Dynamic usable constraints

Media Capture Task Force Interim Meeting

Boston, MA

6 February 2013

Travis Leithead and Dan Burnett

Based on Travis' Settings v6 proposal

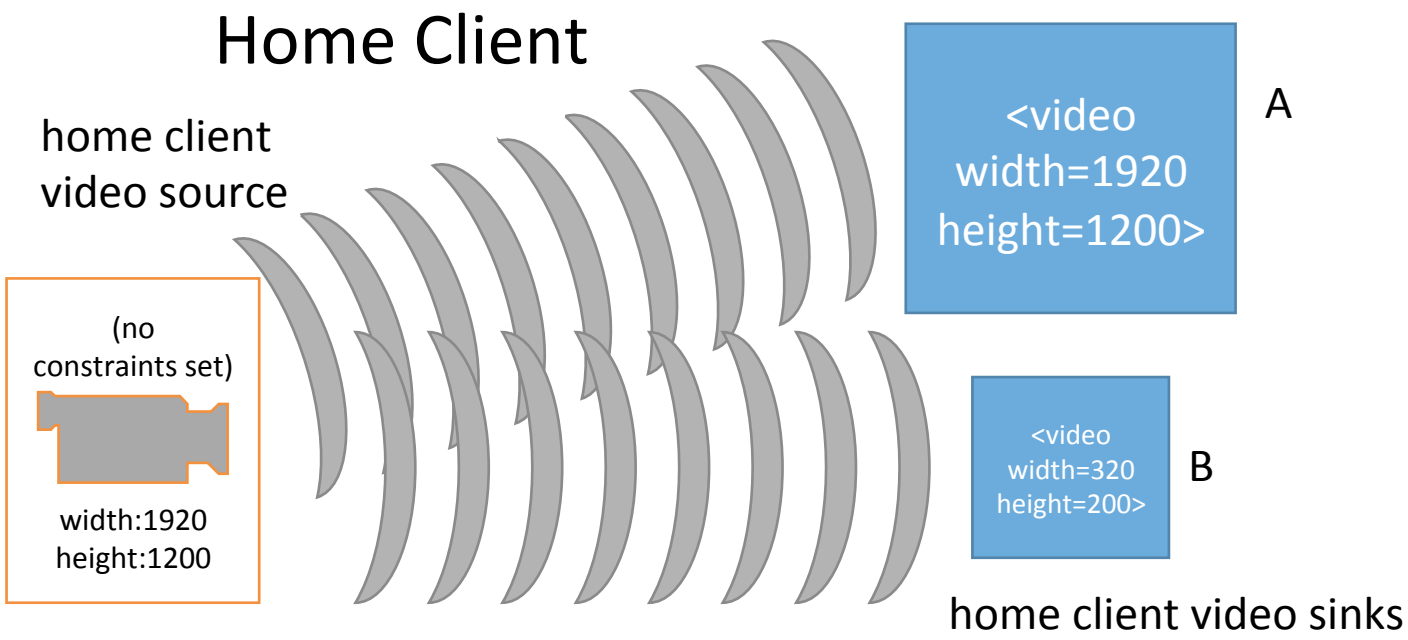
http://lists.w3.org/Archives/Public/public-media-capture/2012Dec/0114.html

