IDATT2104 - Datakom Oblig 2

Jakob Grønhaug (jakobkg@stud.ntnu.no)

14. mars 2023

Innhold

Websocket

Teori

Forskjeller mellom HTTP og Websocket

Det er flere likheter og forskjeller mellom HTTP og Websocket. For begynner alle Websocket-forbindelser med et handshake over HTTP, så man kan ikke implementere Websocket uten å først ha implementert ihvertfall noen deler av HTTP.

Den viktigste forskjellen er at HTTP er bygd på en rigid struktur der hver interaksjon følger formatet `klient sender forespørsel→tjener sender svar`. I HTTP er det et 1:1-forhold mellom forespørsler fra klienten og svar fra tjeneren, og det er alltid klienten som tar initiativ til en slik interaksjon. Dette har visse begrensninger, la oss for eksempel si en klient venter på at tjeneren skal bli ferdig med en større databehandlingsjobb, eller venter på en tilstandsendring av noen form.

Siden HTTP ikke tillater at tjeneren tar initiativ til sending av data til en klient må klienten selv spørre tjeneren med jevne mellomrom 'er du ferdig enda? er du ferdig enda? er du ferdig enda?'. Dette kan føre til at klienten må generere mange unødvendige forespørsler, og tjeneren må bruke tid og ressurser på å svare 'Nei jeg er ikke ferdig enda' i stedet for å kunne vie disse ressursene til å utføre jobben klienten venter på. Websocket har ikke denne samme strukturen med 'forespørsel→respons' og kan dermed unngå den samme problemstillingen. Etter at handshake er gjennomført kan tjeneren sende klienten varsling om at ny data/tilstand er tilgjengelig uten at klienten trenger å spørre gjentatte ganger!

Sikkerhetsmekanisme i Websocket

TODO

Dokumentasjon

Oppkobling/handshake

Ifølge Websocket-standarden (RFC 6455) er en Websocket-kobling noe en HTTP-klient og HTTP-tjener blir enige om å opprette. Først sender klienten en GET-forespørsel, og inkluderer feltet `Upgrade: websocket` i headeren til forespørselen. Denne forespørselen skal også

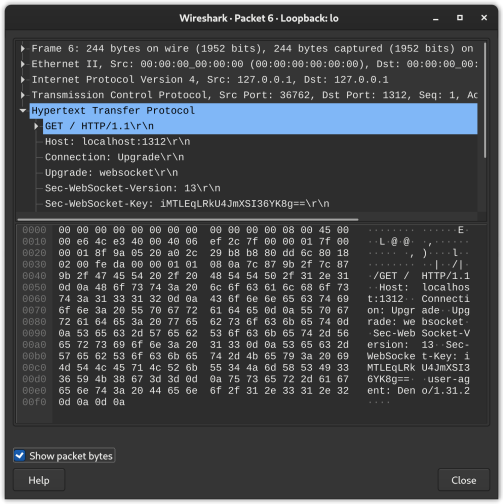
spesifisere hvilken versjon av Websocket som skal brukes, og inneholde en tilfeldig nøkkel[1] `Sec-Websocket-Key` på 16 byte, kodet i Base64, som tjeneren skal bruke for å verifisere svaret sitt.

Tjeneren skal så svare med en HTTP-respons med status `101 Switching Protocols` som må inneholde header-feltene `Upgrade: websocket` og `Connection: upgrade`, og et header-felt `Sec-Websocket-Accept`. Dette feltet er spesielt, og verdien som skal puttes her må utledes fra den tilfeldige nøkkelen som klienten sendte i sin del av handshaket. Tjeneren skal ta nøkkelen som klienten sendte og legge til teksten `258EAFA5-E914-47DA-95CA-C5AB0DC85B11` på slutten av den. Denne teksten er alltid den samme, og er oppgitt i Websocket-spesifikasjonen. Om klienten sender nøkkelen `iMTLEqLRkU4JmXSI36YK8g==` skal tjeneren altså ende opp med `iMTLEqLRkU4JmXSI36YK8g==258EAFA5-E914-47DA-95CA-C5AB0DC85B11`. Deretter skal tjeneren bruke SHA1-algoritmen til å beregne hashen til denne teksten. En SHA1-hash er alltid 20 bytes lang, og hashen av eksempel-nøkkelen blir

