Advanced Network Programming 8
Control servers and hybrid protocols

Juliusz Chroboczek

15 April 2025

Advantages of client-server

Advantages of client-server :

- easier to implement
  client-server is a well-understood structure;
- easier to debug
  the server has global vision, it can serve as a point for logging and debugging;
- fast
  a single request is often enought to fetch all the required data.

Implementability and especially debugabbility are important properties that are sometimes neglected by theorists.

Reasons to do peer-to-peer

Client-server has multiple advantages.

But there are reasons to do peer-to-peer (p2p) :

- server infrastructure is costly the server needs to be paid for, the clients have already been paid for;
- better scaling properties
  a peer-to-peer system scales with demand
  (if done right: load is $O(\log n)$ instead of $O(n)$);
- no single point of failure
  able to survive attacks
  including legal attacks.

Pure p2p is dificult

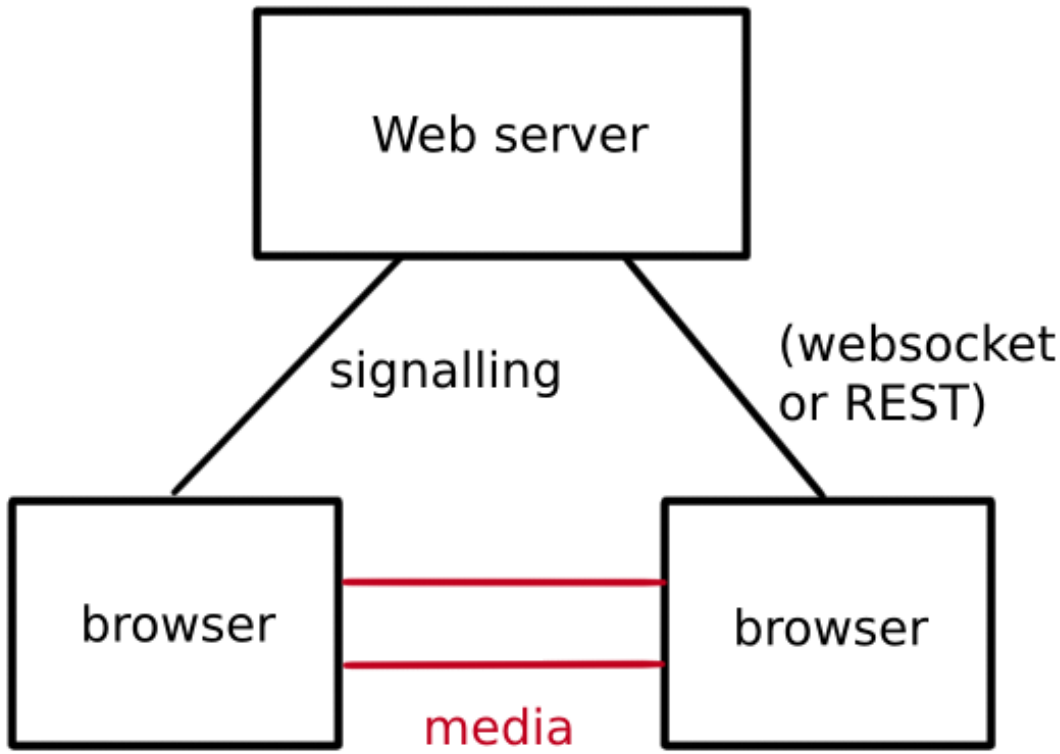
Some things are difficult in p2p:

- rendez-vous: how to find a peer?
- authentication and exchange of cryptographic keys for encryption
  how do you perform encrypted communication without an external source of trust?
- NAT traversal
  how to synchronise if you cannot communicate yet?

Hybrid protocols and control servers

Work around difficulties: hybrid protocols:

- client-server control protocol;
- peer-to-peer data protocol.



