Engineer

Protocole

règles convenues entre deux programmes pour communiquer

Ce que c’est

Un protocole est un accord écrit sur une conversation entre programmes : la forme de chaque message, son sens, le premier interlocuteur à parler, les enchaînements autorisés et la réaction de chacun lorsque l’autre dit quelque chose d’inattendu. Deux programmes écrits par des personnes différentes dans des langages différents peuvent fonctionner ensemble parce que chacun a été construit d’après le même document, plutôt que d’après l’autre programme.

Le glossaire du NIST rassemble plusieurs définitions tirées de ses publications, avec un noyau commun : des règles sur le format des messages et les procédures qui les entourent, suivies par au moins deux parties pour atteindre un objectif commun. L’une d’elles nomme l’aspect facile à oublier : les règles décrivent l’ordre des messages autant que leur structure. Un message au bon format, mais au mauvais moment, rompt la conversation aussi sûrement qu’un message mal formé.

Les spécifications publiées sous forme de RFC énoncent leurs règles avec des mots-clés en capitales. MUST est une exigence absolue. SHOULD n’autorise un écart que si toutes ses implications sont comprises et pesées. MAY désigne quelque chose de véritablement facultatif. Depuis 2017, ces mots n’ont ce sens qu’en capitales : un « must » en minuscules dans le même document relève de l’anglais ordinaire. L’écart entre SHOULD et MUST est celui qui sépare un programme inhabituel d’un programme incorrect.

Les protocoles se superposent. Un protocole applicatif, la partie qui connaît les pages, les menus ou le courrier, repose généralement sur un protocole de transport dont le seul rôle est de déplacer des octets entre deux machines. TCP, le transport utilisé par la plupart d’entre eux, fournit ce que sa spécification appelle « un service de flux d’octets fiable et ordonné » : les octets arrivent au

complet et dans leur ordre d'envoi. Il ne promet en revanche aucun lien entre les morceaux écrits par un côté et ceux lus par l'autre. Un flux ne contient pas de messages : chaque protocole placé au-dessus doit donc marquer la fin des siens par une fin de ligne, une longueur envoyée à l'avance, la fermeture de la connexion ou un autre repère convenu. UDP fait le compromis opposé. Il transporte des datagrammes distincts avec un minimum de mécanismes, sans garantir ni la livraison ni la protection contre les doublons. Chaque transport distingue les services d'une machine par un numéro de port, et TCP et UDP disposent chacun de leur propre numérotation.

Les autres caractéristiques d'un protocole découlent de quelques choix faits, délibérément ou non, lors de toute conception :

- **Qui parle en premier.** Certains serveurs saluent le client, d'autres attendent une requête en silence. Si les deux côtés attendent, rien ne se passe avant l'expiration d'un temporisateur.
- **Ce que le serveur garde en mémoire.** Dans un protocole sans état, chaque requête contient tout ce qui est nécessaire pour y répondre. Dans un protocole avec état, le sens d'un message dépend de ce qui l'a précédé sur la même connexion.
- **Comment il évolue.** Les numéros de version, les codes d'option et les champs à ignorer lorsqu'ils sont inconnus représentent la place qu'un protocole réserve à son avenir.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce qu'un protocole ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Du Wi-Fi, des pare-feu et des accès à distance configurés pour travailler de partout, en laissant entrer uniquement les personnes autorisées.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. 30 instances VPS Ubuntu Linux administrées — 30 serveurs Linux gérés, avec récupération prévue

30 instances VPS Ubuntu Linux administrées Contactez-nous →

Pourquoi c'est important

Le protocole rend chaque extrémité remplaçable. Tout client écrit d'après le document peut communiquer avec tout serveur qui le suit : un lecteur peut donc changer de logiciel et un serveur être réécrit dans un autre langage sans que l'autre côté s'en aperçoive. Des programmes qui ne s'accordent que parce qu'ils ont été construits ensemble ne peuvent pas évoluer séparément.

Le protocole détermine aussi une grande partie de la posture de sécurité avant toute ligne de code. Le côté qui ouvre la connexion décide de ce qu'il faut laisser entrer : un serveur qui attend des clients a besoin d'un port ouvert au monde, tandis qu'un programme qui ne fait qu'établir des connexions sortantes n'en a pas besoin. Le contenu des messages détermine ce qui peut être journalisé, compté ou divulgué. Une requête qui nomme un chemin peut être journalisée par chemin, et masquer l'adresse réseau d'un lecteur n'empêche pas le serveur de compter les requêtes. Un protocole qui fonctionne sans chiffrement expose chaque requête à tous les réseaux traversés.

Enfin, les protocoles survivent aux programmes qui les parlent. Les implémentations sont remplacées périodiquement, après quelques années, tandis que la conversation se poursuit pendant des décennies. Chaque comportement dont dépendent les programmes déployés en devient une partie, quoi qu'en dise le document. L'Internet Architecture Board a décrit les deux faces du problème : les erreurs tolérées jusqu'à devenir la norme et la marge d'évolution inutilisée jusqu'à ne plus fonctionner.

