

WebRTC & WebSockets — Cheat Sheet

TRANSPORT CHOOSER

Transport	Dir	Reliab.	Use
Polling	pull	reliable	dashboards, fallback
Long poll	push	reliable, gaps	fallback
SSE	S→C	ordered, Last-Event-ID	feeds, AI text
WebSocket	2-way	TCP ordered	chat, signalling
RTC media	2-way	lossy, adaptive	calls
Data chan.	2-way	configurable	game state, file
WebTransp.	2-way	streams + datagrams	games (Safari 26.4+)

Simplest that fits; move on only when measured. Media → WebRTC + WS signalling. XMPP for federation.

WEBSOCKET HANDSHAKE & FRAME (RFC 6455)

```
GET /chat HTTP/1.1
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: <16 B base64>
Sec-WebSocket-Version: 13
Origin: https://app.example
→ 101 Switching Protocols
Sec-WebSocket-Accept: base64(sha1(key+G
UID))

GUID 258EAFA5-E914-47DA-95CA-
C5AB0DC85B11 . HTTP/2: RFC 8441, HTTP/3:
RFC 9220 (CONNECT, :protocol).
Frame: FIN 1b, RSV1-3 (RSV1 = permessag
e-deflate, RFC 7692), opcode 4b, MASK 1b, len
7b (126 → +16b, 127 → +64b), masking key 4 B
(client → server always masked).
Opcodes: 0 cont · 1 text · 2 binary · 8 close · 9
ping · 10 pong. Control frames ≤125 B, never
fragmented.
```

CLOSE CODES

1000 normal	1001 going away
1002 protocol error	1003 bad data type
1007 bad payload	1008 policy
1009 too big	1010 extension
1011 server error	1012 restart
1013 try later	4000-4999 app

1005, 1006, 1015 are never sent on the wire. Browsers may send only 1000 and 3000-4999. Chatterbox: 4401 auth, 4429 rate.

BROWSER WEBSOCKET API

```
const ws = new WebSocket("wss://h/ws",
["chatterbox.v1"]);
ws.binaryType = "Arraybuffer";
ws.onopen = () => ws.send(JSON.stringify(m));
ws.onmessage = e => e.data;
ws.onclose = e => e.code, e.reason, e.w
asClean;
ws.close(1000, "bye");
ws.readyState // 0 CONNECTING 1 OPEN 2
CLOSING 3 CLOSED
ws.bufferedAmount // backpressure: wait
to drain

No custom headers: token in cookie, one-time
ticket, first message. WebSocketStream :
Chromium 124+ only.
```

RECONNECT & HEARTBEAT

```
const d = Math.min(30000, 500 * 2 ** n)
* (0.5 + Math.random() / 2); //
jitter
setTimeout(connect, d); // reset n on
open
setInterval(() => ws.send('{"type":"pin
g"}'), 25000);
// heartbeat < shortest idle timeout (A
LB 60 s)

Resume with last seen id; dedupe by client id.
SSE: EventSource reconnects itself.
```

SIGNALR (ASP.NET CORE)

```
builder.Services.AddSignalR();
app.MapHub<ChatHub>("/chat");
Clients.Group(room).SendAsync("msg",
m);

Auth [Authorize], ?access-token=.
Reconnect withAutomaticReconnect(),
stateful reconnect opt-in. Scale: Redis
backplane (AddStackExchangeRedis) or Azure
SignalR Service.
```

SOCKET.IO (NODE) · PYTHON-SOCKETIO

```
npm i socket.io@4.8.4
io.on("connection", s => s.join(r));
io.to(r).emit("msg", m);
io.use((s, next) => check(s.handshake.a
uth));
# pip install python-socketio==5.17.0
sio = socketio.AsyncServer(async_mode
="asgi")
await sio.emit("msg", m, room=r)

Acks, rooms, namespaces, connection state
recovery. Scale: Redis adapter /
AsyncRedisManager + sticky sessions.
Engine.IO v4, allowEIO3 only to migrate.
```

