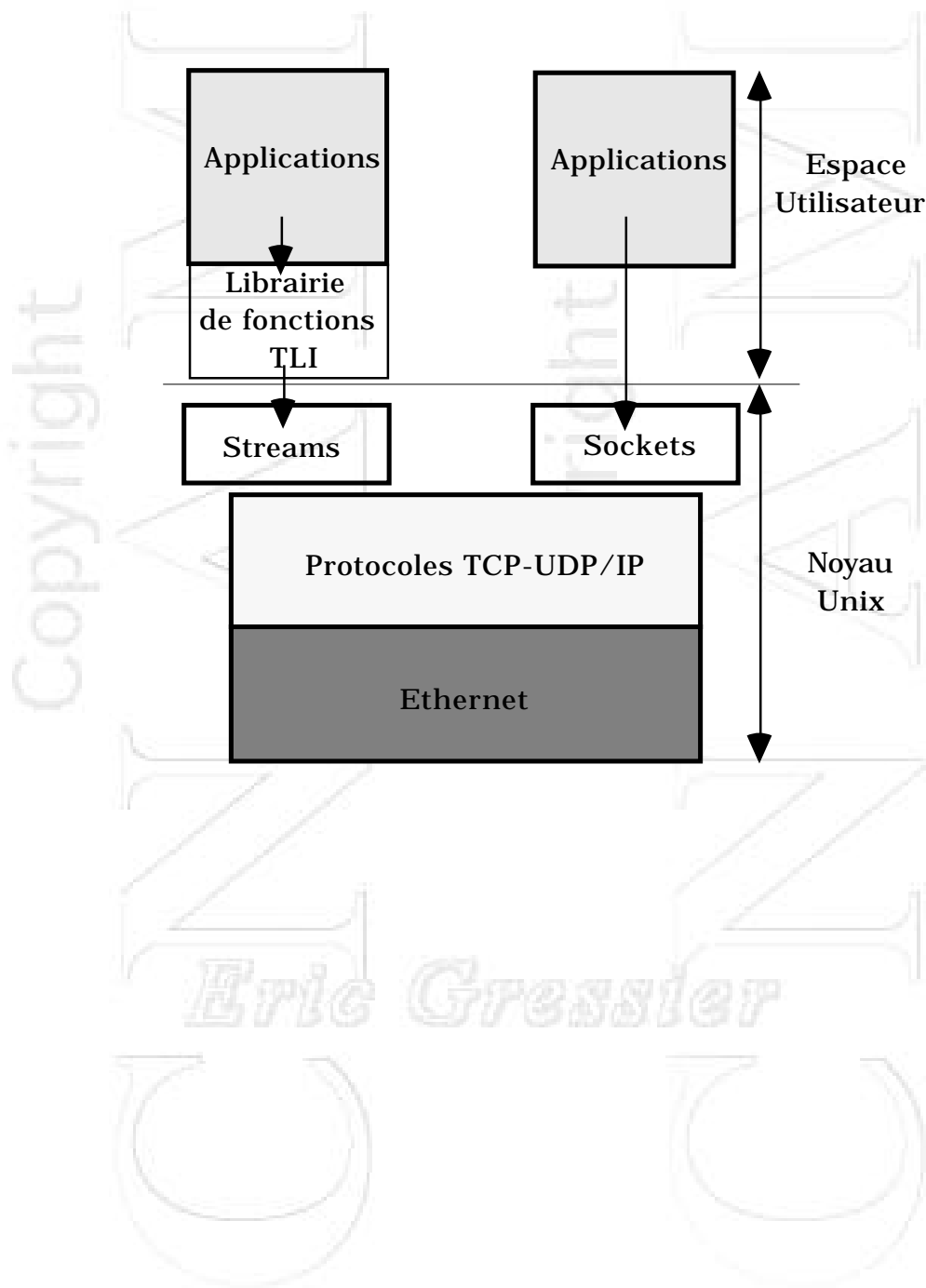
6. API TLI

Copyright

Copyright

Eric Gressier

Les Sockets : Interface avec les protocoles de communication



TLI et les Streams

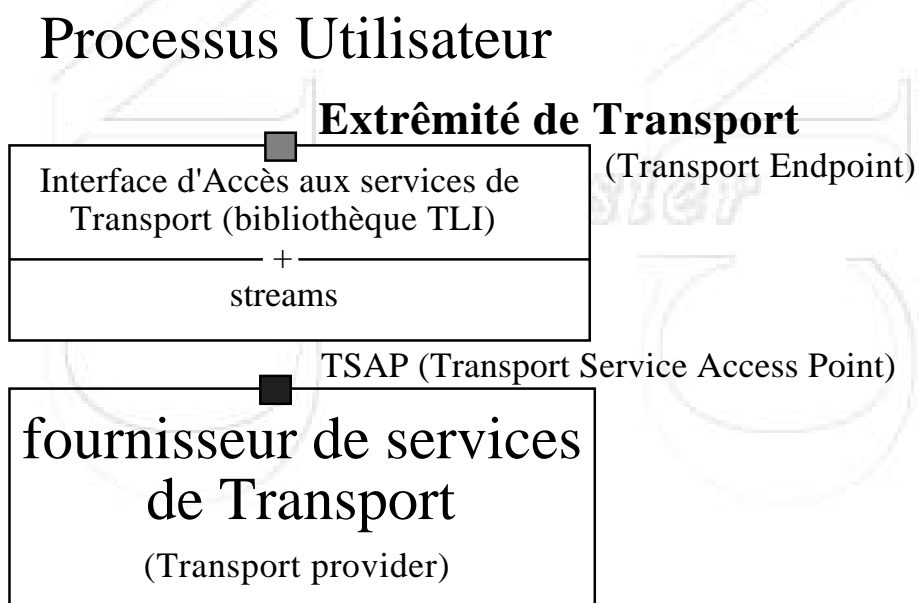
TLI pour Transport Layer Interface, c'est une bibliothèque de fonctions qui sert à accéder des protocoles de communication ... attention, ce ne sont pas les protocoles eux-mêmes !!!!

