Getting of the main JS thread

Option A: Web Workers w/ transferable everything

Step 1: Make everything constructable in a web worker (or transferable)

- `IceTransport(?)`
- `DtlsTransport`
- `SctpTransport`
- `QuicTransport`
- `SourceBuffer`
- `RtpSender/RtpReceiver`

Step 2: Make MediaStreamTrack and/or HtmlMediaElement transferable
(something needs to escape the web worker)

Step 3: Run the pipeline in a web worker; communicate with ports (streams not needed)

Example (everything in worker)

```
// In main.js
var worker = new Worker("workers.js");
worker.onmessage = function(event) {
  signalIceCandidate(event.data);
};

// In worker.js
var ice = new RTCIceTransport(...);
var quic = new RTCQuicTransport(ice);
var ms = new MediaSource();
var mse = ms.appendSourceBuffer(...);
var parser = new TransformStream(parseMessage);
quic.receiveStreams().pipeThrough(parser).pipeInto(mse.appendStream());
...
self.postMessage(iceCandidate)
```

Option B: Web Workers w/ streaming MessagePorts

Step 1: Make WHATWG streams that cross worker boundary, like MessagePorts

Step 2: Run parts of the pipeline in a web worker

```
// Allow pipeThrough(port)
partial interface MessagePort {
  ReadableStream writable;
  WritableStream readable;
}
// Allow pipeThrough(worker)
partial interface Worker {
  ReadableStream writable;
  WritableStream readable;
}
```

Example (just transform in worklet)

```
// In main.js
var quic = ...;
var mse = ...;
var parser = new Worker("parser.js");
quic.receiveStreams().pipeThrough(parser).pipeInto(mse.appendStream());

// In worker.js
var parser = new TransformStream(parseMessage);
self.readable.pipeThrough(parser).pipeInto(self.writable)
```

Option C: TransformStream w/ worklet

Step 1: Make a special WorkletTransformStream

```
interface WorkletTransformStream {
  registerTransformerName(DOMString name);
  [SameObject] readonly attribute Worklet worklet;
}
[Global=(Worklet,TransformStreamWorklet),
 Exposed=TransformStreamWorklet]
interface TransformStreamWorkletGlobalScope
 : WorkletGlobalScope {
    void registerTransformer(DOMString name,
                             Transformer ());
};
```

Step 2: Attach S to the pipeline

(Serializer, Parser)

Example (just transform in worker)

```
// In main .js
var parser = WorkletTransformStream();
parser.worklet.addModule("parser.js");
parser.registerTransformerName("parser");
quic.receiveStreams().pipeThrough(parser).pipeInto(mse.appendStream());

// In "parser.js"
registerTransformer("parser", class {
    transform(chunk, controller) {
      controller.enqueue(parseMessage(chunk));
    }
});
```

Option D: Transferable Streams

https://github.com/whatwg/streams/blob/master/transferable-streams-explainer.md

TL;DR: `var w = new Worker("x.js"); w.postMessage(streams, streams)`

Example (transform and piping in worker)

```
// In main .js
var messages = quic.receiveStreams();
var chunks = mse.appendStream();
var parser = new Worker(parser.js');
parser.postMessage([messages, chunks], [messages, chunks]);

// In parser.js
onmessage = async (evt) => {
  var {messages, chunks} = evt.data;
  var parser = new TransformStream(parseMessage);
  messages.pipeThrough(parser).pipeInto(chunks)
};
```

Questions

Which of these is the easiest to implement?

If we could do any of them, which would be better for developers?

Do we still need WHATWG streams?