Comment marche le fédivers

Stéphane Bortzmeyer
AFNIC
1, rue Stephenson
78180 Montigny-le-Bretonneux

Résumé

Le « fédivers » (contraction de « fédération » et « univers ») est l'ensemble des réseaux sociaux décentralisés qui sont gérés indépendamment mais échangent des messages via les protocoles ActivityPub et analogues. C'est un des grands succès des deux dernières années, avec de nombreux logiciels largement utilisés, Mastodon, Pleroma, PeerTube, FunkWhale, WriteFreely, PixelFed… Ces réseaux sociaux décentralisés sont un élément important de la lutte contre les GAFA, cette poignée de grosses entreprises qui tentent de monopoliser la communication sur l'Internet.

Comment fonctionne ce fédivers ? On lit souvent « ces logiciels utilisent tous ActivityPub ». Mais ActivityPub n'est en fait qu'une petite partie des protocoles nécessaires pour faire tourner le fedivers. Il faut toute une suite de protocoles, qui sont présentés ici. Comme souvent sur l'Internet, la théorie est une chose, la pratique une autre, et on verra aussi qu'il faut parfois expérimenter pour comprendre, ici avec une nouvelle mise en œuvre de ces protocoles, le logiciel Tonola.

Mots-clefs

ActivityPub, ActivityStreams, réseaux sociaux décentralisés, Mastodon, Pleroma, PeerTube, WebFinger

1 Pourquoi le fédivers

L'Internet est un réseau pair à pair dès sa conception. En théorie, lorsque la célèbre Alice parle au non moins fameux Bob, la communication va directement[1] de la machine d'Alice à celle de Bob. De nombreux protocoles suivent ce principe, comme le courrier électronique. À noter une nuance importante : certains de ces protocoles sont « presque pair à pair » en ce sens qu'ils introduisent une division entre les machines purement clientes, celles d'Alice et Bob, qui peuvent être éteintes, injoignables, bloquées derrière un pare-feu qui interdit les connexions entrantes, et des serveurs intermédiaires, qui sont, eux, joignables en permanence. Ces intermédiaires isolent Alice et Bob de certaines difficultés de l'Internet (les connexions malveillantes) et de certains problèmes d'administration système. Mais ils ajoutent une entité supplémentaire en qui il faut avoir confiance. Le courrier électronique fonctionne comme cela depuis de nombreuses années, et on verra que le fédivers le fait également.

1. « Directement », du point de vue de la couche 7 « Application ». Car, évidemment, il y a plusieurs routeurs sur le trajet, donc la couche 3 ne pense pas que c'est une communication directe.

Mais les dernières années ont vu une évolution vers un modèle à serveur unique[2]. Lorsqu'Alice et Bob sont sur Facebook, il n'y a plus de pair à pair : toute communication passe par Facebook, qui peut à loisir capter des données personnelles, censurer, masquer plus ou moins les informations qui ne sont pas rentables pour les actionnaires de Facebook. Il y a actuellement beaucoup de débats sur les meilleures méthodes pour limiter les pouvoirs de Facebook, et autres réseaux sociaux centralisés. Quelles que soient les solutions adoptées, il faut être conscient que le problème fondamental est que l'architecture de ces systèmes centralisés est mauvaise politiquement : lorsqu'on concentre tant de pouvoir entre les mains d'un seul acteur, il va forcément en abuser, quelles que soient les régulations qu'on met sur son chemin. Il est donc essentiel de faire également un effort sur l'architecture de nos outils de communication, et de les **décentraliser**. Cela n'a rien d'extraordinaire ou de nouveau, on a vu que des réseaux sociaux décentralisés existent depuis les débuts de l'Internet[3].

2 Côté utilisateur

À quoi ressemble le fédivers, côté utilisateur ? Alice ou Bob doivent se choisir une ou plusieurs **instances**, qui sont les composants de base du fédivers. Une instance se caractérise par les services offerts (certaines instances sont spécialisées dans le micro-blogging, à la Twitter, d'autres dans les photos, à la Instagram, etc) et par sa politique, par exemple de censure. Le fédivers ne prétend pas résoudre des problèmes politiques et humains comme l'attitude à avoir face à, par exemple, les discours racistes. Il offre simplement un mécanisme qui permet de choisir sa politique. Facebook a une politique unique, très inspirée du puritanisme états-unien : la violence physique, ça va, un sein nu, c'est hors de question. Au contraire, sur le fédivers, chaque instance a sa propre politique, et on va sur celle dont la politique nous semble acceptable. Ou bien, si on n'en trouve pas, on crée sa propre instance, seul·e ou avec des personnes partageant nos opinions.