Pour les utiliser, il faut appeler une bibliothèque à l'édition de liens :

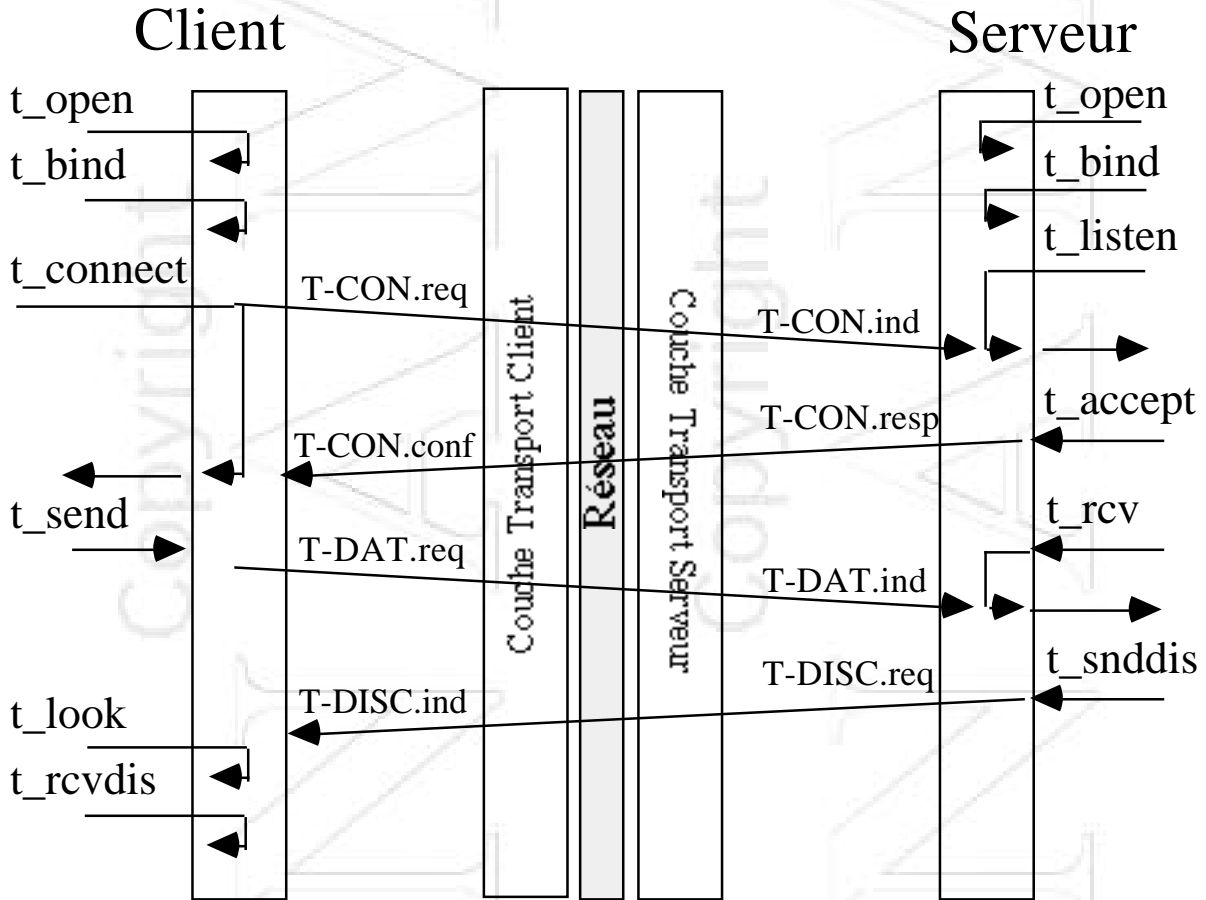
```
cc prog.c -lnsl_s
```

TLI est apparu avec Unix System V release 3.0. (ATT) et doit être vu comme une API pour accéder à la couche Transport d'un réseau de communication.

La version normalisée par le groupe X/Open est : XTI. TLI est donc très marqué par la logique ISO :



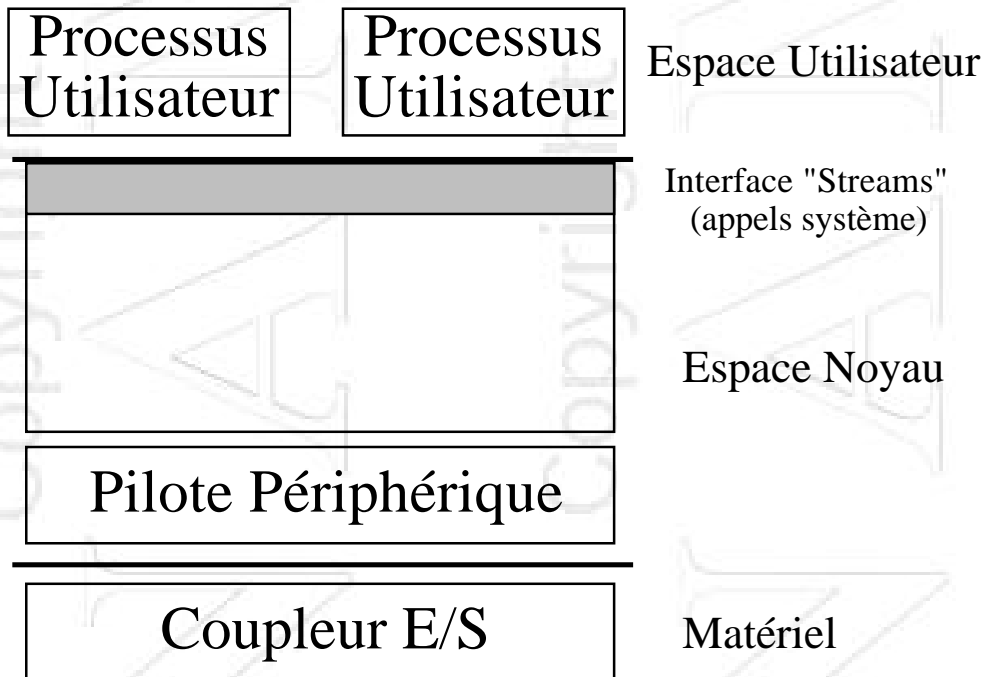
Utilisation de TLI dans l'univers ISO



Ca marche aussi avec TCP/IP !!!

