



dash.js Face-to-Face Meeting 2022

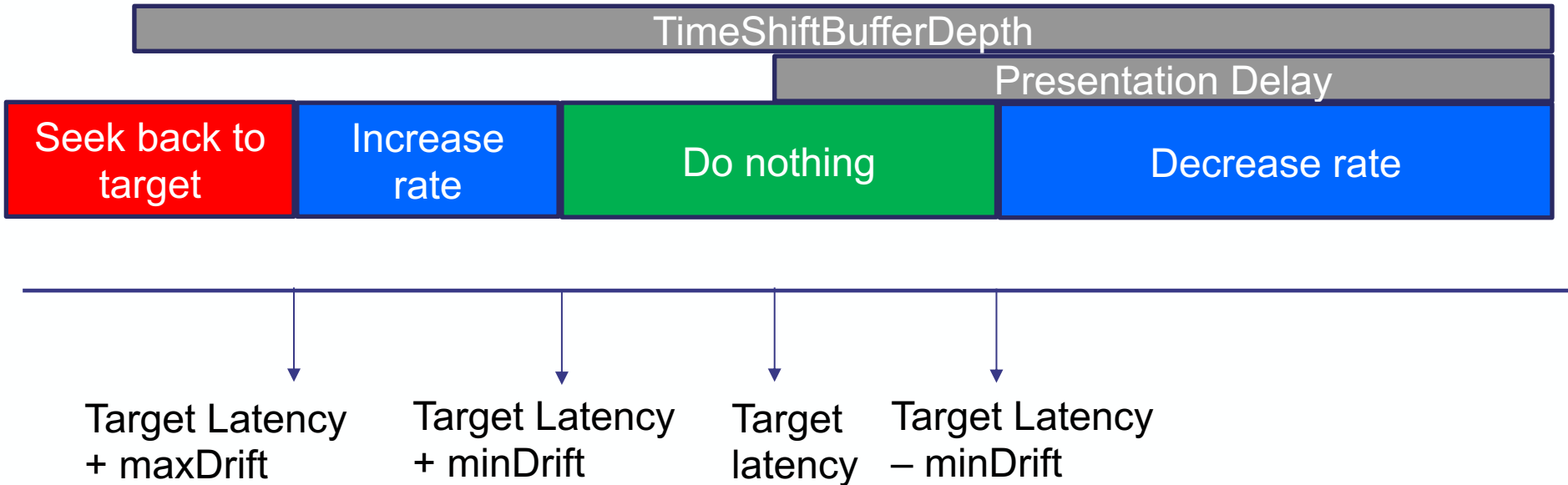
Catchup Logic


Daniel Silhavy

Pull Request

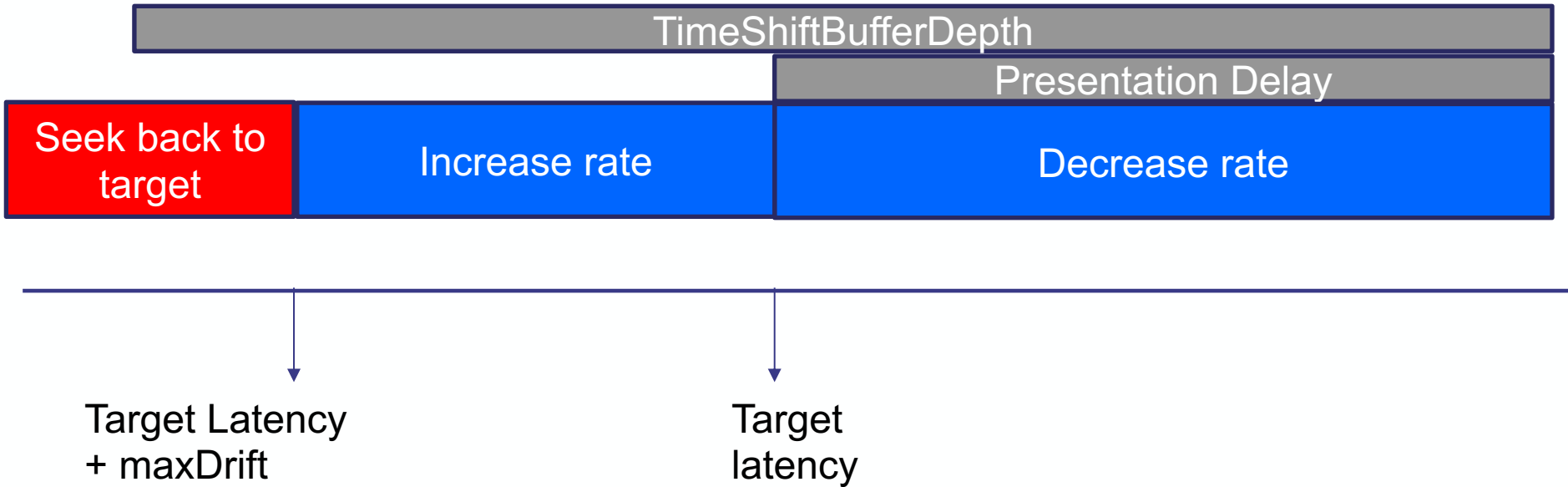
- Changes discussed in this slide deck are part of the following PRs.
 - <https://github.com/Dash-Industry-Forum/dash.js/pull/3831>
 - <https://github.com/Dash-Industry-Forum/dash.js/pull/3895>