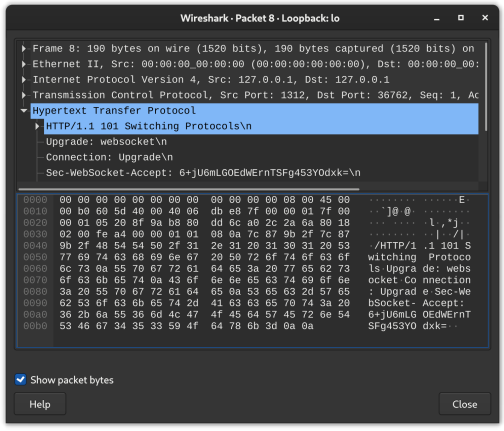
```
0xEB, 0xE8, 0xD4, 0xEA, 0x62, 0xC6, 0x38, 0x47, 0x56, 0x12,
0xB9, 0xD3, 0x48, 0x58, 0x38, 0xE7, 0x76, 0x0E, 0x77, 0x19
```

3	0.000106994	127.0.0.1	127.0.0.1	TCP	74	36762 → 1312 [SYN] Seq=0 Win=65495 Len=0
4	0.000115363	127.0.0.1	127.0.0.1	TCP	74	1312 → 36762 [SYN, ACK] Seq=0 Ack=1 Win=6
5	0.000125582	127.0.0.1	127.0.0.1	TCP	66	36762 → 1312 [ACK] Seq=1 Ack=1 Win=65536
6	0.000168630	127.0.0.1	127.0.0.1	HTTP	244	GET / HTTP/1.1
7	0.000172021	127.0.0.1	127.0.0.1	TCP	66	1312 → 36762 [ACK] Seq=1 Ack=179 Win=6540
8	0.000412835	127.0.0.1	127.0.0.1	HTTP	190	HTTP/1.1 101 Switching Protocols

(a) Websocket-handshake slik det fremstår i pakkelisten i Wireshark



(b) Klientens del av handshake



(c) Tjenerens del av handshake

Figur 1: Skjermbilder av Websocket-handshaket i Wireshark

[1] Denne tilfeldige nøkkelen, og senere maskering av meldinger fra klienter til tjenere, brukes for å unngå at proxyer og cache som kan ligge mellom klienten og tjeneren svarer 'på vegne av' tjeneren med en forhåndslagret kopi av en tidligere respons. Ved bruk av et tilfeldig element som nøkkelen i handshake og maskering av meldinger sørger man for at den faktiske dataen som sendes over nettverket er forskjellig hver gang selv om meldingen som ble sendt kanskje er den samme.

Tjeneren må så kode denne hashen som Base64, og denne Base64-strengen er det som skal sendes fra tjeneren i `Sec-Websocket-Accept`-feltet i headeren. I dette spesifikke eksempelet blir dette feltet `Sec-Websocket-Accept: 6+jU6mLGOEdWErnTSFg453YOdxk=`. Skjermbildene i figur 1 viser dette eksempel-handshaket i faktisk trafikk.

Meldinger fra klient til tjener

Når oppkoblingen er utført som beskrevet over er Websocket-forbindelsen opprettet og klar for trafikk! Både klient og tjener kan sende data over denne koblingen når de vil, med hovedforskjell at klienter alltid burde sende meldingene sine med maskering, mens en tjener ikke nødvendigvis trenger å gjøre det.

Meldinger over Websocket sendes i form av ett eller flere fragment, hvor alle fragmenter følger en bestemt struktur. I denne oppgaven var det kun krav om å implementere meldinger på ett fragment med meldingslengde på opp til 125 bytes. Slike meldinger har følgende struktur:

<table>
<tr><th></th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th></tr>
<tr><td>1</td><td>FIN</td><td colspan="3">RESERVERT</td><td colspan="4">MELDINGSTYPE</td></tr>
<tr><td>2</td><td>MASK</td><td colspan="7">MELDINGSLENGDE</td></tr>
<tr><td>3
4
5
6</td><td colspan="8">MASKERINGSNØKKEL</td></tr>
<tr><td>...</td><td colspan="8">MELDINGSDATA</td></tr>
</table>

Figur 2: Strukturen til en Websocket-datapakke