Obligatory outline

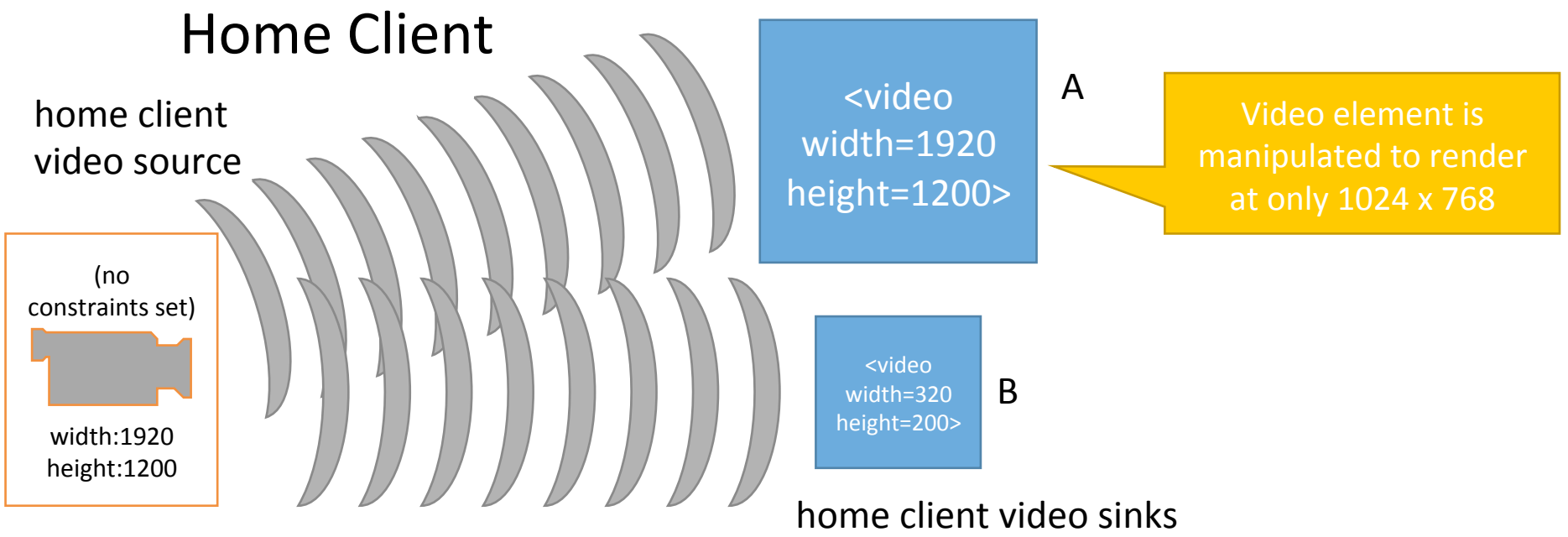
- The Problem
- The Proposal Summary
- The Proposal (Expanded)
- The Other Details
- Candidate Constraints/States/Capabilities

The Problem

- Multiple tracks may use media from the same source
- … but have different requirements (constraints)
- A track may go to a sink with different characteristics than the track
- Source capabilities may change on the fly
- Constraints may need to change on the fly
- Sink configurations may change on the fly



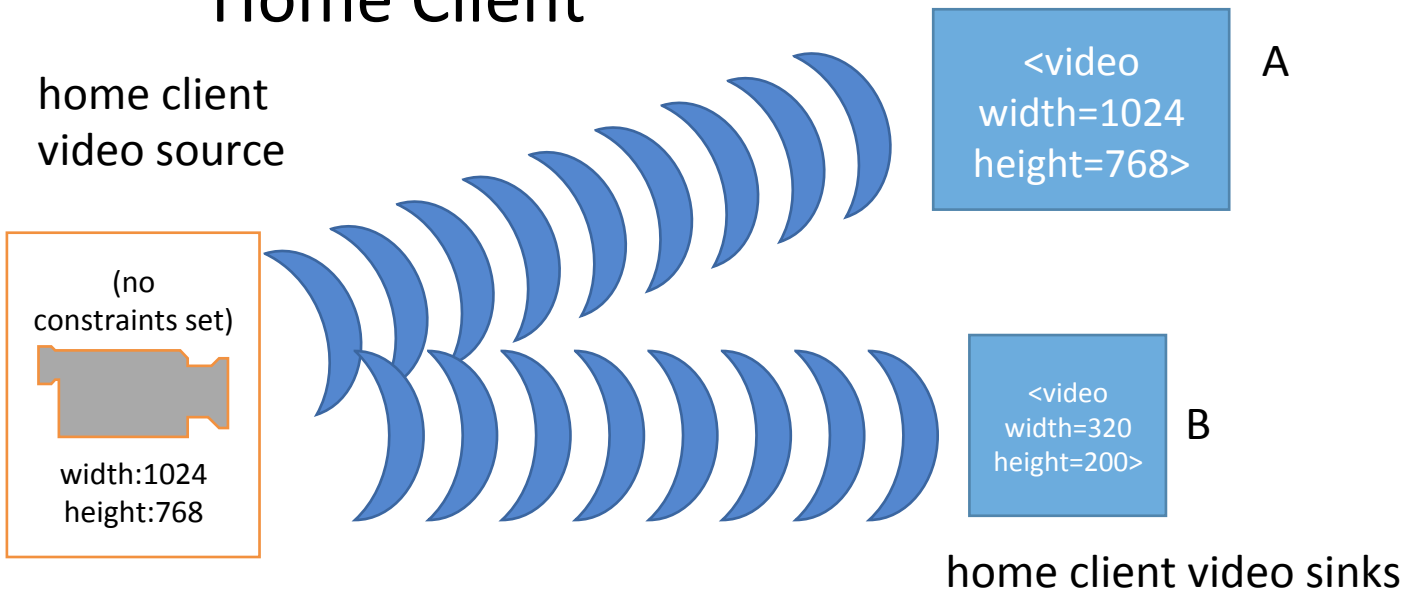
One (local) source: video camera generating 1920x1200 video
Two (local) tracks: from same source, neither constrained
Two (local) sinks: <video> elements with designated (different) dimensions



