Engineer

Protokol

de aftalte regler, to programmer følger for at tale sammen

Hvad det er

En protokol er en skriftlig aftale om en samtale mellem programmer: hvordan hver besked ser ud, hvad den betyder, hvilken side der taler først, hvad der må følge efter hvad, og hvad hver side gør, når den anden siger noget uventet. Programmer skrevet af forskellige folk kan arbejde sammen, fordi de er bygget efter det samme dokument og ikke efter hinanden.

NIST's ordliste samler flere definitioner fra institutets publikationer, og de har en fælles kerne: regler for beskedernes format og for procedurerne omkring dem, som to eller flere parter følger for at nå et fælles mål. En af dem nævner den halvdel, der er let at glemme: reglerne beskriver beskedernes rækkefølge lige så vel som deres opbygning. En besked i det rigtige format på det forkerte tidspunkt bryder samtalen lige så sikkert som en forkert formet besked.

Specifikationer, der udgives som RFC'er, angiver deres regler med nøgleord i versaler. MUST er et absolut krav. SHOULD tillader en afvigelse, men kun når de fulde konsekvenser er forstået og afvejet. MAY markerer noget, der er helt valgfrit. Siden 2017 har ordene kun den betydning med store bogstaver, så et »must« med små bogstaver i samme dokument er almindeligt engelsk. Afstanden mellem SHOULD og MUST er afstanden mellem et program, der er usædvanligt, og et, der er forkert.

Protokoller ligger i lag. En applikationsprotokol, den del, der kender til sider, menuer eller post, kører som regel oven på en transportprotokol, hvis eneste opgave er at flytte bytes mellem to maskiner. TCP, den transport de fleste bruger, leverer ifølge sin specifikation en pålidelig bytestrøm i rækkefølge: bytes når frem i deres helhed og i den rækkefølge, de blev sendt. Den lover til gengæld ingen sammenhæng mellem de bidder, den ene side skriver, og de bidder, den anden side læser. En strøm indeholder ingen beskeder, så hver protokol ovenpå

må selv markere, hvor dens beskeder slutter: med et linjeskift, med en længde sendt forrest, ved at forbindelsen lukkes eller med en anden aftalt markør. UDP indgår den modsatte handel. Den bærer separate datagrammer med et minimum af mekanik, og hverken levering eller beskyttelse mod dubletter er garanteret. Hver transport skelner tjenesterne på én maskine fra hinanden med et portnummer, og de to nummererer deres porte hver for sig.

Resten af en protokols karakter kommer fra nogle få valg, som ethvert design træffer, med vilje eller ej:

- **Hvem taler først.** Nogle servere hilser på klienten, andre venter tavst på en forespørgsel, og når begge sider venter, sker der intet, før en timer udløber.
- **Hvad serveren husker.** I en tilstandsløs protokol bærer hver forespørgsel alt, hvad der skal til for at besvare den. I en protokol med tilstand afhænger en beskeds betydning af det, der kom før den på samme forbindelse.
- **Hvordan den vokser.** Versionsnumre, tilvalgskoder og felter, der skal ignoreres, når de er ukendte, er den plads, en protokol efterlader til sin egen fremtid.

Brug begrebet i en beslutning

Brug »Hvad er en protokol?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Wi-Fi, firewalls og fjernadgang sat op, så medarbejdere kan arbejde hvor som helst, og kun de rette kommer ind.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har administreret 30 Ubuntu Linux VPS-instanser — 30 Linux-servere driftet, med en plan for gendannelse

Har administreret 30 Ubuntu Linux VPS-instanser Kontakt os →

Hvorfor det betyder noget

En protokol er det, der gør begge ender udskiftelige. Enhver klient, der er skrevet efter dokumentet, kan tale med enhver server, der er skrevet efter det, så en læser kan skifte software, og en server kan skrives om i et andet sprog, uden at den anden side mærker det.

Protokollen afgør også meget af sikkerheden, før der findes nogen kode. Hvilken side der åbner forbindelsen, afgør, hvad der skal lukkes ind: en server, der venter på klienter, skal have en port åben mod verden, mens et program, der kun ringer ud, ikke behøver nogen. Hvad beskederne bærer, afgør, hvad der kan logges,

tælles eller lække. En forespørgsel, der angiver en sti, kan logges efter sti, og det at skjule en læsers netværksadresse forhindrer ikke serveren i at tælle forespørgsler. En protokol, der kører uden kryptering, viser hver forespørgsel til hvert netværk, den passerer.

Endelig overlever protokoller de programmer, der taler dem. Implementeringer udskiftes med få års mellemrum, mens samtalen fortsætter i årtier, og enhver adfærd, som udrullede programmer er afhængige af, bliver en del af den, uanset hvad dokumentet siger. Internet Architecture Board har skrevet om begge halvdele af det problem: fejl, der tolereres, indtil de bliver standarden, og plads til vækst, som ingen bruger, indtil den holder op med at virke.