UWEBSOCKETS.JS · WS

```
npm i uNetworking/uWebSockets.js#v20.7
1.0
uWS.App().ws("/*", { upgrade, open(ws)
{ws.subscribe("room"),
message(ws, m, bin){ app.publish("roo
m", m, bin) } })

Auth in upgrade (read headers, res.upgrade).
Defaults: 16 KB payload, idle 120 s,
backpressure 64 KB. Not on npm; custom
licence. No cross-process scaling: add Redis
bridge.
```

DJANGO CHANNELS · SPRING WEBFLUX

```
pip install "channels[daphne]"
class C(AsyncWebsocketConsumer): ...
await self.channel_layer.group_send(g,
{"type":"chat.msg"})
// Spring: WebSocketHandler.handle(sess
ion)
Sinks.many().multicast().directBestEffo
rt()

Channels: AuthMiddlewareStack,
AllowedHostsOriginValidator,
channels_redis layer (in-memory = tests).
WebFlux: spring-boot-starter-webflux,
Spring Security, Redis pub/sub; STOMP only on
Servlet stack.
```

ATMOSPHERE · SWOOLE · REVERB

```
@ManagedService(path="/chat") class Cha
t {...} // 4.0.76
$s = new Swoole\WebSocket\Server("0.0.
0.0", 9501);
$$->on("message", fn($,$f)->$$->push
($fd, $m));
php artisan install:broadcasting --reve
rb
php artisan reverb:start // Echo.priv
ate(...)

Atmosphere: Broadcasters, fallbacks (SSE, long
poll), atmosphere.js 5, Redis/Kafka. Swoole:
coroutines, Swoole\Table, handshake hook
auth, Redis. Reverb: Pusher protocol,
routes/channels.php,
REVERB_SCALING_ENABLED.
```

WEBRTC SEQUENCE & PERFECT NEGOTIATION

```
getUserMedia → addTrack → createOffer →
setLocalDescription → (signal) →
setRemoteDescription → createAnswer →
setLocalDescription → (signal). Trickle ICE:
onicecandidate ↔ addIceCandidate. Then
ICE → DTLS → SRTP/SCTP.

pc.onnegotiationneeded = async()->{ mak
ingOffer=true;
await pc.setLocalDescription(); send(p
c.localDescription);}
// on description: collision = makingOf
fer || state!=="stable"
// ignore = !polite && collision; polit
e rolls back (implicit)
```

CAPTURE & DEVICES

```
getUserMedia({audio:{echoCancellation:t
rue},
video:{width:{ideal:1280},height:idea
l:720},
frameRate:{max:30},facingMode:"user",d
eviceId});
getDisplayMedia({video:true,audio:tru
e});
enumerateDevices(); devicechange; trac
k.stop()

HTTPS only. Fake devices for tests: ---use-
fake-device-for-media-stream ---use-fake-
ui-for-media-stream. Switch camera with
sender.replaceTrack().
```

DATA CHANNELS

```
pc.createDataChannel("chat") // ordere
d, reliable
pc.createDataChannel("state",{ordered:f
alse,maxRetransmits:0})
// negotiated:true,id:n // maxPacketL
ifeTime alt.

Chat/files: reliable, ordered; game input/state:
unordered, maxRetransmits:0. Watch
bufferedAmount /
bufferedAmountLowThreshold; keep
messages <16 KB.
```

SIMULCAST, SVC, STATS

```
pc.addTransceiver(track,{direction:"sen
donly",sendEncodings:[
{rid:"q",scaleResolutionDownBy:4,maxBi
trate:150e3},
{rid:"h",scaleResolutionDownBy:2,maxBi
trate:500e3},
{rid:"f",maxBitrate:1.2e6}])
scalabilityMode:"L1T3" // SVC

getStats(): inbound-rtp packetsLost jitter
framesPerSecond; candidate-pair
currentRoundTripTime
availableOutgoingBitrate;
qualityLimitationReason.
```

TOPOLOGIES

	Mesh	SFU	MCU
Streams	N-1 up, N-1 down	1 up, N-1 down	1 up, 1 down
Server	none	forward	decode+mix
E2EE	easy	SFrame	no
Fit	≤4	5-50+	dial-in

SFU: simulcast, last-N, active speaker (RFC 6464 audio level).