Une fois un compte créé sur une instance, Alice et Bob ont une adresse qui les identifie et qu'ils communiquent. Elle est de la forme `quelquechose@nom.de.domaine`. Si chacun ne pouvait parler qu'aux gens sur la même instance, cela n'aurait évidemment guère d'intérêt. Les instances communiquent donc entre elles, en suivant une série de protocoles décrits dans cet article.

Citons quelques-uns des logiciels les plus connus du fédivers :

- Mastodon et Pleroma visent le micro-blogging, comme le réseau social centralisé Twitter, les messages se nommant pouètes (ou « toots », le bruit d'un mastodonte qui barrit),
- PeerTube fait de l'hébergement de vidéos, comme le réseau social centralisé YouTube, mais en distribuant les vidéos en pair à pair, pour répartir la charge,
- PixelFed permet de montrer ses photos, comme le réseau social centralisé Instagram,

2. Ou bien une oligarchie de quelques serveurs, comme c'est largement le cas du courrier aujourd'hui.

3. Je me souviens d'Usenet, qui était techniquement décentralisé (mais avec serveurs intermédiaires, comme le courrier, et comme le fédivers) et qui soulevait déjà les mêmes questions que certaines personnes semblent découvrir aujourd'hui : censure, engueulades, mensonges, etc.

- WriteFreely est un logiciel d'édition d'articles (plus longs que dans le cas du micro-blogging).

Leur intégration dans le fédivers est très variable. Par exemple, WriteFreely et PeerTube permettent de notifier les abonnés au sujet d'une nouvelle publication, mais, à ma connaissance, ils ne permettent pas encore de commenter articles et vidéos via le fédivers.

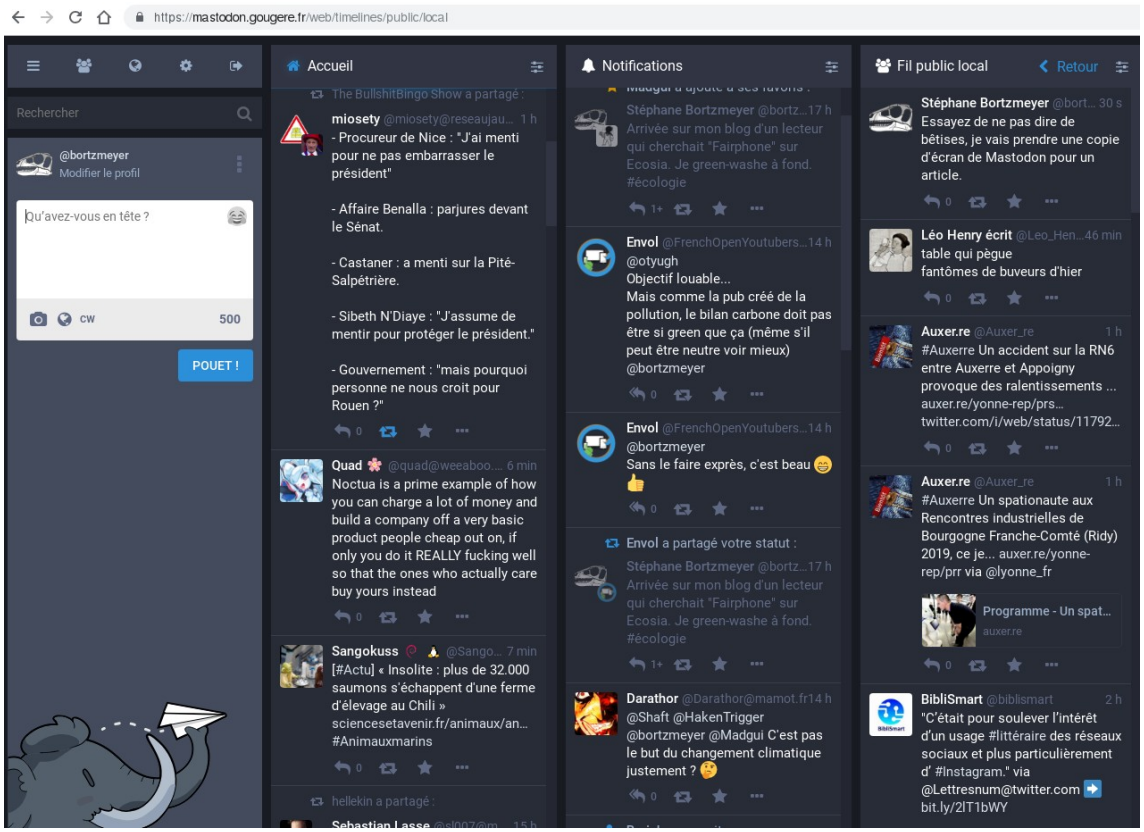

Figure 1 - L'interface Web de Mastodon (mais rappelez-vous que ce n'est qu'une interface, il en existe d'autres). À gauche, les messages des gens que je suis. Au milieu, ceux qui me citent[4].