À quoi cela ressemble en pratique

Ce site répond sur quatre protocoles à partir d'un seul dossier de fichiers sur un seul serveur : HTTP, Gemini, Gopher et un service onion sur Tor. La page décrit Gopher comme un protocole à menus de 1991, sur une connexion TCP, sans chiffrement ni scripts, et donne une requête complète saisie à la main :

```
printf '/contact\r\n' | nc engineer.company 70
```

Elle présente cette ligne comme l'intégralité du protocole. Dans cette architecture, un seul déploiement écrit un seul dossier et quatre portes d'entrée le lisent : aucune n'est donc une copie maintenue séparément susceptible de diverger.

Les règles de pare-feu sont vérifiées par leur position plutôt que par leur présence, car un filtre de paquets évalue une liste ordonnée et s'arrête à la première correspondance. L'un des contrôles porte sur les protocoles : il existe exactement une règle par port, toujours accompagnée d'un protocole, car l'omettre élargit silencieusement la surface exposée. La surface publique déclarée est une courte liste de cinq ports, vérifiée dans son ensemble : un port présent sans avoir été déclaré fait donc échouer la vérification.

La quatrième porte est un miroir onion, dont le protocole a façonné le pare-feu comme les rapports. Le démon d'anonymat atteint le réseau en créant des circuits sortants : aucune connexion n'arrive de l'extérieur, aucun port ne s'ouvre et les deux couches de pare-feu n'ont rien à synchroniser. Les requêtes Gemini et les connexions Gopher sont comptées par jour, sans adresse ni chemin. Le service onion n'a ni compteur ni option pour en ajouter un, car ce site choisit délibérément de ne pas compter ce public.

Où les problèmes surviennent

Une erreur tolérée devient la norme. Le principe de robustesse, « soyez strict dans ce que vous envoyez et tolérant dans ce que vous acceptez », devait aider les programmes à fonctionner ensemble. Appliqué sans limite, il permet à une implémentation défectueuse d'être publiée et de se diffuser. L'implémentation suivante doit accepter le même défaut pour fonctionner avec elle. Celui-ci devient alors une norme de fait avec laquelle tout nouveau programme doit être compatible, bogue pour bogue. La RFC 9413 le dit clairement : les implémentations déployées deviennent un test de conformité qui peut supplanter la spécification. Rejetez ce que la spécification interdit le plus tôt possible dans la vie d'un protocole, et rendez l'erreur visible à une personne capable de la corriger.

Deux programmes ne s'accordent pas sur la fin d'un message. Dans un flux d'octets, la frontière est là où le protocole la place. Un protocole qui propose deux façons de la signaler, ou un analyseur qui en accepte une version approximative, permet à un proxy et au serveur derrière lui de lire les mêmes octets comme des messages différents. Ce que l'un traite comme la fin d'une requête devient pour l'autre une nouvelle requête que le proxy n'a jamais vérifiée. N'autorisez qu'une règle de délimitation, rejetez toute ambiguïté et faites analyser les messages de la même façon à chaque étape.

Une règle nomme un port sans nommer de protocole. Un numéro de port ne signifie rien tant qu'un transport n'est pas nommé. Selon l'outil, une règle de pare-feu qui n'indique qu'un numéro peut l'ouvrir à la fois sur TCP et UDP. Le service qui avait besoin de l'un l'obtient, et l'autre reste ouvert à ce qui y écoutera plus tard. Indiquez le protocole dans chaque règle et comparez toute la surface ouverte à une liste déclarée.

La marge d'évolution cesse de fonctionner. Si aucun émetteur déployé n'utilise jamais les points d'extension d'un protocole, des récepteurs finissent par rejeter ce qu'ils n'ont jamais vu, et la première véritable extension échoue face à eux. Plus un tel récepteur reste déployé, plus il est difficile à corriger. Mettez les points d'extension à l'épreuve dès le début, y compris avec des valeurs sans signification, pour tester constamment la tolérance au lieu de la présumer.

Pour aller plus loin

- Glossaire NIST CSRC : protocole — Les définitions utilisées dans les publications de sécurité de NIST, avec leur document d'origine.
- RFC 9293 : Transmission Control Protocol — Le standard TCP actuel, avec son flux d'octets sans frontières entre messages.
- RFC 768 : User Datagram Protocol — L'intégralité d'UDP en trois pages, datant de 1980.
- RFC 1436 : protocole Internet Gopher — Une description informative d'un protocole applicatif complet, avec des exemples de sessions.
- RFC 9110 : sémantique HTTP — Le sens des requêtes, réponses, méthodes et codes d'état, indépendamment de la version transportée sur le réseau.
- RFC 9413 : maintenir des protocoles robustes — L'analyse de l'Internet Architecture Board sur la façon dont la tolérance des erreurs les enracine.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le parcours de déploiement, les risques opérationnels et les personnes d'astreinte. Nous pouvons définir un changement avec des vérifications, une procédure de reprise et une responsabilité claire.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce qu'un protocole ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/protocol/