One (local) source: video camera generating 1920x1200 video
Two (local) tracks: from same source, neither constrained
Two (local) sinks: <video> elements with designated (different) dimensions

Change in one sink does not change or establish constraints

Home Client



One (local) source: video camera generating 1920x1200 video
Two (local) tracks: from same source, neither constrained
Two (local) sinks: <video> elements with designated (different) dimensions

Change in one sink does not change or establish constraints
… but may result in source configuration change (with effect on tracks)

The Proposal Summary

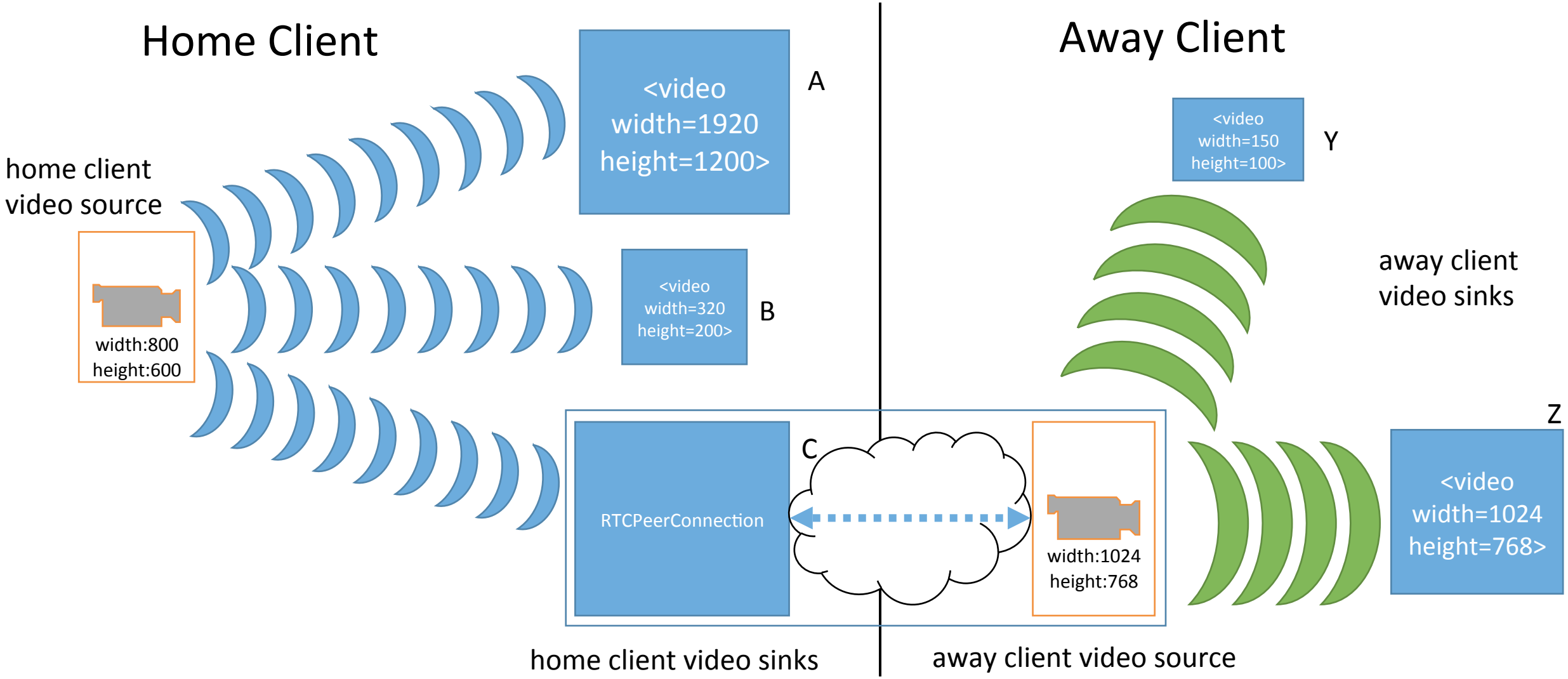
- Separate notions of "source", "track", and "sink"
- Source can change configuration at any time; browser will adjust if possible, event if not
- Constraints can be set only on tracks (not sources)
- … but browser adjusts sources to satisfy all constraints
- Browser notifies (events) when tracks are or become overconstrained
- Capabilities and current state available for every constraint

