Engineer

Protokoll

de avtalte reglene to programmer følger for å snakke sammen

Hva det er

En protokoll er en skriftlig avtale om en samtale mellom programmer: hvordan hver melding ser ut, hva den betyr, hvilken side som snakker først, hva som kan følge etter hva, og hva hver side gjør når den andre sier noe uventet. To programmer skrevet av forskjellige folk i forskjellige språk kan arbeide sammen fordi begge er bygd mot det samme dokumentet, ikke mot hverandre.

Ordlisten til NIST samler flere definisjoner fra organisasjonens egne publikasjoner, og de deler en kjerne: regler for formatet på meldingene og prosedyrene rundt dem, som to eller flere parter følger for å nå et felles mål. Én av dem nevner den halvdelen som er lett å glemme: reglene beskriver rekkefølgen på meldingene like mye som oppbygningen av dem. En melding i riktig format på feil tidspunkt ødelegger samtalen like sikkert som en som er feil formet.

Spesifikasjoner som publiseres som RFC-er, uttrykker reglene sine med nøkkelord i store bokstaver. MUST er et absolutt krav. SHOULD tillater et avvik bare når følgene er forstått og veid fullt ut. MAY markerer noe som virkelig er valgfritt. Siden 2017 har disse ordene den betydningen bare med store bokstaver, så en «must» med små bokstaver i det samme dokumentet er vanlig engelsk. Avstanden mellom SHOULD og MUST er avstanden mellom et program som er uvanlig, og et som er feil.

Protokoller ligger i lag. En applikasjonsprotokoll, den delen som vet om sider, menyer eller e-post, går som regel oppå en transportprotokoll som har én eneste jobb: å flytte byte mellom to maskiner. TCP, transporten de fleste av dem bruker, gir det spesifikasjonen kaller «a reliable, in-order, byte-stream service»: bytene kommer fram hele og i den rekkefølgen de ble sendt. Den lover samtidig ingen sammenheng mellom bolkene den ene siden skriver, og bolkene den andre siden

leser. En strøm har ingen meldinger i seg, så hver protokoll oppå må markere hvor dens egne meldinger slutter: med et linjeskift, en lengde sendt på forhånd, at tilkoblingen lukkes, eller en annen avtalt markør. UDP gjør den motsatte handelen. Den bærer separate datagrammer med et minimum av mekanikk, og verken levering eller vern mot duplikater er garantert. Hver transport skiller tjenestene på én maskin fra hverandre med en nettverksport, og TCP og UDP nummererer portene sine hver for seg.

Resten av en protokolls karakter kommer fra noen få valg hvert design gjør, med vilje eller ikke:

- **Hvem som snakker først.** Noen servere hilser på klienten, andre venter i stillhet på en forespørsel, og når begge sider venter, skjer det ingenting før en tidsgrense går ut.
- **Hva serveren husker.** I en tilstandsløs protokoll har hver forespørsel med seg alt som trengs for å svare på den. I en tilstandsfull protokoll avhenger betydningen av en melding av det som kom før den på den samme tilkoblingen.
- **Hvordan den vokser.** Versjonsnumre, opsjonskoder og felt som skal ignoreres når de er ukjente, er rommet en protokoll holder åpent for sin egen framtid.

Bruk begrepet i en beslutning

Bruk «Hva er en protokoll?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Wi-Fi, brannmurer og fjerntilgang satt opp slik at de ansatte kan jobbe hvor som helst, og bare de rette personene slipper inn.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Administrerte 30 Ubuntu Linux-VPS-instanser — 30 Linux-servere administrert, med gjenoppretting planlagt på forhånd

Administrerte 30 Ubuntu Linux-VPS-instanser Ta kontakt →

Hvorfor det er viktig

En protokoll er det som gjør begge ender utskiftbare. En hvilken som helst klient skrevet etter dokumentet kan snakke med en hvilken som helst server skrevet etter det, så en leser kan bytte programvare, og en server kan skrives om i et annet språk uten at den andre siden merker det. Programmer som bare er enige fordi de ble bygd sammen, kan ikke endres hver for seg.

Protokollen avgjør også mye av sikkerheten før det finnes en eneste kodelinje. Hvilken side som åpner tilkoblingen, bestemmer hva som må slippes inn: en server som venter på klienter, trenger en port som er åpen mot verden, mens et program som bare ringer ut, ikke trenger noen. Hva meldingene bærer, avgjør hva som kan logges, telles eller lekke. En forespørsel som navngir en sti, kan logges per sti, og det å skjule en lesers nettverksadresse hindrer ikke serveren i å telle forespørsler. En protokoll som kjører uten kryptering, viser hver forespørsel til hvert nettverk den krysser.

Til sist lever protokoller lenger enn programmene som snakker dem. Implementasjoner byttes ut med noen års mellomrom, mens samtalen fortsetter i tiår, og hver oppførsel som programmer i drift lener seg på, blir en del av den, uansett hva dokumentet sier. Internet Architecture Board har skrevet om begge halvdelene av det problemet: feil som tåles til de blir standarden, og rom for vekst som ingen bruker før det slutter å virke.