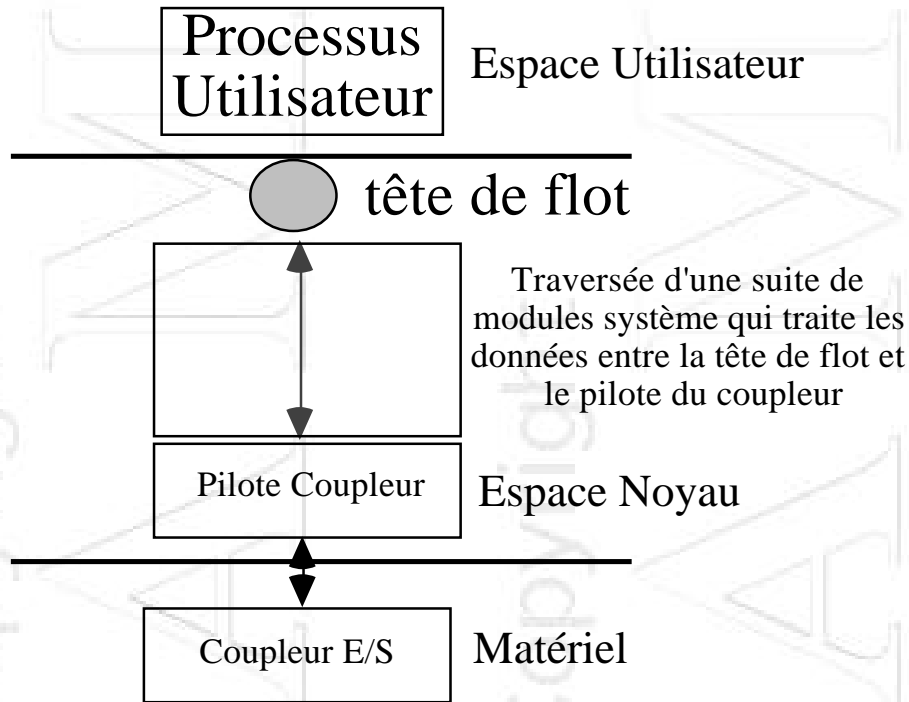
STREAMS

Modèle abstrait :



Streams : image du flot de données établi entre l'utilisateur et un périphérique d'E/S via son "driver"

Streams : vue système



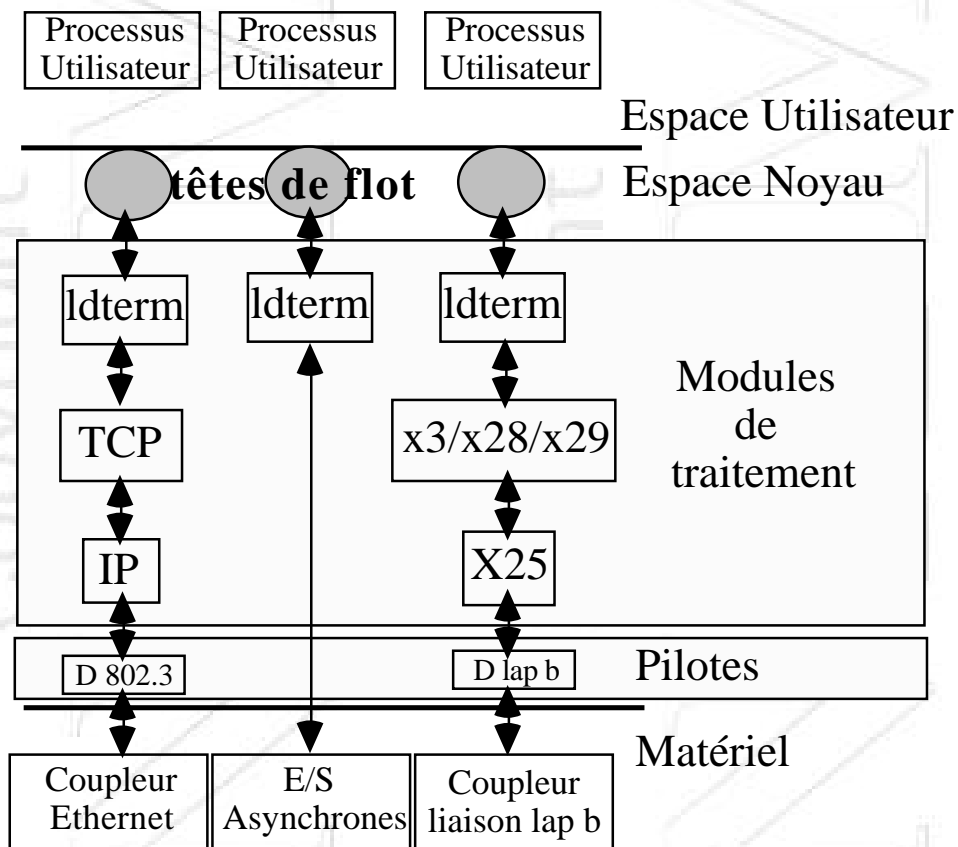
Les flots SVR4¹⁰ représentent un **mécanisme système générique** pour implanter des politiques de gestion de périphériques : réseau, terminal...

L'objectif est de pouvoir développer du code de façon modulaire et réutilisable. Pour cela on peut insérer des "modules de traitement" entre la tête de flot et le pilote de périphérique.

¹⁰ On parlera de flots SVR4 pour Streams system V R4.0, même si le mécanisme de streams existe dans des versions antérieures de System V.