The control protocol

In a hybrid protocol, there is a client-server subprotocol, the control protocol.
Solves the problems that are difficult in pure p2p:

- rendez-vous: peers register their IP address with the server, rendez-vous is a single request to the server;
- exchange of cryptographic keys
  - for peer authentication;
  - for end-to-end encryption;

The control protocol: the server is trusted

The server must be trusted:

- it has full knowledge of the adjacency graph;
- can perform MITM on key exchange unless there is an external source of trust but then, end-to-end is also possible in pure client-server.

Other disadvantages:

- the server is a single point of failure vulnerable to attacks;
- load on the server is $O(n)$ or even $O(n^2)$, but with a very small constant.

Examples

Two examples:

- BitTorrent file transfer;
- peer-to-peer videoconferencing.

BitTorrent

BitTorrent is a file transfer protocol.

A “torrent” is identified by an *Info-Hash*.
Transferred out-of-band.

Double purpose:

- serves as a rendez-vous point;
- used for data integrity.

Peers that are insterested in a given file are called a *swarm*. They exchange file data using a peer-to-peer protocol.

BitTorrent (2)

Original BitTorrent consists of two sub-protocols:

- client-server (HTTP-based) tracker protocol
  used only for rendez-vous;
- p2p (TCP-based) BitTorrent protocol
  used for data transfer.

The BitTorrent tracker protocol

Tracker protocol:

- client sends request to tracker, includes:
  - info-hash;
  - client port number;
- server replies with the socket addresses (IP + port) of up to 50 peers in the swarm;
- the server remembers a peer's address for 30 min; in order to be visible, peers need to recontact the server every 28 minutes.

The BitTorrent data transfer protocol

Peers use information obtained from the tracker to connect to each other.

Once a peer has connected, data transfer protocol:

- initial handshake: the connecting peer proves knowledge of the info-hash;
- data transfer, integrity protected by the info-hash (and the torrent file, a Merkle tree would be better).

Evolution of BitTorrent

BitTorrent is an extensible protocol.

Two notable extensions:

- Kademlia DHT, performs rendez-vous in a purely decentralised manner;
- Peer Exchange, learns new peers using the gossip algorithm;

BitTorrent has evolved to be a pure peer-to-peer protocol.

Reason: the tracker is a single point of failure, and therefore susceptible to (legal) attacks.

The reason is *not* load on the tracker: the tracker handles O(n) load, but constants make it negligible.

Pure p2p is difficult, a good reason is needed.

Videoconferencing protocols

Two kinds of videoconferencing protocols.

Client-server videoconferencing
Client sends a single data stream to the server, which resends it to all $n - 1$ participants of the group.

Scales well to large groups (scale up the server).
Ex: Galene, Zoom, Cisco Webex, etc.

Peer-to-peer videoconferencing
Every client sends $n - 1$ video streams to all participants of the group.

Better latency, lower server-side cost.

Optimal for 1-1 communication, doesn't scale well to larger groups.
Ex: Google Duo

Peer-to-peer videoconferencing

Peer-to-peer videoconferencing uses a control server :

- authentication and authorisation;
- exchange of cryptographic keys:
  - mutual (peer) authentication;
  - end-to-end encryption;
- exchange of session information (number of streams, codecs, etc.);
- session-layer negotiation (ICE):
  - negotiation of transport parameters (port numbers);
  - NAT traversal (STUN);
  - fallback to UDP tunnelling, to TCP tunnelling (TURN).

Comparison

The BitTorrent tracker is minimal:
almost anything that can be done peer-to-peer is avoided.

The videoconferencing server is maximal: everything except data transfer is carried by the server.

	BitTorrent	**videoconferencing server**
server authentication	info-hash in URL	password, token, etc.
session information	n/a	SDP through server
p2p authentication	info-hash in handshake	key exchange through server
end-to-end encryption	ad-hoc	key exchange through server
enb-to-end integrity	out-of-band info-hash	key exchange through server
transport parms	through server	through server
NAT traversal	peer-to-peer	through server (STUN)
tunnelling fallback	n/a	negotiated through server

This is due to different constraints:

- BitTorrent is sometimes used for controversial traffic, resist attacks, minimise cost;
- for videoconferencing, low latency is more important than attack resistance
  reliability is essential.