3 Présentation générale

3.1 Les concepts

Participer au fédivers nécessite de mettre en œuvre plusieurs techniques. Commençons par un peu de terminologie : un utilisateur est identifié par une adresse, qui a la même forme syntaxique qu'une adresse de courrier électronique, et c'est cet identificateur que verront les utilisateurs ordinaires. Mon principal compte sur le fédivers est `bortzmeyer@mastodon.gougere.fr`. Première difficulté, cette syntaxe n'est pas celle qui sera utilisée dans le protocole ActivityPub. (C'est un premier exemple du fait qu'ActivityPub n'est pas tout le fédivers.) Dans ActivityPub, un utilisateur se nomme un **acteur**. Un acteur est identifié par un URI. Par exemple, sur mon compte

4. On peut également avoir des messages privés ou semi-privés, mais j'ai pris soin de ne pas les montrer ici.

Mastodon principal, je suis `https://mastodon.gougere.fr/users/bortzmeyer`. C'est le protocole WebFinger qui permettra de faire le lien entre les adresses et les URI ActivityPub.

Ces acteurs ont une clé cryptographique, composée d'une partie publique et d'une partie privée[5]. Toute l'authentification repose sur cette clé. La récupération de cette clé se fait en HTTPS sur le serveur. On voit donc que la sécurité du fédivers repose sur des techniques classiques, HTTPS et certificats PKIX (et avec DNSSEC, c'est encore mieux).

Les serveurs, les instances, s'échangent entre eux des **activités**, normalisées dans les standards ActivityStreams[2] et ActivityPub[1]. Ce sont des petits bouts de JSON, chacun ayant un type. Ainsi, une activité de type Create va créer un message, une activité de type Follow indiquera le désir de suivre les activités de quelqu'un, etc.

Les activités portent sur des objets qui sont du texte, des images, de la vidéo… Tout objet, toute activité, tout acteur, est identifié par un URI.

Le Web n'a pas de mécanisme de signature des objets qui soit largement déployé. Il n'y a pas de moyen standard de vérifier qu'une page récupérée est bien authentique. HTTPS ne protège que le canal, pas la donnée. Comme ActivityPub repose sur un modèle « push », où le serveur émetteur envoie l'activité au client, et que HTTPS n'authentifie en général pas le client, la technique de sécurité habituelle repose sur les signatures HTTP[5], un mécanisme non normalisé de signature d'une requête HTTP. On signe avec la partie privée de la clé de l'acteur.

3.2 Exemples

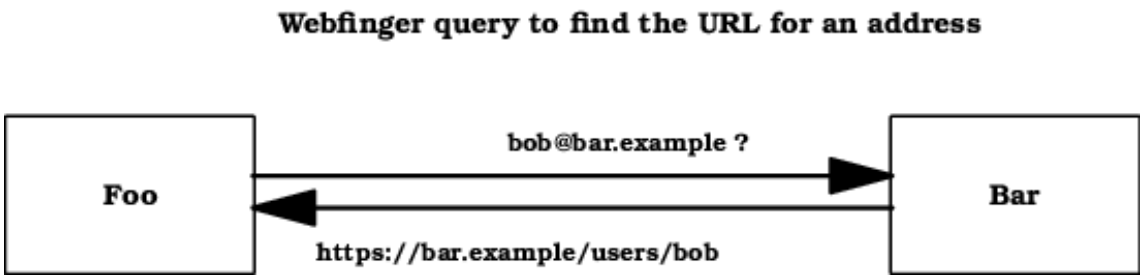

Figure 2 - Requête Webfinger

Si cela vous semble confus, c'est normal. Il est temps de montrer du concret. D'abord, on va utiliser WebFinger, protocole normalisé dans le RFC 7033, pour trouver la description de l'acteur. WebFinger permet de récupérer de l'information sur une personne ou une organisation à partir de son identificateur `utilisateur@domaine`. Il fonctionne sur HTTPS, comme tant de protocoles aujourd'hui, et l'information est renvoyée en JSON.