dash.js catchup behavior before 4.4.0



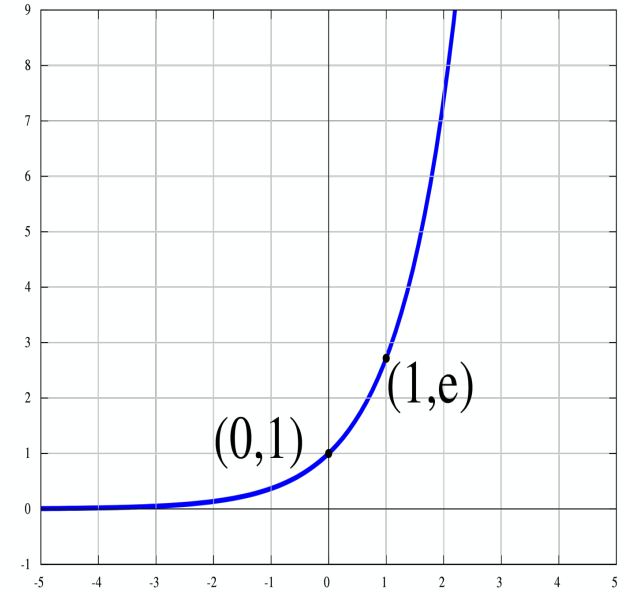
 We stopped catchup mode before target latency was reached

dash.js catchup in 4.4.1



Calculating the new playback rate

- `const s = (cpr * 2) / (1 + Math.pow(Math.E, -d));`
- `newRate = (1 - cpr) + s;`
- `cpr = catchup playback rate`
- `d = multiple of the delta latency`
- `Math.e = base of natural logarithm`

