Engineer

protocolo

reglas acordadas para que se comuniquen dos programas

Qué es

Un protocolo es un acuerdo escrito sobre una conversación entre programas: qué aspecto tiene cada mensaje, qué significa, qué lado habla primero, qué puede seguir a qué y qué hace cualquiera de las partes cuando la otra dice algo inesperado. Dos programas escritos por personas distintas en lenguajes diferentes pueden funcionar juntos porque cada uno se ha construido conforme al mismo documento, no conforme al otro programa.

El glosario de NIST reúne varias definiciones de sus publicaciones, y todas comparten un núcleo: reglas sobre el formato de los mensajes y los procedimientos que los rodean, seguidas por dos o más partes para alcanzar un objetivo conjunto. Una de ellas nombra la mitad que suele olvidarse: las reglas describen tanto el orden de los mensajes como su estructura. Un mensaje con el formato correcto en el momento equivocado rompe la conversación con la misma certeza que uno mal formado.

Las especificaciones publicadas como RFC expresan sus reglas mediante palabras clave en mayúsculas. MUST es un requisito absoluto. SHOULD solo permite apartarse de la regla cuando se comprenden y se sopesan todas las consecuencias. MAY indica algo verdaderamente opcional. Desde 2017, esas palabras solo tienen ese significado cuando están en mayúsculas, por lo que un «must» en minúsculas dentro del mismo documento es inglés corriente. La distancia entre SHOULD y MUST es la que separa un programa poco habitual de uno incorrecto.

Los protocolos se organizan en capas. Un protocolo de aplicación, la parte que entiende de páginas, menús o correo, suele funcionar sobre un protocolo de transporte cuyo único cometido es mover bytes entre dos máquinas. TCP, el transporte que utiliza la mayoría de ellos, proporciona lo que su especificación

llama «un servicio fiable de flujo de bytes en orden»: los bytes llegan completos y en el orden en que se enviaron. Tampoco promete ninguna relación entre los fragmentos que escribe un lado y los que lee el otro. Un flujo no contiene mensajes, por lo que cada protocolo que se apoya en él debe señalar dónde terminan los suyos, con un fin de línea, una longitud enviada de antemano, el cierre de la conexión u otro marcador acordado. UDP opta por lo contrario. Transporta datagramas independientes con un mínimo de mecanismos y no garantiza ni la entrega ni la protección frente a duplicados. Cada transporte distingue los servicios de una máquina mediante un número de puerto, y ambos numeran los suyos por separado.

Las demás características de un protocolo proceden de unas cuantas decisiones que se toman en todo diseño, de forma deliberada o no:

- **Quién habla primero.** Algunos servidores saludan al cliente; otros esperan en silencio una petición, y cuando ambos lados esperan no pasa nada hasta que vence un temporizador.
- **Qué recuerda el servidor.** En un protocolo sin estado, cada petición lleva todo lo necesario para responderla. En uno con estado, el significado de un mensaje depende de lo ocurrido antes en la misma conexión.
- **Cómo crece.** Los números de versión, los códigos de opción y los campos que deben ignorarse cuando no se conocen son el espacio que un protocolo deja para su futuro.

Utilice el concepto en una decisión

Aplique «¿Qué es un protocolo?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Wi-Fi, cortafuegos y acceso remoto configurados para trabajar desde cualquier lugar y permitir el acceso solo a las personas autorizadas.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. Administración de 30 VPS Ubuntu Linux — 30 servidores Linux administrados con recuperación planificada

Administración de 30 VPS Ubuntu Linux Póngase en contacto →

Por qué importa

Un protocolo hace que cualquiera de los extremos pueda sustituirse. Cualquier cliente escrito conforme al documento puede hablar con cualquier servidor escrito conforme a él, de modo que un lector puede cambiar de software y un

servidor puede reescribirse en otro lenguaje sin que el otro lado lo note. Los programas que solo se entienden porque se construyeron juntos no pueden cambiar por separado.

El protocolo también fija buena parte de la postura de seguridad antes de que exista código alguno. Qué lado abre la conexión determina qué hay que dejar entrar: un servidor que espera clientes necesita un puerto abierto al mundo, mientras que un programa que solo inicia conexiones salientes no necesita ninguno. Lo que contienen los mensajes determina qué puede registrarse, contarse o filtrarse. Una petición que indica una ruta puede registrarse por ruta, y ocultar la dirección de red de un lector no impide que el servidor cuente peticiones. Un protocolo que funciona sin cifrado muestra cada petición a todas las redes que atraviesa.

Por último, los protocolos sobreviven a los programas que los hablan. Las implementaciones se sustituyen cada pocos años mientras la conversación continúa durante décadas, y todo comportamiento del que dependen los programas desplegados pasa a formar parte de ella, diga lo que diga el documento. El Internet Architecture Board ha escrito sobre las dos caras de ese problema: errores que se toleran hasta convertirse en la norma y espacio para crecer que nadie utiliza hasta que deja de funcionar.

