ACTION-62, http://www.w3.org/2011/04/webrtc/track/actions/62, “Propose Priority API” was assigned to me at Lyon.

There are several way to do this, e.g.
* Constraints at addStream time
* Fortran style, e.g.
** pc.setPriority(track, priority);
* Fortran style with constraints
** pc.applyConstraints(track, constraints); //constraint for priority included
* Follow what we did for DTMF, allow the creation of a separate object
** pc.createTransportController(track);
** Operate on the “TransporController”.

After some thinking, I think I prefer the last solution (i.e. enable the creation of a separate object to handle transport related things) in combination with re-using constraints in the way Travis proposed in v6 [1].

There are a couple of reasons for this:
* Constraints at addStream time can’t handle tracks added to a stream at a later time, nor does it allow for changes
* I think we will not only want to change priority, but also things like bit-rate, video codec operation (CBR, VBR), DTX on/off, … - this means the Fortral style design would make the PeerConnection API grow a lot
* Fortran style with Constraints is quite OK, but gives no natural place for reporting if during the session a constraint can (temporarily) not be met

So what I propose is basically:

* Add one new method to PeerConnection:
** createTransportHandler - takes a track (must be in a MediaStream in localStreams - otherwise there will be an error) as argument and returns a RTCTransportHandler object

* The RTCTransportHandler (and please propose better names!) uses constraints in the same way is outlined in section 6.2 of [1]
** Initial constraints are priority and bitrate - we can add later as we see need

This design is very similar to the one selected for DTMF, re-uses constraints and how they are proposed to be used with MediaStreamTracks.

[1] http://dvcs.w3.org/hg/dap/raw-file/tip/media-stream-capture/proposals/SettingsAPI_proposal_v6.html

Addition to PeerConnection:

RTCTransportHandler createTransportHandler (MediaStreamTrack track);

New object:

```
interface RTCTransportHandler {
        readonly attribute MediaStreamTrack track;
   any                 getConstraint (DOMString constraintName, optional boolean mandatory =
false);
   void                setConstraint (DOMString constraintName, any constraintValue, optional
boolean mandatory = false);
   TrackTransportConstraints? constraints ();
   void                applyConstraints (MediaTrackConstraints constraints);
   void                prependConstraint (DOMString constraintName, any constraintValue);
   void                appendConstraint (DOMString constraintName, any constraintValue);
   readonly attribute unsigned long?          bitrate;
   readonly attribute unsigned long?          priority;
   readonly attribute RTCTransportState transportState;
                        attribute EventHandler onoverconstrained;
                        attribute EventHandler ontransportstatechange;
};

enum RTCTransportState {
        "inactive",
        "active",
        "removed"
};
```

/* The idea above is that before any media starts flowing the state will be "inactive", once flowing has started the state will be "active"; if the media for some reason can not be streamed it will go back to "inactive", perhaps temporarily. "removed" covers the cases when a track, that is still part of a stream in localStreams, has been removed from the session (i.e. by some SDP manipulation).*/

Constraints:

bitrate unsigned long or MinMaxConstraint
priority PriorityEnum

```
enum RTCPriority {
        "very-low",
        "low",
        "medium", //default
        "high",
```

```
        "very-high"
}
```

Constraints that can be considered for addition in the future:

- Video codec operation (CBR, VBR)
- Use discontinous transmission/comfor noise
- Use of FEC

Example:

```
var audioMinBitrate = 15; // we would not like the audio to go below 15 kbps
var audioMaxBitrate = 30; // we would not like the audio to go above 30 kbps
var videoMinBitrate = 400; // we would not like the video to go below 400 kbps
var videoMaxBitrate = 1000; // we would not like the audio to go above 1000 kbps

/* stream has two tracks, one audio (Id = xyz) and one video (Id = zyx) */
{
        pc.addStream(stream);
        var audioTrsp = pc.createTransportHandler(stream.getTrackById(xyz));
        audioTrsp.setConstraint("bitrate", {min: audioMinBitrate, max: audioMaxBitrate}, false);
        var videoTrsp = pc.createTransportHandler(stream.getTrackById(zyx));
        videoTrsp.setConstraint("bitrate", {min: videoMinBitrate, max: videoMaxBitrate}, false);
}

 /* onnegoitationneeded fires, the app carries out the JSEP / SDP o/a
 stuff, and rtp starts flowing.

 At some later point in time there is a reason to put the priority for the
 video to very-high, and that is the most important thing: */

{
        var videoTrsp = pc.createTransportHandler(stream.getTrackById(zyx));
        videoTrsp.prependConstraint("priority", "very-high", false);
}
```