Supposons que quelqu'un veuille suivre ce que j'écris sur l'instance `write.as`, qui utilise le logiciel WriteFreely. Mon adresse est `bortzmeyer@write.as`. Son serveur va donc faire une requête WebFinger, que je fais ici avec le client HTTP curl (le résultat est du JSON, qu'on formatte avec le logiciel jq, notez que j'ai simplifié en omettant une partie du JSON, dans les exemples qui suivent) :

5. La clé est bien spécifique à l'acteur, pas à l'instance. Mais sa partie privée est en général stockée sur l'instance, le fédivers ne fonctionne pas « de bout en bout », l'administrateur de l'instance peut facilement usurper l'identité d'un acteur.

```
% curl -s https://write.as/.well-known/webfinger\
resource=acct:bortzmeyer@write.as \
   | jq .
{
  "subject": "acct:bortzmeyer@write.as",
  "links": [
    {
      "href": "https://write.as/bortzmeyer/",
      "type": "text/html",
      "rel": "https://webfinger.net/rel/profile-page"
    },
    {
      "href": "https://write.as/api/collections/bortzmeyer",
      "type": "application/activity+json",
      "rel": "self"
    }
  ]
}
```

On a désormais l'URI de l'acteur (c'est celui de type MIME `application/activity+json`), `https://write.as/api/collections/bortzmeyer`. On peut maintenant l'utiliser pour récupérer des informations sur l'acteur. Cette fois, on passe au protocole ActivityPub :

```
% curl -s --header 'Accept: application/activity+json' \
       https://write.as/api/collections/bortzmeyer | jq .
{
  "type": "Person",
  "id": "https://write.as/api/collections/bortzmeyer",
  "inbox": "https://write.as/api/collections/bortzmeyer/inbox",
  "outbox": "https://write.as/api/collections/bortzmeyer/outbox",
  "url": "https://write.as/bortzmeyer/",
  "icon": {
    "type": "Image",
    "mediaType": "image/png",
    "url": "https://write.as/img/avatars/b.png"
  },
  "summary": "https://www.bortzmeyer.org/",
  "publicKey": {
    "id": "https://write.as/api/collections/bortzmeyer#main-key",
    "publicKeyPem": "-----BEGIN PUBLIC KEY-----\nMIIB...\n-----
END PUBLIC KEY-----\n"
  },
}
```

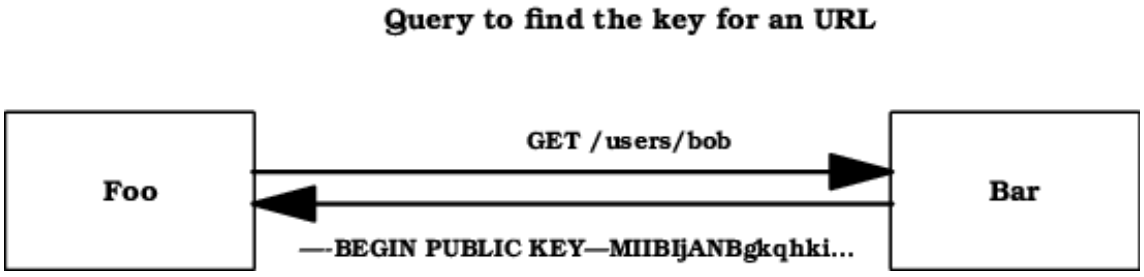

Figure 3 - Récupération d'une clé en ActivityPub

On voit qu'on a récupéré entre autre la clé publique de l'acteur, ce qui permettra de vérifier les messages qu'il enverra.

On peut maintenant s'abonner. C'est plus difficile à faire avec curl, donc je vais simplement montrer un exemple de ce qu'une instance du fédivers envoie à une autre, par une requête HTTPS utilisant la méthode POST, lorsqu'un des utilisateurs de la première (ici, `https://mastodon.gougere.fr/users/bortzmeyer`) veut suivre un utilisateur de la seconde (ici, `https://write.as/api/collections/bortzmeyer`):

```
{
  "id": "https://mastodon.gougere.fr/e2da8821-29f3-4eec-8058-
cc99a8216144",
  "type": "Follow",
  "actor": "https://mastodon.gougere.fr/users/bortzmeyer",
  "object": "https://write.as/api/collections/bortzmeyer" ...
```

On voit que la demande de suivi a un identificateur, un URI, comme toutes les activités, ici, `https://mastodon.gougere.fr/e2da8821-29f3-4eec-8058-cc99a8216144`. Elle a le type Follow (suivre quelqu'un), est envoyée par l'acteur `https://mastodon.gougere.fr/users/bortzmeyer` et porte sur l'acteur `https://write.as/api/collections/bortzmeyer`. Les amateurs de RDF reconnaitront les triplets {sujet, action, objet}.

