# Le réseau vu du langage de programmation : quelle API pour le réseau ?

*Stéphane Bortzmeyer*
*AFNIC*
*bortzmeyer@nic.fr*

# Le réseau vu du langage de programmation : quelle API pour le réseau ?

*Stéphane Bortzmeyer*
*AFNIC*
*bortzmeyer@nic.fr*

afnic

# Terminologie

# Terminologie

- API = *Application Programming Interface*

# Terminologie

- API = *Application Programming Interface*
- Bibliothèque : **une** mise en œuvre particulière d'une API

afnic

# Terminologie

- API = *Application Programming Interface*
- Bibliothèque : **une** mise en œuvre particulière d'une API
- L'API, c'est ce que la · e programmeur · e doit lire pour utiliser une bibliothèque

afnic

# Terminologie

- API = *Application Programming Interface*
- Bibliothèque : **une** mise en œuvre particulière d'une API
- L'API, c'est ce que la · e programmeur · e doit lire pour utiliser une bibliothèque
- Protocole : ce qui est circule sur le réseau. Planqué dans le code de la bibliothèque et documenté par exemple dans un RFC.

afnic

# La situation aujourd'hui

# La situation aujourd'hui

- Toutes ( ?) les applications utilisent le réseau,

# La situation aujourd'hui

- Toutes ( ?) les applications utilisent le réseau,
- Et dépendent donc d'une API réseau,

afnic

# La situation aujourd'hui

- Toutes ( ?) les applications utilisent le réseau,
- Et dépendent donc d'une API réseau,
- Il n'y a pas d'API parfaite, toutes ont de sérieuses limites.

afnic

# Avec un framework qui cache presque tout

Python + Quart (compatible avec Flask) :

```
@app.route('/', methods=['HEAD', 'GET', 'POST'])
async def index():
   if request.method == 'POST':
      ct = request.headers.get('content-type')
      if ct != "application/dns-udpwireformat":
          abort(415)
      data = await request.get_data()
      r = bytes(data)
   else:
      abort(405)
   return (response,
           {'Content-Type': 'application/dns-udpwireformat',
            'Cache-Control': 'no-cache'})
```

afnic

# Exemple simple

Une application Java veut récupérer un fichier :

```
String url = args[0];
InputStream is = new URL(url).openStream();
try {
    BufferedReader rd = new BufferedReader(new InputStreamReader(is));
    StringBuilder sb = new StringBuilder();
    int cp;
    while ((cp = rd.read()) != -1) {
        sb.append((char) cp);
    }
    text = sb.toString();
} finally {
    is.close();
}
System.out.println(text);
```

afnic

# On dépend bien sûr du langage

Et de sa bibliothèque standard (ici, Go) :

```
import (
    "http"
)

url string = "http://data.keolis-rennes.com/xml/"

completeurl := url + "?version=1.0&key=" + http.URLEscape(key) +
    "&cmd=getstation&param[request]=number&param[value]=" +
    http.URLEscape(query)
response, _, error := http.Get(completeurl)
body := make([]byte, maxsize)
n, _ := response.Body.Read(body)
```

afnic

# C avec libcurl

Si pas de bibliothèque standard, plus de code portable (libcurl vs. neon)

```
char            data[MAX_FILE_SIZE];
char           *data_ptr = data;
size_t
write_data(void *inputbuffer, size_t size, size_t sizescale, void *user
{
        size_t          nbytes = size * sizescale;
        strncpy(data_ptr, inputbuffer, nbytes);
        data_ptr += nbytes;
        written += nbytes;
        return nbytes;
}
curl_easy_setopt(curl, CURLOPT_ERRORBUFFER, errbuf);
curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, write_data);
curl_easy_setopt(curl, CURLOPT_URL, url);
result = curl_easy_perform(curl);
/* Do something with data */
```

afnic

# API socket

Bitcoin Core, écrit en C++ :

```
bool ConnectSocketDirectly(const CService &addrConnect, const SOCKET& h
    bool manual_connection)
{
    struct sockaddr_storage sockaddr;
    socklen_t len = sizeof(sockaddr);
    if (connect(hSocket, (struct sockaddr*)&sockaddr, len) == SOCKET_ER
    {
        fd_set fdset;
        FD_ZERO(&fdset);
        FD_SET(hSocket, &fdset);
        int nRet = select(hSocket + 1, nullptr, &fdset, nullptr, &timeo
```