Streams : Vue système - Exemple (1)

Streams et Contrôle d'E/S Terminal : Chaque processus utilisateur pense être connecté à une ligne de terminal

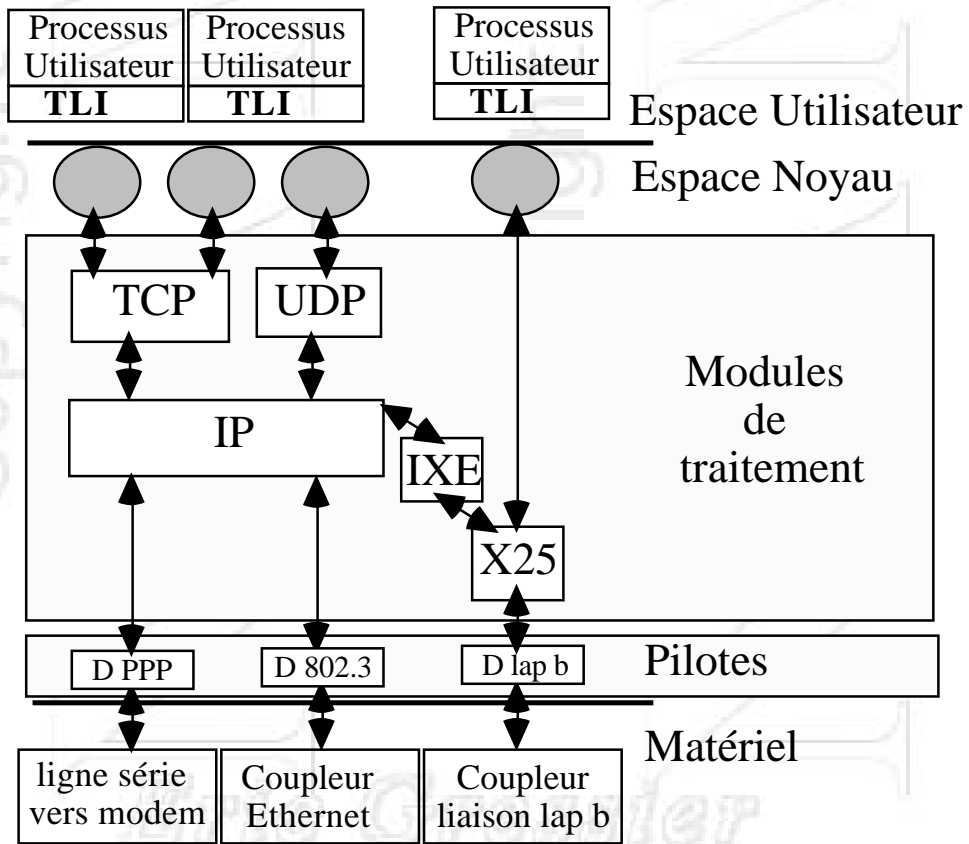


- Une liaison terminal emprunte TCP/IP sur un réseau local Ethernet.
- Une liaison terminal emprunte une ligne série asynchrone.
- Une liaison terminal emprunte une connexion X25.

Exemple de réutilisation du module ldterm

Streams : Vue système - Exemple (2)

Streams et Communication TCP/IP



Avec ce type d'architecture, on peut imaginer que chaque module peut être fourni par un constructeur différent.

TLI : bibliothèque de fonctions

t_open : création d'une extrémité de transport qui identifie le protocole de Transport utilisé et la qualité de service demandée, l'extrémité de Transport est vue comme un fichier

t_bind : association d'une adresse (TSAP en OSI) à l'extrémité de transport

t_unbind : contraire de t_bind

t_look : permet de connaître l'état courant associé à l'extrémité de connexion

t_getstate : permet de connaître l'évènement courant associé à l'extrémité de connexion

t_getinfo : permet de connaître les paramètres de l'extrémité de transport qui ont pu changer après l'acceptation d'une ouverture de connexion

t_optmgmt : modification des paramètres d'un protocole

t_listen : demande d'attente d'une requête de connexion

t_accept : acceptation d'une demande d'ouverture de connexion, une nouvelle extrémité de transport peut être fournie par l'utilisateur, au retour de l'appel système cette extrémité servira aux échanges de données ultérieurs, on peut spécifier la même extrémité de transport que celle qui a servi à accepter la connexion

t_snd/t_rcv : émettre/recevoir en mode connecté

t_sndrel : demande de fermeture de connexion sans perte de données

t_rcvrel : indication de fermeture de connexion sans perte de données

t_sndis : demande de fermeture de connexion abrupte ou refus d'ouverture de connexion

t_rcvdis : indication de fermeture de connexion abrupte ou attente de rejet de connexion

t_sndudata/t_rcvudata : émettre/recevoir en mode non connecté

t_rcvuerr : réception d'erreur pour l'émission/réception de données en mode non connecté

t_close : fermeture de connexion avec libération des ressources associées à l'extrémité de transport

Etats d'une extrêmité de transport

T_UNINIT : avant création

T_UNBND : créé mais pas d'adresse associée

T_IDLE : en attente de données ou de connexion

T_INCON : ouverture de connexion passive (serveur)

T_OUTCON : ouverture de connexion active (client)