Comme l'activité affirme venir de `https://mastodon.gougere.fr/users/bortzmeyer`, l'instance réceptrice va vérifier que la signature dans les en-têtes HTTP correspond bien au corps du message, **et** a été faite par la clé de cet acteur. Si la clé n'était pas connue de cette instance réceptrice, elle va alors la demander.[6]

6. Vous noterez que c'est pour cela qu'on ne peut pas écrire un serveur fédivers uniquement sous la forme d'un client. Il doit forcément être capable d'être serveur, d'écouter les requêtes entrantes.

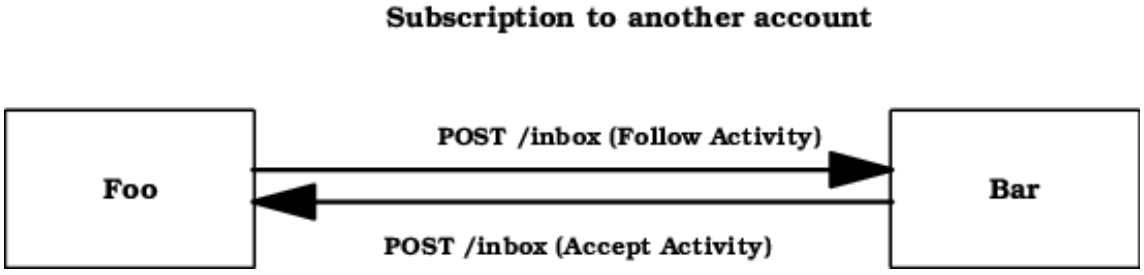

Figure 4 - Abonnement

Si elle est d'accord, l'instance réceptrice va alors envoyer une activité de type Accept, référençant l'URI de la demande. Désormais, elle enverra une copie des messages à l'abonné (toujours en les signant).

Et si on écrit un message, par exemple dans le cas du micro-blogging ? L'instance envoie alors ce genre d'activité, à toutes les instances dont un utilisateur suit l'acteur qui a écrit le message :

```
{
  "id":
"https://mastodon.gougere.fr/users/bortzmeyer/statuses/1014705058
83466588/activity",
  "type": "Create",
  "actor": "https://mastodon.gougere.fr/users/bortzmeyer",
  "published": "2019-01-24T08:04:05Z",
  "to": [
    "https://www.w3.org/ns/activitystreams#Public"
  ],
  "cc": [
    "https://mastodon.gougere.fr/users/bortzmeyer/followers"
  ],
  "object": {
    "id": "https://mastodon.gougere.fr/users/bortzmeyer/statuses/
101470505883466588",
    "type": "Note",
    "published": "2019-01-24T08:04:05Z",
    "url":
"https://mastodon.gougere.fr/@bortzmeyer/101470505883466588",
    "content": "<p>Sur Twitter, récit d&apos;Amaelle Guiton,
retour du <a
href=\"https://twitter.com/amaelle_g/status/1088141070691721217\"
>FIC</a>.</p> »
    ]
```

Les points à noter dans cette activité : elle est de type Create, elle est destinée à tous (`https://www.w3.org/ns/activitystreams#Public`), et elle inclut une activité de type Note (un court texte) dont le contenu est par défaut en HTML. L'instance réceptrice va, si elle accepte, enregistrer le message dans sa base de données, et le rendre accessible à tous ses utilisateurs (puisque ce message est public).