# Problèmes et limites des API

# Problèmes et limites des API

- Trop haut niveau : empêche d'utiliser certaines fonctions du réseau, « fuites » des problèmes réseau

afnic

# Problèmes et limites des API

- Trop haut niveau
- Trop bas niveau : trop compliqué pour l · e · a programmeur · e, trop dangereux pour la sécurité - exemple d'OpenSSL

afnic

# Problèmes et limites des API

- Trop haut niveau
- Trop bas niveau
- Si on veut juste récupérer une ressource en HTTP(S), c'est bon

afnic

# Problèmes et limites des API

- Trop haut niveau
- Trop bas niveau
- Si on veut juste récupérer une ressource en HTTP(S), c'est bon
- Mais d'autres applications sont plus ambitieuses !

afnic

# La bibiothèque n'essaie qu'une seule adresse

Mastodon est écrit en Ruby :

tootsuite / **mastodon**

Watch 529

Code | Issues 874 | Pull requests 79 | Projects 0 | Insights

## Issue connecting to dual-stack instances #3762

Open **mherrb** opened this issue on Jun 15, 2017 · 16 comments

**mherrb** commented on Jun 15, 2017

It looks like that when mastodon instances like mastodon.rocks publish an IPv6 address (via a DNS AAAA record) but isn't reachable on it, other IPv6 enabled instances can't communicate with it, even though the legacy IPv4 protocol still works on both.

The IPv6 connection timeouts and doesn't fall back to IPv4

- [x] I searched or browsed the repo's other issues to ensure this is not a duplicate.
- [x] This bug happens on a tagged release and not on `master` (If you're a user, don't worry about this).

**Shnoulle** commented on Jun 15, 2017

But : it's an instance issue , not a mastodon issue.

# Ce qu'attend l'application

# Ce qu'attend l'application

- L'application attend du réseau des **services**. Exemple : délivrer les données intactes et dans l'ordre. . .

# Ce qu'attend l'application

- L'application attend du réseau des **services**
- Chaque application est différente. Elles n'ont pas toutes besoin des mêmes services.

afnic

# Ce qu'attend l'application

- L'application attend du réseau des **services**
- Chaque application est différente
- Les API de trop haut niveau limitent en général sévèrement les services (options réseau, barre de progression...)

afnic

# Ce qu'attend l'application

- L'application attend du réseau des **services**
- Chaque application est différente
- Les API de trop haut niveau limitent en général sévèrement les services
- Les API de trop bas niveau obligent à se taper des détails non-pertinents. (Faire de l'UpnP ou du PCP, utiliser STUN ou TURN, se déconnecter/reconnecter...)

afnic

# Ce qu'attend l'application

- L'application attend du réseau des **services**
- Chaque application est différente
- Les API de trop haut niveau limitent en général sévèrement les services
- Les API de trop bas niveau obligent à se taper des détails non-pertinents.
- Sans API standard, pas de portabilité des applications.

afnic

# Fonctions possibles

# Fonctions possibles

- Délivrance des données fiable (transfert de fichiers) ou « au mieux » (vidéo en direct)

# Fonctions possibles

- Délivrance des données fiable ou « au mieux »
- Structuration en messages, ou en flot continu d'octets