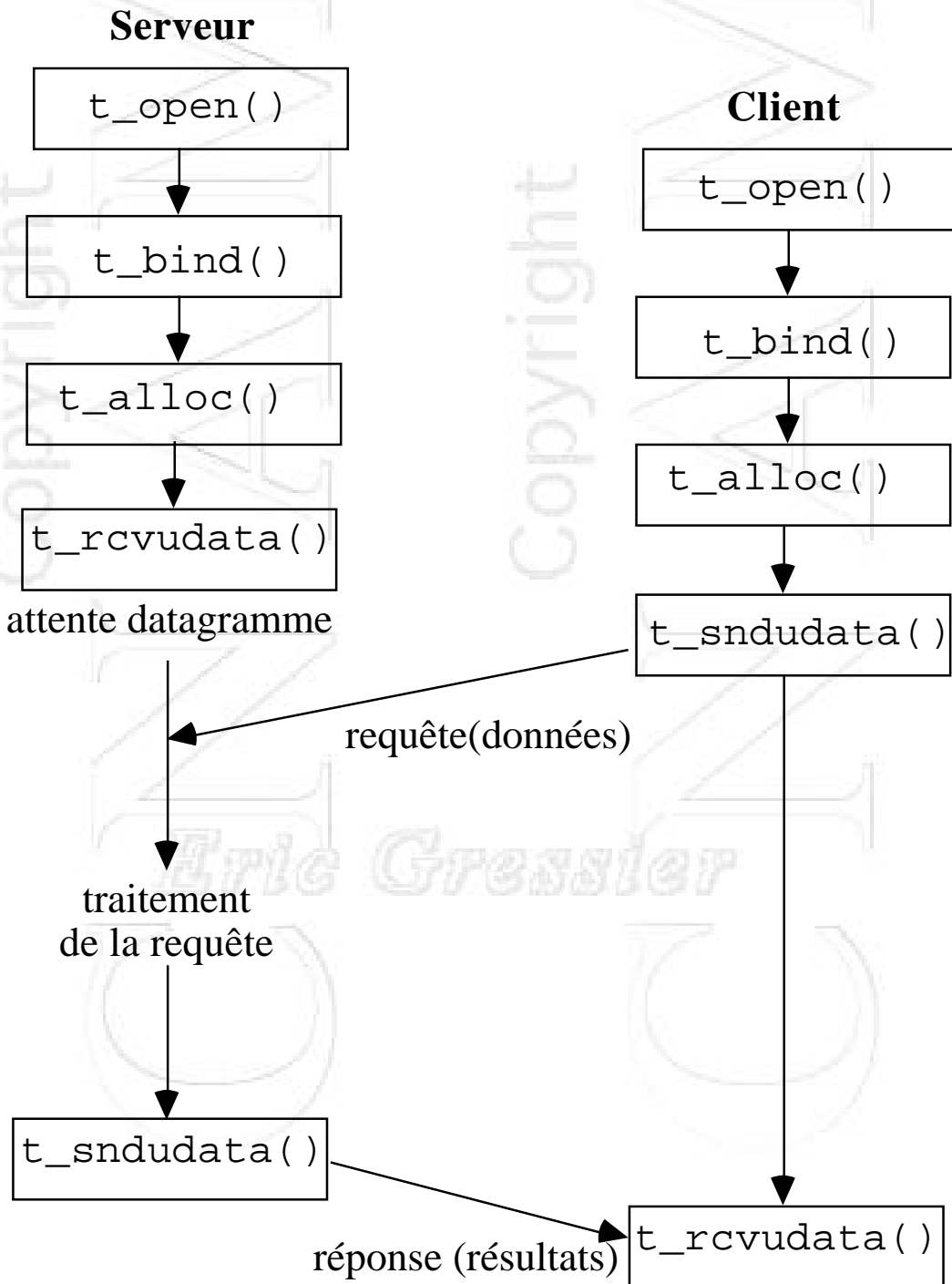
T_DATAXFER : transfert de données possible sur une extrêmité de transport avec connexion établie

T_INREL : fermeture de connexion ordonnée reçue

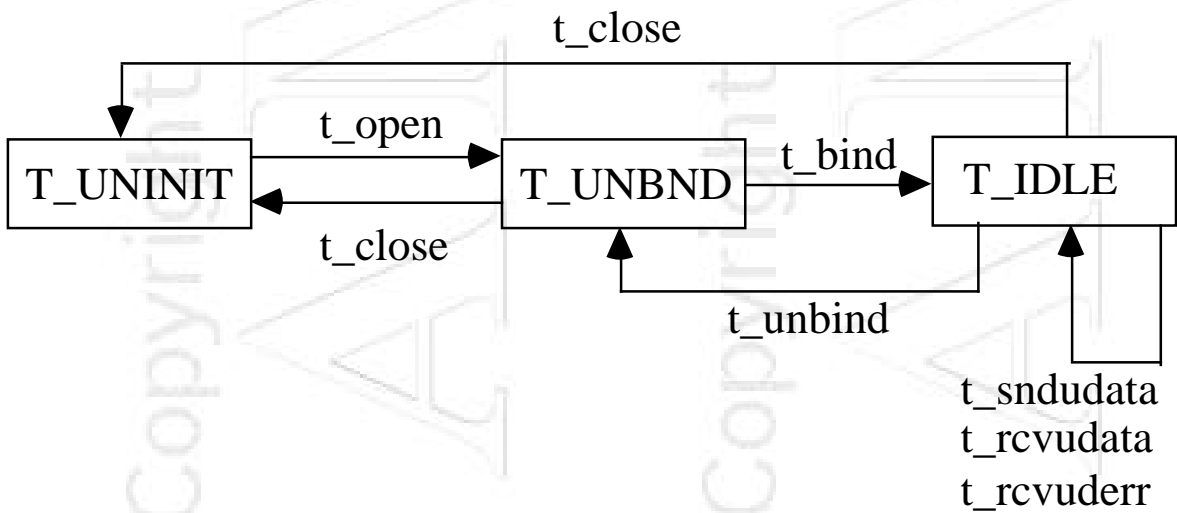
T_OUTREL : fermeture de connexion ordonnée demandée

L'état d'une extrêmité de transport autorise l'utilisateur à se servir de certaines fonctions de la bibliothèque TLI. Ces fonctions en retour modifient l'état de l'extrêmité de transport.