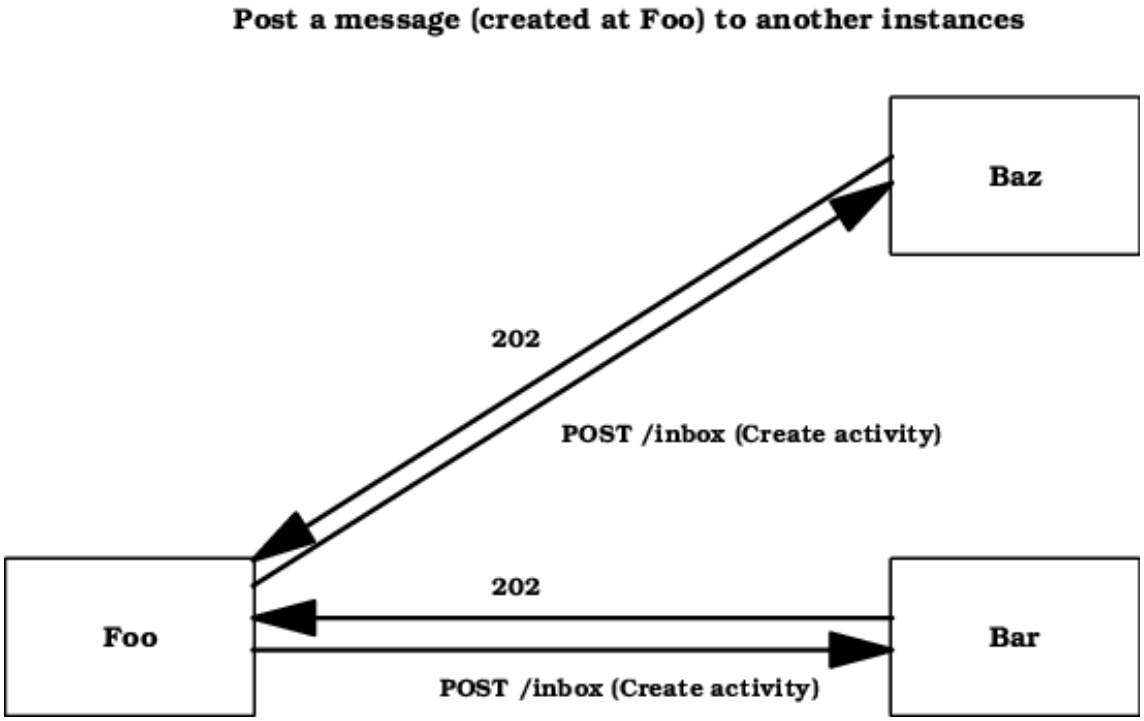

Figure 5 - Relayage d'un message d'une instance à toutes les instances (ici deux) où quelqu'un suit l'auteur

4 Les normes et références

On a vu qu'il n'y a pas réellement **un** protocole du fédivers, mais plutôt un ensemble pas forcément bien organisé. On dit souvent que le fédivers fonctionne grâce à ActivityPub mais c'est tellement résumé que ça en devient trompeur : un programme qui ne connaitrait qu'ActivityPub serait bien incapable d'interagir avec qui que ce soit dans le fédivers. Mais comment appeler l'ensemble des protocoles nécessaires, ensemble qui n'est documenté nulle part ? Il y avait eu des discussions autour du terme « Activity Suite », qui n'est pas mal mais est déjà pris par un logiciel commercial. Il avait été également suggéré de dire « Mastodon Suite », pour refléter le fait que Mastodon est la locomotive du fédivers, et la référence (toute mise en œuvre du fédivers cherche d'abord à être compatible avec Mastodon). Pour l'instant, aucun terme ne s'est imposé.

ActivityPub est donc très insuffisant pour les besoins du fédivers. Ses identificateurs sont des URI, que certaines personnes trouvent peu pratiques. Et il n'a **aucune** forme d'authentification, ce qui est évidemment inenvisageable pour la grande majorité des applications sur l'Internet. Il ne s'agit pas d'un oubli ou d'une erreur de conception, mais d'un choix délibéré, dû au fait qu'il n'y a actuellement pas de consensus sur la bonne façon de faire, et que le domaine est encore bouillonnant d'activité.

ActivityPub est en fait plutôt un cadre qu'un protocole : il spécifie un certain nombre de choses, qu'il faudra compléter (au sein d'une « Activity Suite » ou « Mastodon Suite ») pour faire un système utilisable. C'est pour cela qu'il n'existe pas d'application cliente générique pour tous les services ActivityPub : ils sont bien trop différents.

À propos d'application cliente, notons qu'ActivityPub a deux parties, une serveur<->serveur, entre les instances, et une client<->serveur, entre un client (par exemple sur un ordiphone) et une instance. En pratique, personne n'utilise cette deuxième partie, trop complexe et nécessitant de pouvoir recevoir des appels entrants. Tous les logiciels serveur ont une API à eux[7], et c'est cette API que doivent connaître les clients.

Les normes à connaître, si on veut comprendre le fonctionnement technique du fédivers, sont donc :

- ActivityPub[1], norme du W3C (World Wide Web Consortium),
- ActivityStreams[2] (en deux parties, la terminologie, et la norme elle-même), également au W3C, c'est la couche de description des objets, sur laquelle s'appuie ActivityPub,
- Les signatures HTTP[5], brouillon de norme à l'IETF, la principale méthode d'authentification,
- Webfinger[6], pour faire le lien entre les adresses visibles et les URI d'ActivityPub,
- Les « linked data signatures »[7], qui ne sont pas encore utilisées[8] mais qui permettraient de mieux authentifier, notamment les messages relayés.