# Fonctions possibles

- Délivrance des données fiable ou « au mieux »
- Structuration en messages, ou en flot continu d'octets
- Confidentialité

afnic

# Fonctions possibles

- Délivrance des données fiable ou « au mieux »
- Structuration en messages, ou en flot continu d'octets
- Confidentialité
- Gestion explicite ou implicite des adresses IP multiples (*multipath*)

afnic

# Fonctions possibles

- Délivrance des données fiable ou « au mieux »
- Structuration en messages, ou en flot continu d'octets
- Confidentialité
- Gestion explicite ou implicite des adresses IP multiples
- Gestion de la congestion

afnic

# Fonctions possibles

- Délivrance des données fiable ou « au mieux »
- Structuration en messages, ou en flot continu d'octets
- Confidentialité
- Gestion explicite ou implicite des adresses IP multiples
- Gestion de la congestion
- Privilégier la latence ou le débit ?

afnic

# Fonctions possibles

- Délivrance des données fiable ou « au mieux »
- Structuration en messages, ou en flot continu d'octets
- Confidentialité
- Gestion explicite ou implicite des adresses IP multiples
- Gestion de la congestion
- Privilégier la latence ou le débit ?
- Décroissance : « Je ne suis pas pressé, je laisse la place aux autres »

afnic

# Fonctions possibles

- Délivrance des données fiable ou « au mieux »
- Structuration en messages, ou en flot continu d'octets
- Confidentialité
- Gestion explicite ou implicite des adresses IP multiples
- Gestion de la congestion
- Privilégier la latence ou le débit ?
- Décroissance : « Je ne suis pas pressé, je laisse la place aux autres »
- Un ensemble cohérent de **fonctions** est un **service**

afnic

# L'API socket

# L'API socket

- « Responsable du succès de l'Internet » ?

afnic

# L'API socket

- « Responsable du succès de l'Internet » ?
- À première vue, permet de spécifier un service (`SOCK_STREAM`)

afnic

# L'API socket

- « Responsable du succès de l'Internet » ?
- À première vue, permet de spécifier un service (`SOCK_STREAM`)
- Mais en fait non (si on veut juste un flot de données, cela pourrait être fait avec TCP ou SCTP, mais seul TCP sera sélectionné)

afnic

# L'API socket

- « Responsable du succès de l'Internet » ?
- À première vue, permet de spécifier un service (`SOCK_STREAM`)
- Mais en fait non (si on veut juste un flot de données, cela pourrait être fait avec TCP ou SCTP, mais seul TCP sera sélectionné)
- Pas assez abstraite

afnic

# Les réalités du réseau

# Les réalités du réseau

- Le réseau n'est pas un fichier

# Les réalités du réseau

- Le réseau n'est pas un fichier
- Le réseau plante (souvent ?)

# Les réalités du réseau

- Le réseau n'est pas un fichier
- Le réseau plante (souvent ?)
- Plusieurs fonctions différentes sont possibles sur le réseau (par exemple avec ou sans confidentialité)

afnic

# En dessous de l'API : les protocoles

# En dessous de l'API : les protocoles

- Un protocole est la réalisation d'un ou plusieurs service(s)

afnic

# En dessous de l'API : les protocoles

- Un protocole est la réalisation d'un ou plusieurs service(s)
- Plusieurs protocoles peuvent réaliser un service donné

afnic

# En dessous de l'API : les protocoles

- Un protocole est la réalisation d'un ou plusieurs service(s)
- Plusieurs protocoles peuvent réaliser un service donné
- TCP :
  - Délivrance d'octets (pas de structuration)
  - Fiable et dans l'ordre
  - Contrôle de congestion
  - Aucune sécurité

afnic

# En dessous de l'API : les protocoles

- Un protocole est la réalisation d'un ou plusieurs service(s)
- Plusieurs protocoles peuvent réaliser un service donné
- TCP :
  - Délivrance d'octets
  - Fiable et dans l'ordre
  - Contrôle de congestion
  - Aucune sécurité
- UDP :
  - Délivrance de messages
  - Aucune garantie : peuvent se perdre, se doubler...
  - Aucun contrôle de congestion
  - **Très dangereux** : ne vous croyez pas capable de réimplémenter TCP