The Proposal (Expanded)

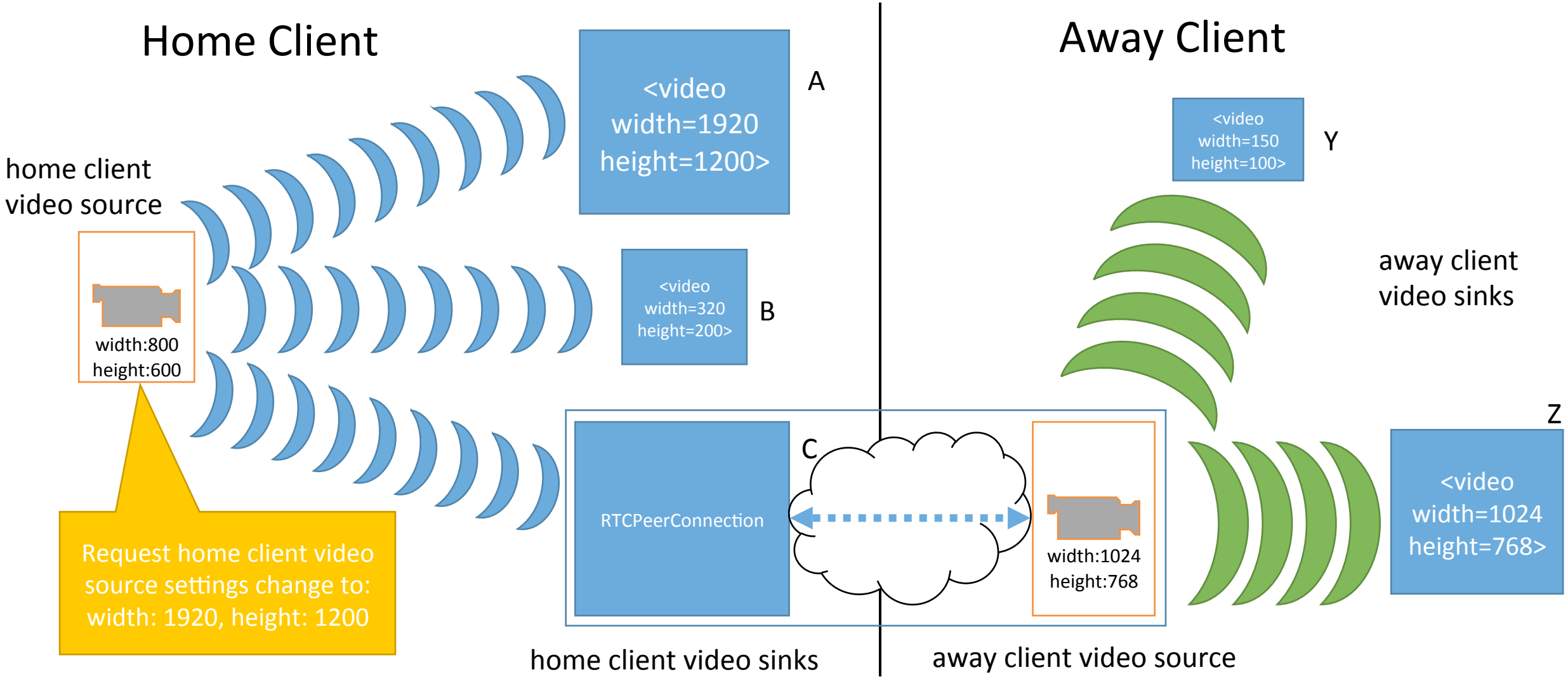
- Separate notions of "source", "track", and "sink"
  - Source
    - Camera, microphone, RTCPeerConnection, file?
    - No direct access or control by app
    - Browser configures
  - Track
    - When local, merely a carrier for constraints, stats, etc.
    - But logically represents specific incarnation of media
  - Sink
    - <video>, RTCPeerConnection, file?
    - No direct access or control via Media Capture/WebRTC APIs