For et team betyr det at valget av protokoll, eller valget om å lage en egen, hører til de beslutningene som er tyngre å endre enn koden. Et program kan skrives om. En samtale som andre allerede har bygd klienter for, må dere leve med så lenge de klientene finnes.

Slik ser det ut i praksis

Dette nettstedet 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 menyprotokoll fra 1991, med ren TCP, uten kryptering og uten skripting, og viser en hel forespørsel skrevet for hånd:

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

Siden kaller den linjen hele protokollen. Under overflaten skriver én produksjonssetting én mappe, og fire inngangsdører leser fra den, så ingen av dem er en kopi som vedlikeholdes for seg og kan gli fra de andre.

Brannmurreglene verifiseres etter plassering, ikke etter tilstedeværelse, fordi et pakkefilter evaluerer en ordnet liste og stopper ved første treff. Én av reglene i sjekken handler om protokoller: det finnes nøyaktig én regel per port, og alltid

med protokoll, fordi en regel uten protokoll utvider flaten i det stille. Den deklarerte offentlige flaten er en kort liste med fem porter, kontrollert som en helhet, så en port som dukker opp uten å være deklarert, får sjekken til å feile.

Den fjerde døren er et onion-speil, og protokollen formet både brannmuren og rapporteringen. Daemonen som kobler verten til anonymitetsnettverket, når det ved å opprette utgående kretser, så ingenting kobler seg inn utenfra, ingen port åpnes, og ingen av brannmurlagene har noe å synkronisere. Gemini-forespørsler og Gopher-tilkoblinger telles per dag uten adresse og uten sti. Onion-tjenesten har ingen teller og ingen innstilling for å legge til en, fordi dette nettstedet bevisst lar være å telle dette publikummet.

Hvor det går galt

Det som tåles, blir standarden. Robusthetsprinsippet, vær konservativ i det du sender og liberal i det du tar imot, var ment å hjelpe programmer å fungere sammen. Brukt uten grense lar det en mangelfull implementasjon komme ut og spre seg. Neste implementasjon må godta den samme feilen for å fungere med den, og feilen blir en de facto-standard som alt nytt må være kompatibelt med, feil for feil. RFC 9413 sier det rett ut: implementasjonene som er i drift, blir en samsvarstest som kan gå foran spesifikasjonen. Avvis det spesifikasjonen forbyr, så tidlig i protokollens liv som mulig, og gjør feilen synlig for noen som kan rette den.

To programmer er uenige om hvor en melding slutter. På en bytestrøm går grensen der protokollen sier at den går. En protokoll som gir to måter å si det på, eller en parser som godtar en slurvete variant, lar en proxy og serveren bak den lese de samme bytene som forskjellige meldinger. Det den ene leser som halen på en forespørsel, leser den andre som en ny forespørsel proxyen aldri sjekket. Tillat én regel for innramming, avvis alt som er tvetydig, og la hvert ledd tolke den på samme måte.

En regel nevner en port, men ingen protokoll. Et portnummer betyr ingenting før en transport er nevnt, og avhengig av verktøyet kan en brannmurregel som bare har et nummer, åpne det på både TCP og UDP. Tjenesten som trengte den ene, får den, og den andre står åpen for det som måtte lytte der senere. Skriv protokollen inn i hver regel, og sammenlign hele den åpne flaten med en deklarert liste.

Rommet for vekst slutter å virke. Hvis ingen avsender i drift noen gang bruker utvidelsespunktene i en protokoll, blir det skrevet mottakere som avviser det de aldri har sett, og den første ekte utvidelsen knekker mot dem. Jo lenger en slik mottaker er i drift, jo vanskeligere er den å rette. Bruk utvidelsespunktene fra starten, også med verdier som ikke betyr noe, slik at toleransen testes hele tiden i stedet for å bli tatt for gitt.

Les mer

- NIST CSRC glossary: protocol — definisjonene som brukes i sikkerhetspublikasjonene til NIST, hver med dokumentet den kommer fra.
- RFC 9293: Transmission Control Protocol — den gjeldende TCP-standarden, med bytestrømmen som ikke har meldingsgrenser.
- RFC 768: User Datagram Protocol — hele UDP på tre sider, fra 1980.
- RFC 1436: The Internet Gopher Protocol — en informativ beskrivelse av en hel applikasjonsprotokoll, med eksempler på økter.
- RFC 9110: HTTP Semantics — hva forespørsler, svar, metoder og statuskoder betyr, uavhengig av versjonen som går over linja.
- RFC 9413: Maintaining Robust Protocols — Internet Architecture Board om hvorfor det å tåle feil sementerer dem.

Ordliste

Velg ut fra deres faktiske rammer

Ta med utrullingsløpet, driftsrisikoene og dem som har vakt. Vi kan avgrense en endring med kontroller, en vei tilbake og tydelig ansvar.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er en protokoll?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/protocol/