TLI : Client-Serveur en mode non connecté

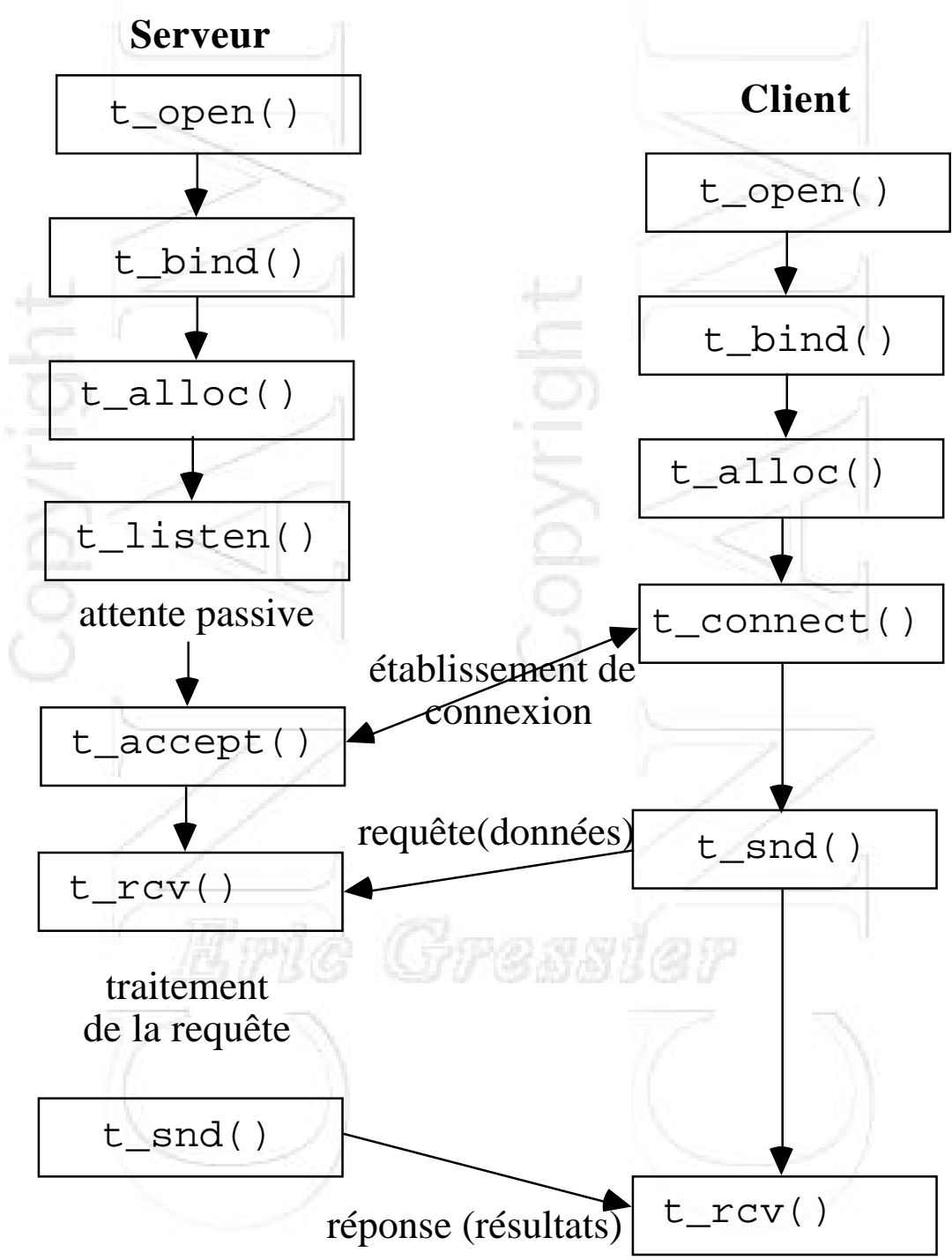


Automate de changement d'états d'une extrêmité de transport pour le mode non connecté¹¹



¹¹ source : cours IIE-CNAM sur l'interface TLI de Laurence Duchien. 1995.