ICE, STUN, TURN

Candidates: host, srflx (STUN), relay (TURN). Order tried by priority; iceTransportPolicy:"relay" to force. Restart: pc.restartIce().

```
# coturn
use-auth-secret
static-auth-secret=change-me
user = expiry:name; pass = base64(HMAC-
SHA1(secret,user))
ports 3478 udp/tcp, 5349 tls, relay ran
ge

Ephemeral creds, refresh at 80% TTL. Deny
private peer ranges (abuse). eturnal is the
alternative.
```

MEDIA SERVERS

mediasoup	SFU library, you build signalling
LiveKit	full SFU, tokens, SDKs, egress
Janus	gateway + plugins (VideoRoom, SIP)
Jitsi	ready-made product, Videobridge
Pion (Go)	build peers, SFUs
aiortc	bots, test peers (not an SFU)

WHIP (RFC 9725): POST SDP offer, application/sdp → 201 + Location. **WHEP**: playback, still a draft. E2EE: SFrame (RFC 9605), RTCTrtpScriptTransform. Record: MediaRecorder or egress.

XMPP ESSENTIALS

JID local@domain/resource. Stanzas <message/> <presence/> <iq type=get|set|result|error/>. RFC 6120/6121.

0045 MUC	0313 MAM
0280 Carbons	0198 Stream Mgmt
0184 receipts	0333 markers
0085 typing	0308 correction
0424 retraction	0444 reactions
0363 upload	0357 push
0384 OMEMO	0166 Jingle

RFC 7395: subprotocol xmpp, <open xmlns="urn:ietf:params:xml:ns:xmpp-framing"/>. Paths: Prosody /xmpp-websocket :5280/5281; Openfire /ws/:7070/7443; MongooseIM /ws-xmpp; ejabberd ejabberd_http_ws.

```
new Strophe.Connection("wss://h/xmpp-we
bsocket")
client({service:"wss://h/xmpp-websocket",domain,username,password})

Jingle: session-initiate, session-accept,
transport-info, session-terminate.
```

CHAT & CALL CHECKLISTS

- Chat**: client ids (idempotent), server order per room, presence, typing, delivered/read receipts, paginated history, offline catch-up, edits, reactions, moderation, backplane
- Huddle**: lobby, mute/camera, device switch, screen share, active speaker, TURN, simulcast, record, E2EE, reconnect with ICE restart

GAME NETWORKING TIMELINE

- Tick**: fixed server step (20-60 Hz)
- Snapshot**: state at tick n, delta + binary
- Prediction**: client applies input at once
- Reconciliation**: rewind to server state, replay unacked inputs
- Interpolation**: render others a fixed delay (e.g. 100 ms) in the past
- Lag compensation**: server rewinds targets (cap ~200 ms)

SECURITY CHECKLIST

- Validate Origin (CSWSH); wss only
- Auth before upgrade or first frame; authorise each message
- Max payload, rate limit, connection caps per user/IP
- Short-lived tokens, no secrets in URLs or logs
- TURN: ephemeral creds, deny internal IPs
- permessage-deflate: length-oracle risk

SCALE & PROXY

Many nodes → backplane (Redis pub/sub, Streams, Kafka). Sticky sessions only for polling fallbacks. Per SFU worker ~500 consumers/core (plan 60%). TURN sized by relayed users.

```
proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
proxy_read_timeout 3600s;
```

OBSERVE & TEST

Metrics: connections, msgs/s, fan-out latency, close codes, ICE success, RTT, loss, jitter (Prometheus, OpenTelemetry). Tests: Playwright (page.on('websocket'), fake media), k6/Artillery/JMeter load, containerised XMPP, mobile: push not open sockets.

CHANGED SINCE OLDER TUTORIALS

- JSEP RFC 8829 → **9429**; Plan B → Unified Plan; addStream → addTrack
- createEncodedStreams → **RTCTrtpScriptTransform**
- adapter.js optional (v9)
- Socket.IO 2 → 4 (Engine.IO v4)
- SignalR 2 → Core, stateful reconnect, auth refresh
- Channels 4: no built-in Daphne
- Atmosphere 4 (JDK 21, atmosphere.js 5)
- Echo + Pusher → **Reverb**
- Spring 7: no Undertow
- BOSH → WebSocket (RFC 7395); Compliance Suites
- ws vs Node global WebSocket client