The Proposal (Expanded)

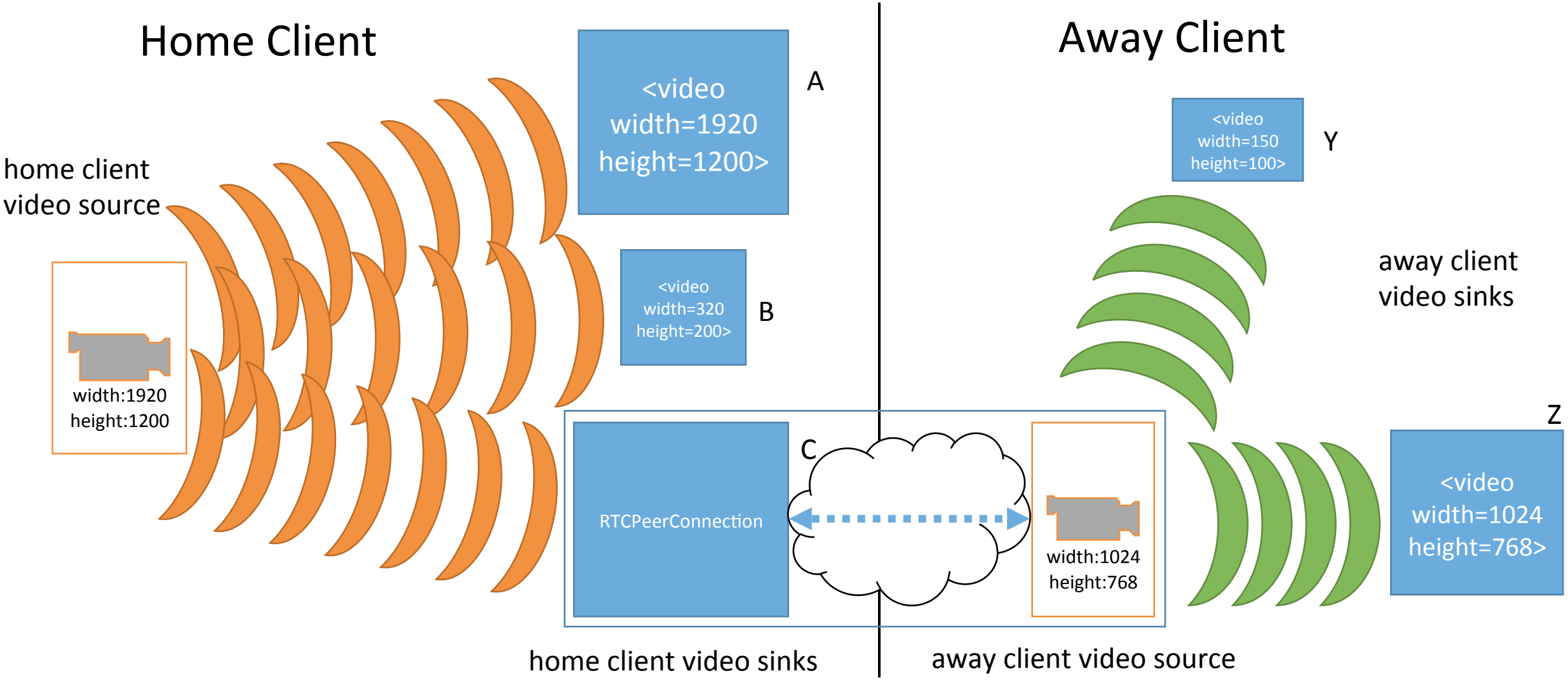
- Source can change configuration at any time; browser will adjust if possible, event if not
  - E.g., camera device may have user-controllable configs
  - Browser might request resolution change for camera to decrease up/ downscaling work



One (local) source: video camera generating 800x600 video
Three (local) sinks: 2 <video> elements and 1 PC
Two remote sinks: 2 <video> elements with 1024x768 PC track as source



Original video source changes resolutions (e.g., through user request)
What happens?



