Goals

- Align with existing things (WS, Event Source, …)
- Re-use
- Reduce amount of speccing needed now
- Make the simple case simple, (OK if advanced cases are a bit more complex)
- Allow for extension later

Basic proposal

- PeerConnection set up to always offer(?) data
  - If other end says no, there will be no data
- Send method directly on PeerConnection
- Onmessage deliver a MessageEvent
- Initially support DOMString, blob and ArrayBuffer (as WS)

```
Interface PeerConnection {
    [...]
    // messaging
    [TreatNonCallableAsNull] attribute Function?
onmessage;
    void send(DOMString? data);
    void send(Blob data);
    void send(ArrayBuffer data);

    readonly attribute unsigned long
bufferedAmount; //buffered amount

    readonly attribute DOMString extensions;
    readonly attribute DOMString protocol;

    [...]
};
```

Changes to PeerConnection

W3C Editor's Draft

messaging, and channel messaging use the `message` event. [EVENTSOURCE] [WEBSOCKET]

The following interface is defined for this event:

IDL

```
[Constructor(DOMString type, optional MessageEventInit
eventInitDict)]
interface MessageEvent : Event {
  readonly attribute any data;
  readonly attribute DOMString origin;
  readonly attribute DOMString lastEventId;
  readonly attribute WindowProxy? source;
  readonly attribute MessagePort[]? ports;
};

dictionary MessageEventInit : EventInit {
  any data;
  DOMString origin;
  DOMString lastEventId;
  WindowProxy? source;
  MessagePort[]? ports;
};
```

This box is non-normative. Implementation requirements

```
var pc = new PeerConnection('TURNS
example.net', sendSignalingChannel);

//setup what to do when a message arrives
pc.onmessage = function (evt) {
     do something using evt.data
}


// send a string
pc.send("foo");
// the message is delivered; remote end gets
a MessageEvent and handles the incoming
message
pc.send(some blob);
```

Simple example

```
var sendD = New Object();

sendD.type = “A”;
sendD.body = “the data you want to send”;
pc.send(JSON.stringify(sendD));

//setup what to do when a message arrives
pc.onmessage = function (evt) {
     var recD = JSON.parse(evt.data);
     switch (recD.type) {
        case “A”:
           Do something with recD.body
           break;
        Case “B”: ….
}
```

App multiplexing example

```
Interface PeerConnection {
    [...]
 void send(DOMString? data, optional DataOptions
options);
 void send(Blob data, optional DataOptions
options);
 void send(ArrayBuffer data, optional DataOptions
options);
  [...]
};

dictionary DataOptions {
    octet priority = 0;
    boolean reliable = true;

}
```

To be added if need to PeerConnection

5.2 Message channels

IDL

```
[Constructor]
interface MessageChannel {
  readonly attribute MessagePort port1;
  readonly attribute MessagePort port2;
};
```


HTML5 Web Messaging - Mozilla Firefox

Ubuntu Start Page | Holiday Inn Express | eHost | The WebSocket API | HTML5 Web Messaging

dev.w3.org/html5/postmsg/

port is received by the other port, and vice versa.

IDL

```
interface MessagePort : EventTarget {
  void postMessage(any message, optional sequence<Transferable>
transfer);
  void start();
  void close();

  // event handlers
  [TreatNonCallableAsNull] attribute Function? onmessage;
};
MessagePort implements Transferable;
```

```
var channel1 = new MessageChannel();

pc.send("a port for you", [channel1.port2]);

channel1.port1.onmessage = function (event) {
    //do something with event.data
};
channel1.port1.start();
channel1.port1.postMessage("hello");
-----------------------------------------
var port;
pc.onmessage = function (event) {
     port = event.ports[0];
     port.onmessage = function (event) {
         //do something with even.data
     };
     port.start();
     port.postMessage("test");
};
```

Use MessageChannel

```
var channel1 = new MessageChannel();
var channel2 = new MessageChannel();

pc.send("some ports for you", [channel1.port2,
channel2.port2]);

channel1.port1.onmessage =
channel2.port1.onmessage = function (event) {
     console.log("received some data from the
     other peer on the " + (event.target ===
     channel1.port1 ? "first" : "second") + "
     channel: " + event.data);
};

channel1.port1.start();
channel2.port1.start();
channel2.port1.postMessage("hello");
```

Use MessageChannel

```
var ports;
pc.onmessage = function (event) {
     ports = event.ports;
     ports[0].onmessage = ports[1].onmessage =
function (event) {
          console.log("received some data from the
other peer on the " + (event.target ===
ports[0] ? "first" : "second") + " channel: " +
event.data);
     };
     ports[0].start();
     ports[1].start();
     ports[0].postMessage("test");
};
```

Use MessageChannel