Cómo se aplica en la práctica

Este sitio responde a través de cuatro protocolos, desde una sola carpeta de archivos en un servidor: HTTP, Gemini, Gopher y un servicio onion sobre Tor. La página describe Gopher como un protocolo de menús de 1991 con TCP sin más, sin cifrado ni scripts, y muestra una petición completa escrita a mano:

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

Llama a esa línea el protocolo entero. Por debajo, un despliegue escribe una carpeta y cuatro puertas de entrada la leen, por lo que ninguna es una copia mantenida por separado que pueda divergir.

Las reglas del cortafuegos se verifican por su posición, no por su presencia, porque un filtro de paquetes evalúa una lista ordenada y se detiene en la primera coincidencia. Una de las reglas de la comprobación trata sobre protocolos: existe

exactamente una regla por puerto y siempre con un protocolo, porque una regla sin él amplía silenciosamente la superficie. La superficie pública declarada es una lista breve de cinco puertos, comprobada como un todo, de modo que un puerto que aparezca sin estar declarado hace fallar la comprobación.

La cuarta puerta es un espejo onion, y su protocolo condicionó tanto el cortafuegos como los informes. El demonio de anonimato llega a la red estableciendo circuitos salientes, de modo que ninguna conexión se inicia desde fuera, no se abre ningún puerto y ninguna de las dos capas de cortafuegos tiene nada que sincronizar. Las peticiones Gemini y las conexiones Gopher se cuentan por día, sin dirección ni ruta. El servicio onion no tiene contador ni opción para añadirlo, porque este sitio deja deliberadamente sin contar a esos lectores.

Dónde surgen los problemas

Un error tolerado se convierte en la norma. El principio de robustez, «sea conservador en lo que envía y liberal en lo que acepta», pretendía ayudar a los programas a funcionar juntos. Aplicado sin límites, permite publicar y extender una implementación defectuosa. La siguiente implementación tiene que aceptar el mismo defecto para funcionar con ella, y el defecto se convierte en una norma de facto con la que todo lo nuevo debe ser compatible, error por error. El RFC 9413 lo dice con claridad: las implementaciones desplegadas se convierten en una prueba de conformidad que puede imponerse a la especificación. Rechace lo que prohíba la especificación lo antes posible en la vida de un protocolo y haga visible el error a alguien que pueda corregirlo.

Dos programas discrepan sobre dónde termina un mensaje. En un flujo de bytes, el límite está donde lo indique el protocolo. Un protocolo que ofrece dos formas de señalarlo, o un analizador que acepta una versión poco rigurosa, permite que un proxy y el servidor que tiene detrás interpreten los mismos bytes como mensajes distintos. Lo que uno trata como el final de una petición, el otro lo trata como una petición nueva que el proxy nunca comprobó. Permita una sola regla de delimitación, rechace toda ambigüedad y haga que cada salto la interprete de la misma manera.

Una regla indica un puerto, pero no un protocolo. Un número de puerto no significa nada hasta que se identifica el transporte y, según la herramienta, una regla de cortafuegos escrita solo con un número puede abrirlo tanto en TCP

como en UDP. El servicio que necesitaba uno lo obtiene, y el otro queda abierto a lo que escuche allí más adelante. Escriba el protocolo en cada regla y compare toda la superficie abierta con una lista declarada.

El espacio para crecer deja de funcionar. Si ningún emisor desplegado utiliza los puntos de extensión de un protocolo, acaban escribiéndose receptores que rechazan lo que nunca han visto, y la primera extensión real falla al encontrarse con ellos. Cuanto más tiempo permanezca desplegado un receptor así, más difícil será corregirlo. Ponga a prueba los puntos de extensión desde el principio, incluso con valores que no signifiquen nada, para comprobar continuamente la tolerancia en lugar de darla por supuesta.

Lecturas adicionales

- Glosario de NIST CSRC: protocolo — Las definiciones utilizadas en las publicaciones de seguridad de NIST, cada una con su documento de origen.
- RFC 9293: protocolo de control de transmisión — El estándar TCP actual, incluido el flujo de bytes sin límites de mensaje.
- RFC 768: protocolo de datagramas de usuario — Todo UDP en tres páginas, de 1980.
- RFC 1436: el protocolo Gopher de Internet — Una descripción informativa de un protocolo de aplicación completo, con ejemplos de sesiones.
- RFC 9110: semántica de HTTP — El significado de peticiones, respuestas, métodos y códigos de estado, independiente de la versión transmitida por la red.
- RFC 9413: mantenimiento de protocolos robustos — El Internet Architecture Board explica por qué tolerar errores los consolida.

Glosario

Elija según sus restricciones reales

Describa la ruta de despliegue, los riesgos operativos y quiénes se encargan de las guardias. Podemos definir el alcance de un cambio con comprobaciones, una vía de recuperación y responsabilidades claras.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es un protocolo?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

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

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

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

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/protocol/