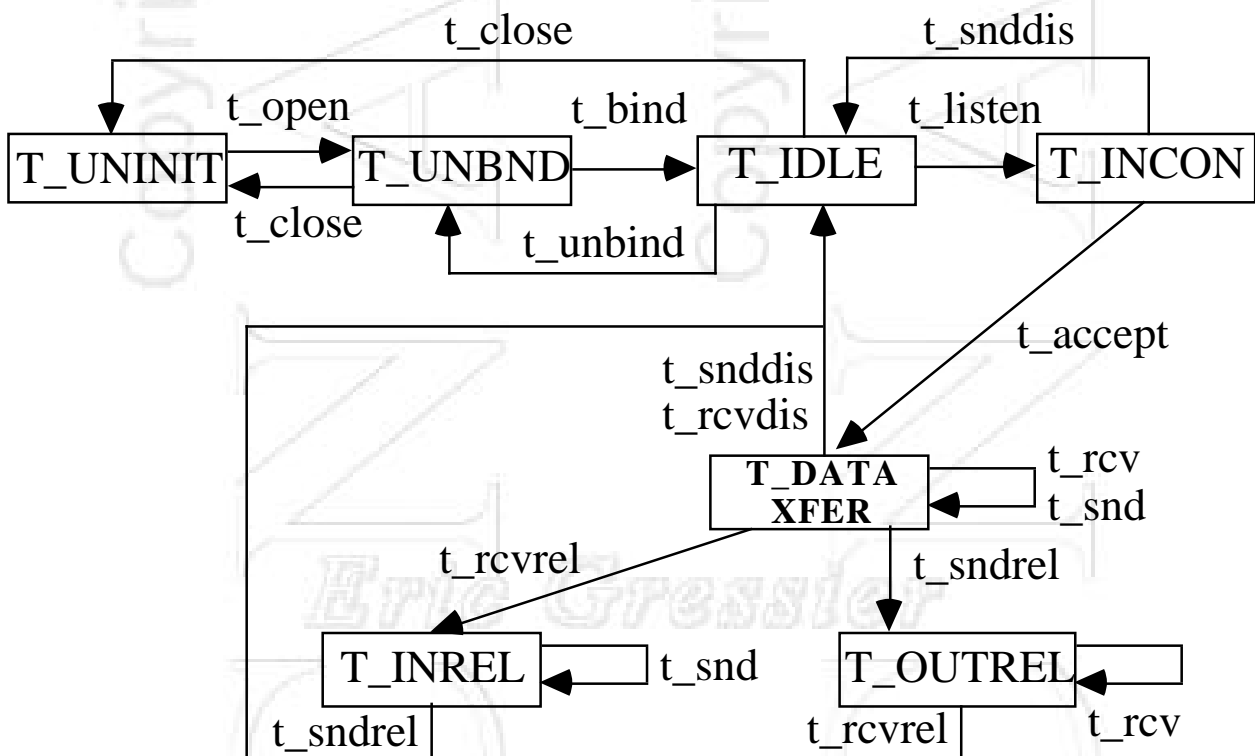
TLI : Client-Serveur en mode connecté



Automate de changement d'états d'une extrêmité de transport pour le mode connecté, côté serveur ¹² (1)

Le `t_connect()` ne se comporte pas comme le `connect()` de l'interface socket même s'il a une fonction identique, il y a deux possibilités :

a) On continue le dialogue sur l'extrêmité de transport qui a servi à écouter

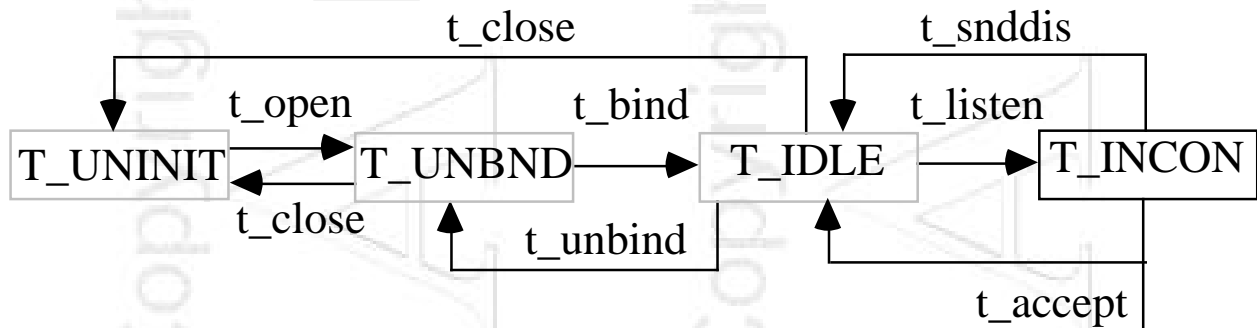


¹² source : cours IIE-CNAM sur l'interface TLI de Laurence Duchien. 1995.

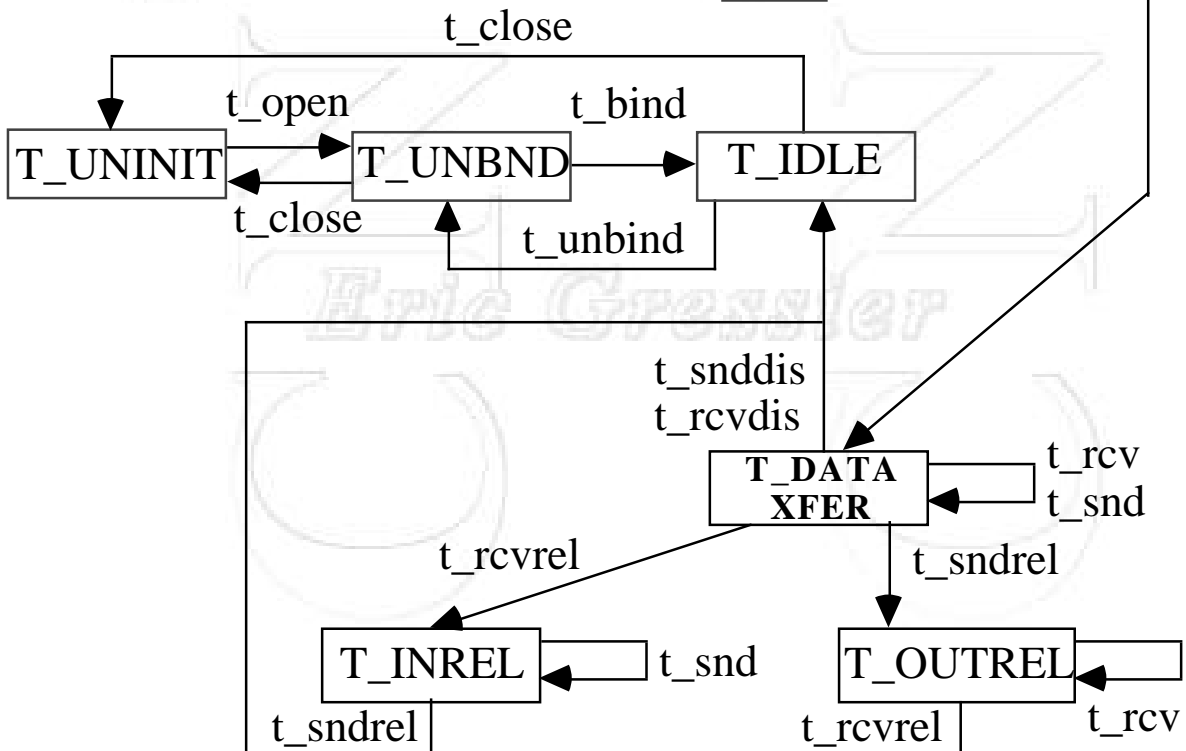
Automate de changement d'états d'une extrêmité de transport pour le mode connecté, côté serveur (2)

b) On continue le dialogue sur une autre extrêmité préalablement créée et initialisée avec une adresse de transport, la première extrêmité est l'extrêmité d'écoute, la seconde est l'extrêmité de dialogue