5 Un exemple de serveur

Tonola est une mise en œuvre des protocoles du fédivers, développée par Chloé Baut et moi-même. Elle n'a pas vocation à concurrencer les logiciels existants, son but est purement expérimental, afin d'explorer les possibilités du fédivers et de comprendre son fonctionnement et l'interopérabilité (pas toujours facile à atteindre). Tonola doit son nom à Tonola de Magnasco, la mère de Francisco de Tassis[9], fondateur du premier service postal européen au 15e siècle. Le serveur Tonola est écrit en Python, et stocke les données dans une base SQLite. Pour la partie serveur, qui répond à des requêtes qui peuvent possiblement arriver en parallèle, Tonola utilise le cadriciel Quart. Pour la partie cliente (envoyer des requêtes à l'extérieur), il se sert d'aiohttp. Il faut travailler avec des fils d'exécution différents, ou bien opérer de façon asynchrone (ce que fait Tonola) puisque les activités de serveur et de client d'une instance du fédivers se produisent en parallèle.

Tonola est mono-utilisateur, contrairement aux autres logiciels serveur existants. Il a une interface Web très sommaire et une API spécifique (cf. p. 9). Mais il permet de s'abonner, d'être abonné, et d'écrire. Ici, par exemple, je suis le compte `aprilorg@pouet.april.org` :

```
% curl --user 'foobar:XXXXX' \
```

7. Pleroma, outre son API propre, met également en œuvre celle de Mastodon.

8. Mastodon les génère, mais ne les vérifie pas.

9. Également écrit Franz von Taxis, puisque l'entreprise familiale était - est toujours - multinationale.

```
    https://ap.bortzmeyer.fr/api/custom/follow\?
address=aprilorg@pouet.april.org
Request to follow aprilorg@pouet.april.org sent (return status is
202)
```

Et dans le journal du serveur Tonola, on voit bien la requête Webfinger faite par l'instance `pouet.april.org` pour nous connaître, puis la demande d'informations sur l'acteur (notamment sa clé), et enfin la réponse (positive) :

```
TONOLA - INFO - 2019-10-02 08:57:12Z - Webfinger info about
acct:foobar@ap.bortzmeyer.fr for 62.210.101.52

TONOLA - INFO - 2019-10-02 08:57:12Z - Actor info about foobar
for 62.210.101.52

TONOLA - DEBUG - 2019-10-02 08:57:12Z - Reply to my request to
follow from https://pouet.april.org/inbox: 202 ()

TONOLA - DEBUG - 2019-10-02 08:57:12Z - Activity received on the
general inbox:
"{"id":"https://pouet.april.org/users/aprilorg#accepts/follows/12
67","type":"Accept","actor":"https://pouet.april.org/users/
aprilorg","object":{"id":"https://ap.bortzmeyer.fr/requests/
4569747","type":"Follow","actor":"https://ap.bortzmeyer.fr/
actors/foobar","object":"https://pouet.april.org/users/
aprilorg"}}"
```

Notez que l'activité Accept incluait l'activité Follow à laquelle elle répondait. Avec du JSON bien formaté, c'est plus joli :

```
{
  "id":
"https://pouet.april.org/users/aprilorg#accepts/follows/1267",
  "type": "Accept",
  "actor": "https://pouet.april.org/users/aprilorg",
  "object": {
    "id": "https://ap.bortzmeyer.fr/requests/4569747",
    "type": "Follow",
    "actor": "https://ap.bortzmeyer.fr/actors/foobar",
    "object": "https://pouet.april.org/users/aprilorg"
  }
}
```

Et pour écrire un message ? On tape :

```
% curl --user 'foobar:XXXX' --data "Bonjours, JRES" \
       https://ap.bortzmeyer.fr/api/custom/post
#13425 Done
```

