How to serve HTTP 1, 2, and 3 in Python

2019-08-26

The latest release of Hypercorn, 0.8.0, adds HTTP/2 prioritisation (completing HTTP/2 support) and introduces HTTP/3 support based on the latest draft specification, 22. This is made possible using the priority and aioquic libraries respectively. This allows any ASGI application to be served over HTTP/1, HTTP/2, and HTTP/3 without having to alter any code. If you aren't using an ASGI framework take a look at any of these great frameworks,

- BlackSheep
- Django
- FastAPI
- Quart
- Responder
- Sanic
- Starlette
- and more... (please remind me)

As I'm most familiar with Quart I'm going to install quart `pip install quart` and use this example in a file called `run.py`,

```
from quart import Quart

app = Quart(__name__)

@app.route("/")
async def index():
    return "<b>Hello World!</b>"
```

To get Hypercorn with HTTP/3 support you will need to install it including the `h3` extra, i.e. `pip install hypercorn[h3]`. Then you can run

```
hypercorn --bind localhost:8000 run:app
```

and get HTTP/1 responses from `http://localhost:8000`. To get HTTP/2 responses you will need to serve over HTTPS[1], which will require a SSL certificate. In production you should use Let's Encrypt, for development though a self signed certificate will work,

```
openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365 -nodes
```

which can be used with Hypercorn,

```
hypercorn --certfile cert.pem --keyfile key.pem --bind localhost:8000 run:app
```

allowing HTTP/2 responses from `https://localhost:8000`. To get HTTP/3 responses Hypercorn needs to be instructed to serve over a quic bind (with certificates), ideally alongside a HTTP/1 & HTTP/2 TCP bind. This is done via,

```
hypercorn --quic-bind localhost:4433 --certfile cert.pem --keyfile key.pem --bind localhost:8000 run:app
```

allowing HTTP/3 responses from `https://localhost:4433` and HTTP/2 from `https://localhost:8000`.

Finding a HTTP/3 client

As of August 2019 it is difficult to find a HTTP/3 client. HTTP/3 support in curl is currently in development (although it does work) and browsers haven't yet released any support. Happily aioquic is installed with Hypercorn and it's example client can be used to make a HTTP/3 request,

```
wget
https://raw.githubusercontent.com/aiortc/aioquic/62df497eaf4a6b8bd28dbf8d8f801daa452ea062/examples/http3_clien
t.py
python http3_client.py https://localhost:4433/
```

You can also view this website over HTTP/2 at `https://pgjones.dev` and HTTP/3 at `https://pgjones.dev:4433`.

HTTP/3 caveats

The HTTP/3 specification is still under development, so this isn't production ready. This is why it is an optional extra in Hypercorn.

Bonus WebSockets

Hypercorn supports WebSockets over HTTP/1, HTTP/2, and HTTP/3 without any additional configuration. It does so using the great wsproto Sans-IO library. A quick example using quart is (in a file called `ws_run.py`),

```
from quart import Quart, websocket

app = Quart(__name__)

@app.websocket("/ws")
async def ws():
    while True:
        data = await websocket.receive()
        await websocket.send(data)
```

which can be run as,

```
hypercorn --quic-bind localhost:4433 --certfile cert.pem --keyfile key.pem --bind localhost:8000 ws_run:app
```

allowing HTTP/1 & HTTP/2 WebSocket connections at `wss://localhost/ws` and HTTP/3 WebSocket connections at `wss://localhost:4433/ws`.

Footnotes

1: Technically this isn't required, except that there are very few HTTP/2 clients that support unencrypted HTTP/2. Hypercorn will serve unencrypted HTTP/2 and supports the h2c upgrade process if you find one.