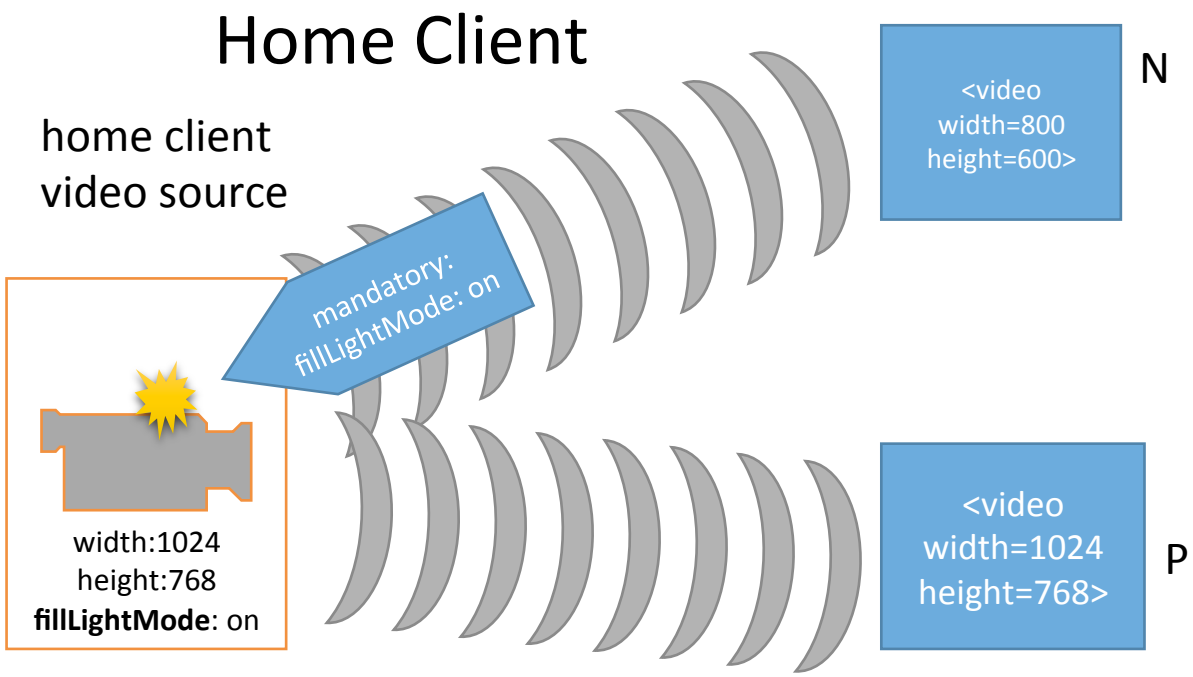
If no constraints on local tracks, track content may change but sinks still the same

The Proposal (Expanded)