Sådan ser det ud i praksis

Dette site svarer på fire protokoller, fra én mappe med filer på én server: HTTP, Gemini, Gopher og en onion-tjeneste over Tor. Siden beskriver Gopher som en menuprotokol fra 1991 med ren TCP, ingen kryptering og ingen scripting, og viser en hel forespørgsel skrevet i hånden:

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

Den kalder den linje hele protokollen. Nedenunder skriver én udrulning én mappe, og fire hoveddøre læser den, så ingen af dem er en version af sitet, der vedligeholdes for sig.

Firewallregler verificeres efter position snarere end tilstedeværelse, fordi et pakkefilter evaluerer en ordnet liste og standser ved det første match. Én af tjekkets regler handler om protokoller: der findes præcis én regel pr. port og altid med en protokol, for en regel uden protokol udvider i det stille overfladen. Den erklærede offentlige overflade er en kort liste på fem porte, efterprøvet som en helhed, så en port, der dukker op uden at være erklæret, får tjekket til at fejle.

Den fjerde hoveddør er et onion-spejl, og dens protokol har formet både firewallen og optællingen. Anonymitetsdæmonen når nettet ved at oprette udgående kredsløb, så der kommer ingen indgående forbindelser, ingen port åbnes, og ingen af de to firewall-lag har noget at synkronisere. Gemini-

forespørgsler og Gopher-forbindelser tælles pr. dag uden adresse og uden sti. Onion-adressen har ingen tæller og intet flag til at tilføje en, fordi dette site bevidst undlader at tælle dette publikum.

Hvor det går galt

En tolereret fejl bliver standarden. Robusthedsprincippet, vær konservativ i det, du sender, og liberal i det, du tager imod, skulle hjælpe programmer med at arbejde sammen. Anvendt uden grænser lader det en fejlbehæftet implementering blive udgivet og brede sig. Den næste implementering må acceptere den samme fejl for at fungere med den, og fejlen bliver en de facto-standard, som alt nyt skal være kompatibelt med, fejl for fejl. RFC 9413 siger det ligeud: de udrullede implementeringer bliver en overensstemmelsestest, der kan tilsidesætte specifikationen. Afvis det, specifikationen forbyder, så tidligt i en protokols liv som muligt, og gør fejlen synlig for nogen, der kan rette den.

To programmer er uenige om, hvor en besked slutter. På en bytestrøm er grænsen der, hvor protokollen siger, den er. En protokol, der tilbyder to måder at sige det på, eller en parser, der godtager en sjusket udgave, lader en proxy og serveren bag den læse de samme bytes som forskellige beskeder. Det, den ene opfatter som halen af en forespørgsel, opfatter den anden som en ny forespørgsel, proxyen aldrig har tjekket. Tillad én regel for afgrænsning, afvis alt tvetydigt, og sørg for, at hvert led fortolker den på samme måde.

En regel nævner en port, men ingen protokol. Et portnummer betyder intet, før en transport er nævnt, og afhængigt af værktøjet kan en firewallregel, der kun er skrevet med et tal, åbne porten for både TCP og UDP. Den transport, der ikke var brug for, står så åben for det, der måtte lytte der senere. Skriv protokollen ind i hver regel, og sammenlign hele den åbne overflade med en erklæret liste.

Pladsen til vækst holder op med at virke. Hvis ingen udrullet afsender nogensinde bruger en protokols udvidelsespunkter, bliver der skrevet modtagere, der afviser det, de aldrig har set, og den første rigtige udvidelse går i stykker mod dem. Brug udvidelsespunkterne fra begyndelsen, også med værdier, der intet betyder, så tolerancen bliver testet hele tiden i stedet for at blive taget for givet.

Læs mere

- NIST CSRC-ordliste: protocol — definitionerne fra NIST's sikkerhedspublikationer, hver med det dokument, den stammer fra.
- RFC 9293: Transmission Control Protocol — den gældende TCP-standard, herunder bytestrømmen uden beskedgrænser.
- RFC 768: User Datagram Protocol — hele UDP på tre sider, fra 1980.
- RFC 1436: The Internet Gopher Protocol — en informativ beskrivelse af en hel applikationsprotokol, med eksempler på sessioner.
- RFC 9110: HTTP Semantics — betydningen af forespørgsler, svar, metoder og statuskoder, uafhængigt af den version, der bruges på ledningen.
- RFC 9413: Maintaining Robust Protocols — Internet Architecture Board om, hvorfor tolererede fejl slår rod.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring udrulningsforløbet, driftsrisiciene og dem, der har vagten. Vi kan afgrænse en ændring med kontroller, en vej tilbage og tydeligt ansvar.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er en protokol?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/protocol/