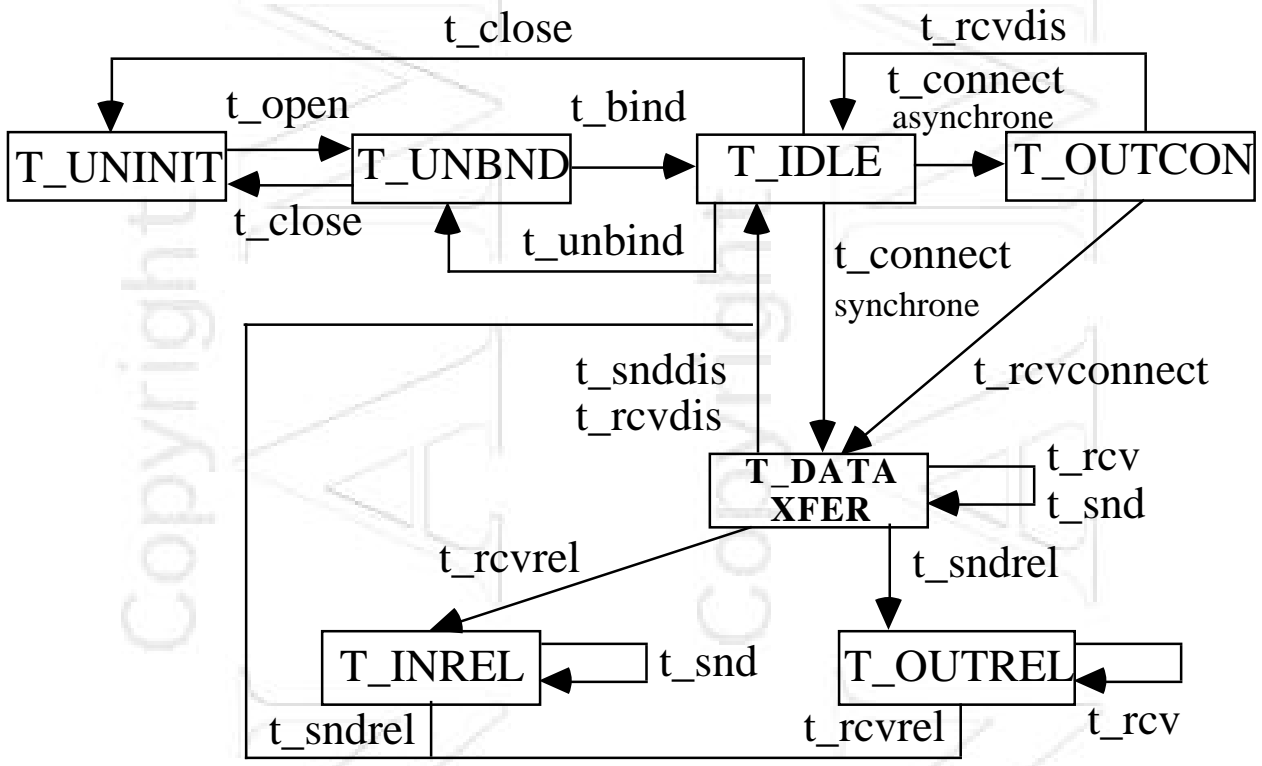
Extrêmité d'écoute :



Extrêmité de dialogue préalablement initialisée :



Automate de changement d'états d'une extrémité de transport pour le mode connecté, côté client ¹³



Eric Gressier

¹³ source : cours IIE-CNAM sur l'interface TLI de Laurence Duchien. 1995.
 3 Mars 1994 102

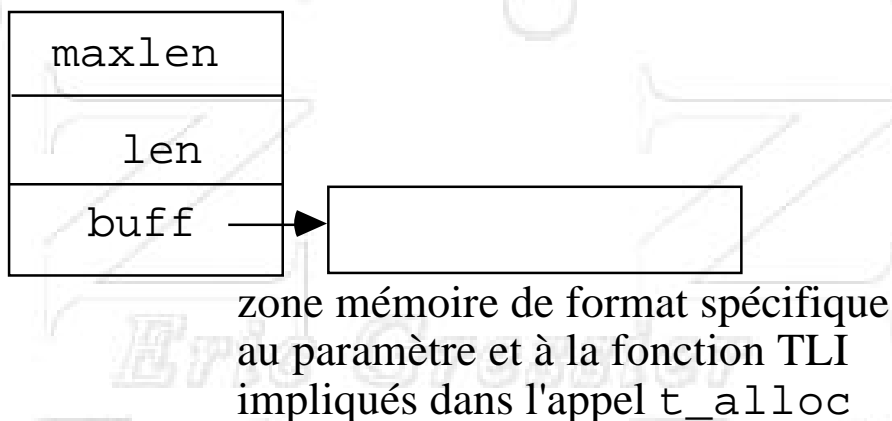
TLI - Primitives d'allocation de ressources

Deux primitives ont un rôle particulier : allocation/désallocation de ressources mémoire pour l'échange de données entre le processus utilisateur et la tête de flot via la bibliothèque TLI (penser à la notion d'IDU du modèle ISO-OSI)

`t_alloc()` et `t_free()`

Les buffers qui traversent l'interface ont un format spécifique à chaque protocole, mais aussi à chaque fonction de la bibliothèque, ils sont composés d'une ou plusieurs structures de type `netbuf` :

`struct netbuf` :



t_alloc() (1)

```
#include <tiuser.h>
char *t_alloc(int fd, int structtype, int fields);
```

- fd est le résultat retourné par l'appel à t_open()

- structtype peut être :

T_BIND pour la fonction t_bind()

T_DIS pour les fonctions t_rcvdis()

T_INFO pour la fonction t_getinfo()

T_OPTMGMT pour la fonction t_optmgmt()

T_UNITDATA pour les fonctions t_*udata()

T_UDERROR pour la fonction t_rcvuderr()

T_CALL pour les autres fonctions qui nécessite une structure de données de type t_call

- fields permet le contrôle des buffers alloués, plus facile T_ALL