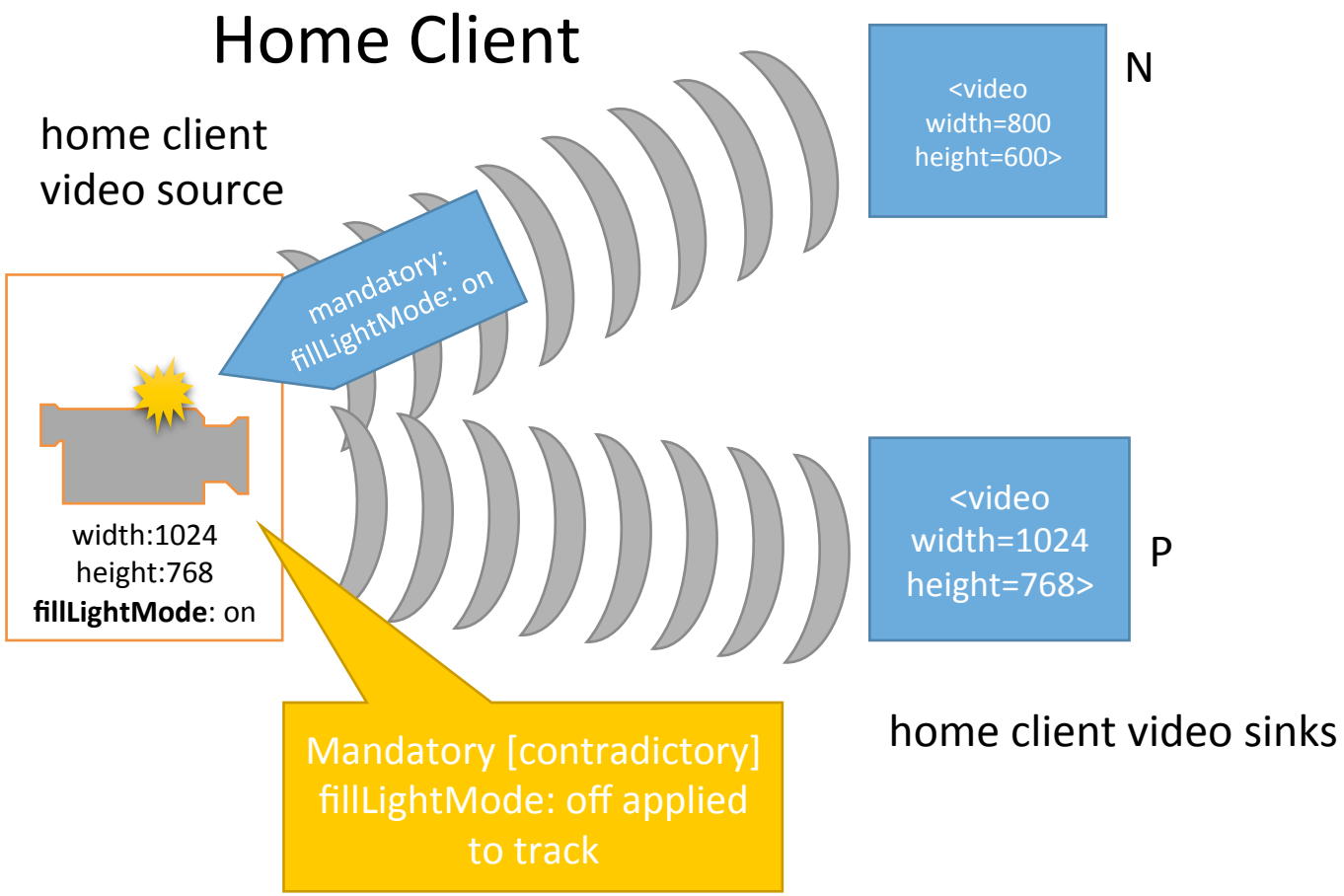
- Constraints can be set only on tracks (not sources)
- … but browser adjusts sources to satisfy all constraints
  - Constraints on all tracks from same source are intersected
  - Browser satisfies this combined constraint set
  - Constraints can be changed on a track (replacement of prior track constraints)

The Proposal (Expanded)