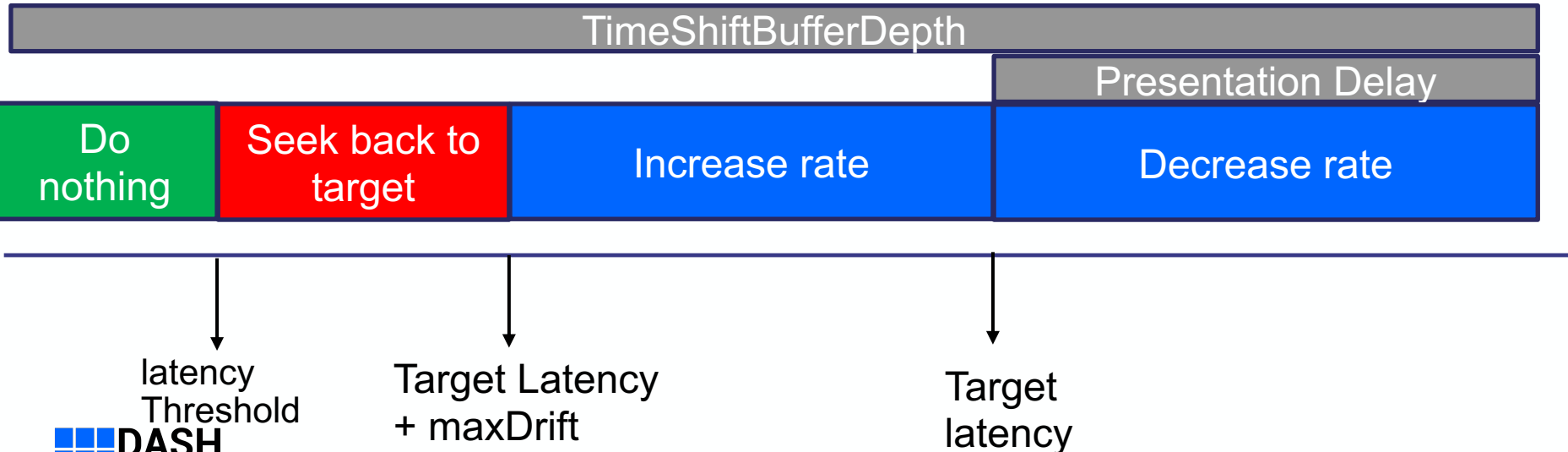


$$y = e^x$$

```
cpr = 0.5
delta latency = current latency - target latency = 5 - 2 = 3
d = delta latency * 5 = 3 * 5 = 15
s = (cpr * 2) / (1 + Math.pow(Math.E, -d)) = (0.5 * 2) / (1 + 3.0590232050182605e-7) = 0.999999694097773
new rate = (1 - cpr) + s = (1 - 0.5) + 0.999999694097773 = 1.499999694097773
```

Seek Problem

- How to deal with seeks triggered by the user?
- Currently we define a “latencyThreshold”: Use this parameter to set the maximum threshold for which live catch up is applied. For instance, if this value is set to 8 seconds, then live catchup is only applied if the current live latency is equal or below 8 seconds.



Seek Problem - Options

- Option 1:
 - Leave as it is
- Option 2:
 - Seek triggered by the user always deactivates catchup mechanism. Re-activating the catchup mode is up to the application. For instance, in the reference UI we wait for the user clicking on “Live” to re-activate
- Option 3:
 - We set the target live delay to the seek target. Catchup mechanism is applied at this “new” live delay. We need to maintain the original live delay if the user wants to seek back to the initial live edge.

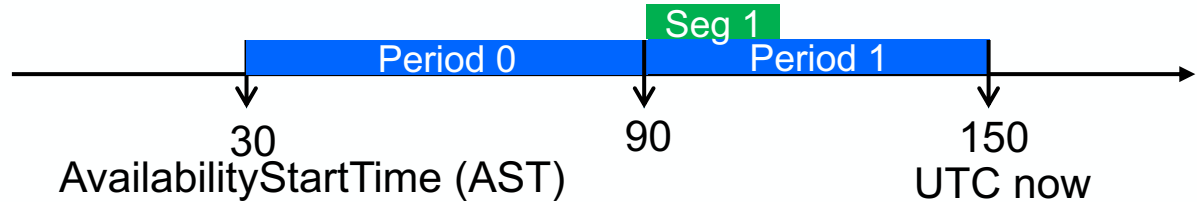
Additional Refactoring as part of the PR

- New *ServiceDescriptionController* class to handle all tasks related to the <ServiceDescription> element
 - Added support for <OperatingBandwidth> element: min,max,target bitrate
 - Ensure easy implementation of profile specific logic
- New *CatchupController* class to handle all tasks related to applying the catchup mechanism
- Reduced size of *PlaybackController* to ensure better maintainability
- Use *MediaPlayerModel* as a proxy between classes and *Settings* if additional logic needs to be applied and the settings can not be used 1:1
- New class *CustomParametersModel* to save all callback functions and additional parameters defined by the app
- Remove *enableLowLatency* from *Settings*. Switch to low latency mode internally based on MPD parameters *availabilityTimeComplete*
- Calculate the live latency at a central place *PlaybackController.computeAndSetLiveDelay*

ProducerReferenceTime

- BBC implemented support for <ProducerReferenceTime> according to TS 103 285 Clause 10.20.4.

- Example:



General

- AST = 30
- Latency@target = 10
- timescale = 1

Period 0

- @start = 0

Period 1

- @start = 60
- EPT first segment = 2
- @presentationTimeOffset = 2
- PRFT
 - WCT = 95
 - PT = 5 (relative to P1@start)
 - type = encoder

Calculation

ExpectedPresentationTime(WCT)
 = WCT – AST – Period@start + PTO

ExpectedPresentationTime(95) =
 95 – 30 – 60 + 2 = 7

Offset = PRFT@PT – 7 = -2

Latency@target =
 Latency@target – Offset = 12