afnic

# En dessous de l'API : les protocoles

- Un protocole est la réalisation d'un ou plusieurs service(s)
- Plusieurs protocoles peuvent réaliser un service donné
- TCP :
  - Délivrance d'octets
  - Fiable et dans l'ordre
  - Contrôle de congestion
  - Aucune sécurité
- UDP :
  - Délivrance de messages
  - Aucune garantie : peuvent se perdre, se doubler...
  - Aucun contrôle de congestion
  - **Très dangereux** : ne vous croyez pas capable de réimplémenter TCP
- UDP-Lite (UDP mais avec corruption possible), DCCP (UDP mais avec contrôle de congestion)

afnic

# Les protocoles, suite

# Les protocoles, suite

- SCTP :
  - Délivrance de messages
  - Fiable et dans l'ordre
  - Contrôle de congestion
  - Plein d'options en plus de TCP, notamment de gestion des multi-adresses

afnic

# Les protocoles, suite

- SCTP :
  - Délivrance de messages
  - Fiable et dans l'ordre
  - Contrôle de congestion
  - Plein d'options en plus de TCP
- QUIC (le nouveau TCP au dessus d'UDP, made in Google)

afnic

# Les protocoles, suite

- SCTP :
  - Délivrance de messages
  - Fiable et dans l'ordre
  - Contrôle de congestion
  - Plein d'options en plus de TCP
- QUIC (le nouveau TCP au dessus d'UDP, made in Google)
- TLS :
  - Oui, vu de l'application, c'est un protocole de transport
  - Délivrance d'octets
  - Fiable et dans l'ordre
  - Contrôle de congestion (grâce à TCP en dessous)
  - Confidentialité et authentification
  - Pas d'API standard

afnic

# Les protocoles, fin

# Les protocoles, fin

- LEDBAT : protocole humble et poli, cède toujours sa place à TCP

# Les protocoles, fin

- LEDBAT : protocole humble et poli, cède toujours sa place à TCP
- HTTP :
  - Oui, vu de l'application, c'est un protocole de transport
  - Souvent utilisé quand on n'a pas le choix

afnic

# Du service au protocole

# Du service au protocole

- Une fois défini le service, plusieurs protocoles peuvent convenir

# Du service au protocole

- Une fois défini le service, plusieurs protocoles peuvent convenir
- C'est au programmeur de choisir, il n'y a pas d'API qui permette cela

afnic

# Du service au protocole

- Une fois défini le service, plusieurs protocoles peuvent convenir
- C'est au programmeur de choisir, il n'y a pas d'API qui permette cela
- Exemple, si une application veut que ses octets soient distribués dans l'ordre, et sans pertes, plusieurs protocoles peuvent convenir (TCP, SCTP, TLS, HTTP)

afnic

# Le groupe de travail TAPS

# Le groupe de travail TAPS

- Groupe de travail IETF chargé de définir rigoureusement les services de la couche 4 (couche Transport)

afnic

# Le groupe de travail TAPS

- Groupe de travail IETF chargé de définir rigoureusement les services de la couche 4
- Lisez son RFC 8303, ça vaut la peine

afnic

# Le groupe de travail TAPS

- Groupe de travail IETF chargé de définir rigoureusement les services de la couche 4
- Lisez son RFC 8303, ça vaut la peine
- Des travaux en cours : définir un jeu minimum de services et les services de sécurité

afnic

# Vers une nouvelle API ?

# Vers une nouvelle API ?

- Faire une API indépendante du langage ?

afnic

# Vers une nouvelle API ?

- Faire une API indépendante du langage ?
- Non. (Exceptions vs. codes de retour, asynchronisme vs. parallélisme. . .)

afnic

# Vers une nouvelle API ?

- Faire une API indépendante du langage ?
- Non. (Exceptions vs. codes de retour, asynchronisme vs. parallélisme. . .)
- Plusieurs projets en cours, par exemple `draft-trammel-taps-interface` (ex-PostSockets) ou NEAT `draft-fairhurst-taps-neat`

afnic

# Merci !

www.afnic.fr
contact@afnic.fr