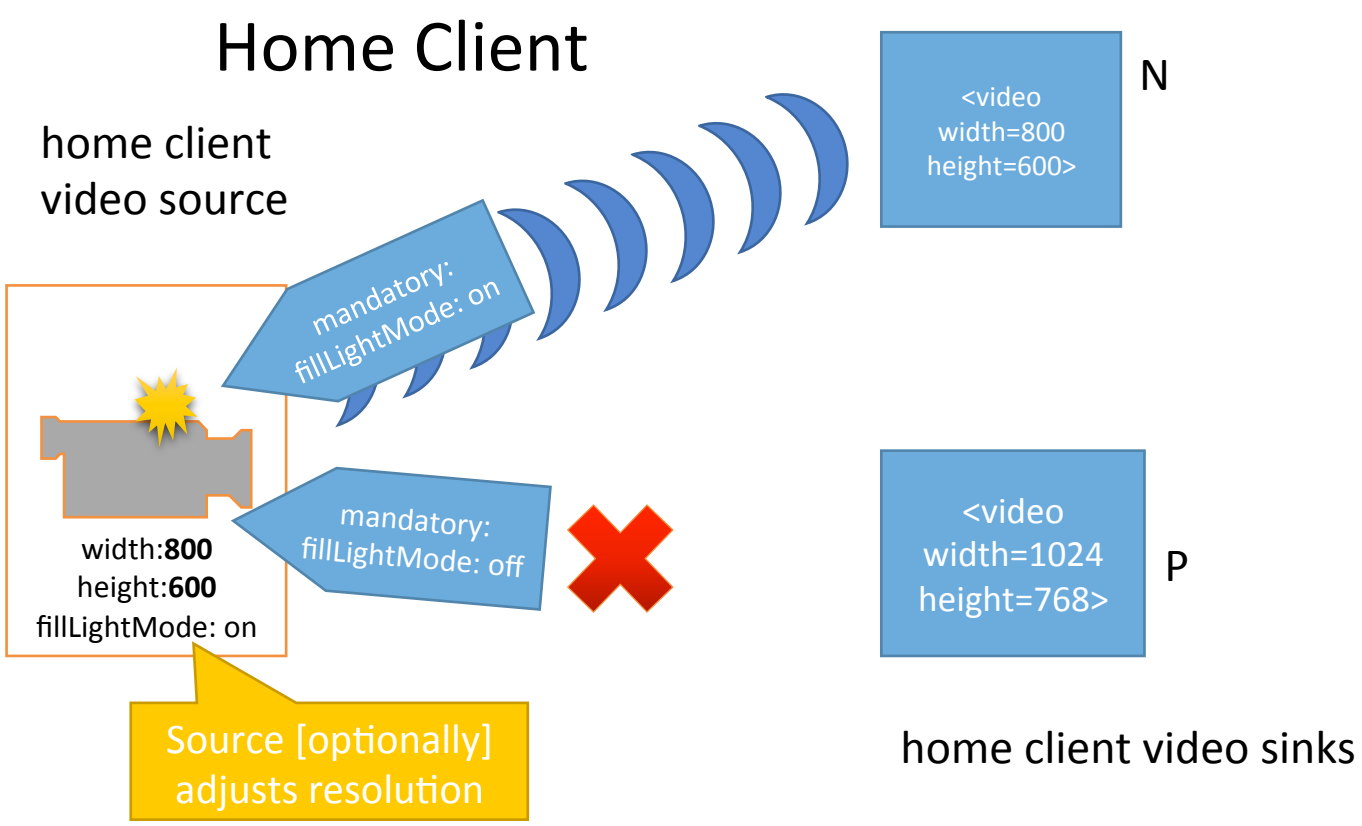
- Browser notifies (events) when tracks are or become overconstrained
  - Track can make itself overconstrained by changing its constraints
  - Can become overconstrained due to source change
    - Device removal
    - Broken Peer Connection
    - Change in stream from Peer Connection



One source: video camera
Two sinks: <video> elements with different resolutions
One mandatory track constraint upon creation: fillLightMode
Source satisfies if it can; otherwise, track (stream) fails to be created



New conflicting constraint from another track
Results in overconstrained error fired at track, and track is muted
NOTE: no required side-effect on the source as a result of this condition!



HOWEVER,

Perhaps the source (browser) notes only one remaining sink
. . . and decides to adjust its resolution to match

NOTE: this is allowed but neither required or to be expected

The Proposal (Expanded)

- Capabilities and current state associated with each constraint
  - Capabilities and states are properties of a source, not a track
  - Constraint "width" (min/max) <->
    Capability "width" (min/max) <->
    Current state "width"
  - "Once a source has been "released" to the application (either via a permissions UI, pre-configured allow-list, or some other release mechanism) the application will be able discover additional source-specific capabilities."

The Other Details

- State/constraint types
  - Min/max, primary types (string, int, float, Boolean)
- Source IDs
  - Capability, state, and constraint
- Separate audio and video track subtypes
- Explicit track creation (using subtypes) before getUserMedia() call
  - Uses Martin's recommended gUM model

Proposed (mod of Settings v6)

- Tracks/sources can also be
  - Readonly and/or remote
  - Both available as states

About the proposal

- Good?
- Bad?

- Plan is to start applying to Media Capture and Streams draft

Candidate constraints/states/capabilities

- Video:
  - sourceId
  - width
  - height
  - aspectRatio
  - framerate
  - facingMode
  - zoom
  - focusMode
  - fillLightMode
  - whiteBalanceMode
  - brightness
  - contrast
  - saturation
  - sharpness
  - photoWidth
  - photoHeight
  - exposureMode
  - isoMode
  - orientation
  - apertureSize
  - shutterSpeed
  - denoise
  - effects
  - faceDetection
  - antiShake
  - geoTagging
  - highDynamicRange
  - skinToneEnhancement
  - shutterSound
  - redEyeReduction
  - sceneMode
  - antiFlicker
  - zeroShutterLag
  - rotation
  - mirror
  - bitRate

Candidate constraints/states/capabilities

- Audio:
  - sourceId
  - volume
  - gain