Le message 13425 est enregistré[10], et transmis à toutes les instances dont un utilisateur s'est abonné. Dans le journal de Tonola, on voit la transmission à l'instance `mastodon.social` :

```
TONOLA - DEBUG - 2019-10-02 12:12:16Z - Forward toot 13425 to
peer mastodon.social

TONOLA - DEBUG - 2019-10-02 12:12:16Z - Toot {"id":
"https://ap.bortzmeyer.fr/activities/13425", "type": "Create",
"actor": "https://ap.bortzmeyer.fr/actors/foobar", "object":
{"id": "https://ap.bortzmeyer.fr/toots/13425", "type": "Note",
"mediaType": "text/html", "published": "2019-10-02T12:12:16Z",
"content": "<p>Bonjours, JRES</p>", "to": "https://www.w3.org/ns/
activitystreams#Public"}}

TONOLA - DEBUG - 2019-10-02 12:12:16Z - Starting POST request for
https://mastodon.social/inbox. I will send: <CIMultiDict('Date':
'Wed, 02 Oct 2019 12:12:16 GMT', 'Signature':
'keyId="https://ap.bortzmeyer.fr/actors/foobar#main-
key",algorithm="rsa-sha256",headers="(request-target) host date
digest",signature="UpLa...=="', 'Digest': 'SHA-
256=aYi2n2rpS3VzciBSp8CdmdCOxQa/s4ehutyz+4Aj5Vg=', 'User-Agent':
'Tonola/0.0')>
```

Le message ici est du pur texte[11]. C'est plus compliqué si on envoie des données « multimédia ». ActivityStreams a des objets de type Image ou Audio mais ces objets sont évidemment plus lourds, et JSON ne permet pas de transporter du binaire. On envoie donc un lien vers le contenu multimédia, ce dernier restant stocké sur l'instance d'origine.

Une caractéristique fondamentale du fédivers est son caractère décentralisé. Chaque instance a sa politique. Ainsi, Tonola honore l'activité Delete (suppression d'un contenu) si elle est signée, mais une instance désireuse de tout archiver pourrait parfaitement ignorer ces Delete. De même, Mastodon ne permet pas de chercher dans les messages, ce qui est présenté comme une sécurité, mais d'autres instances peuvent le permettre.

Tonola ne met pas (encore?) en œuvre les Linked Data Signatures[7].

10. Et visible sur le Web : tous les objets ActivityStreams ont un identificateur qui n'est pas juste un URI mais également un URL, donc auquel on peut accéder avec un client Web. Essayez https://ap.bortzmeyer.fr/toots/13425.

11. C'est Tonola qui en a fait de l'HTML, en ajoutant <p>.

Comme vous l'avez vu, ActivityPub est plus « push » que « pull ». Les instances qui ont quelque chose à dire vous l'envoient par une requête HTTPS de méthode POST[12]. Pas besoin d'attente active. L'URI exact utilisé est donné par l'information nommée « inbox » dans l'acteur. Notez qu'il existe des boîtes aux lettres (« inbox ») collectives (communes à toute l'instance) et des individuelles.

6 Conclusion

Le fédivers est en pleine activité, de nouveaux projets apparaissent régulièrement, les projets existants sont nombreux et actifs[13].La première conférence ActivityPub a eu lieu à Prague en septembre 2019. Il est difficile de dire quel sera l'avenir du fédivers mais en tout cas, pour l'instant, l'idée est bien vivante.

Bibliographie

[1] La norme ActivityPub

[2] La terminologie ActivityStreams et la norme ActivityStreams.

[3] Big List of ActivityPub Projects, une liste de logiciels mettant en œuvre les protocoles du fédivers.

[4] How to implement a basic ActivityPub server et How to make friends and verify requests, deux bons articles par l'auteur de Mastodon, pour comprendre l'articulation des différents composants du fédivers .

[5] Signing HTTP Messages, le brouillon de RFC.

[6] RFC 7033 "WebFinger", il en existe un résumé en français.

[7] Linked Data Signatures 1.0, un projet de norme pour la signature des objets.

[8] L'excellent rapport de stage de Nathalie Rafaralahy sur l'aspect technique d'ActivityPub. La même auteure a fait un autre mémoire, plus philosophique.

[9] Dépôt officiel de Tonola.

[10] Un exemple d'utilisation de l'API de Mastodon (p. 9), pour faire un « bot » DNS sur le fédivers.

12. On peut ainsi « spammer » une instance qui n'a rien demandé, en lui envoyant des messages en quantité. Mais elle ne les stockera pas forcément (cela dépend de ses contrôles) et ne les affichera pas dans tous les cas aux utilisateurs. Et il y a traçabilité, puisque les messages sont signés. On saura donc que le message vient bien de https://badguy.example/evil. Et le message d'origine, qui fait foi, reste sur l'instance de départ.

13. Parmi les derniers, citons Mobilizon, qui vise à créer un logiciel permettant l'organisation d'évènements, que ce soit des goûters d'anniversaires, des manifestations de rue, ou des rencontres professionnelles, sans laisser ses données personnelles à Facebook ou Eventbrite.