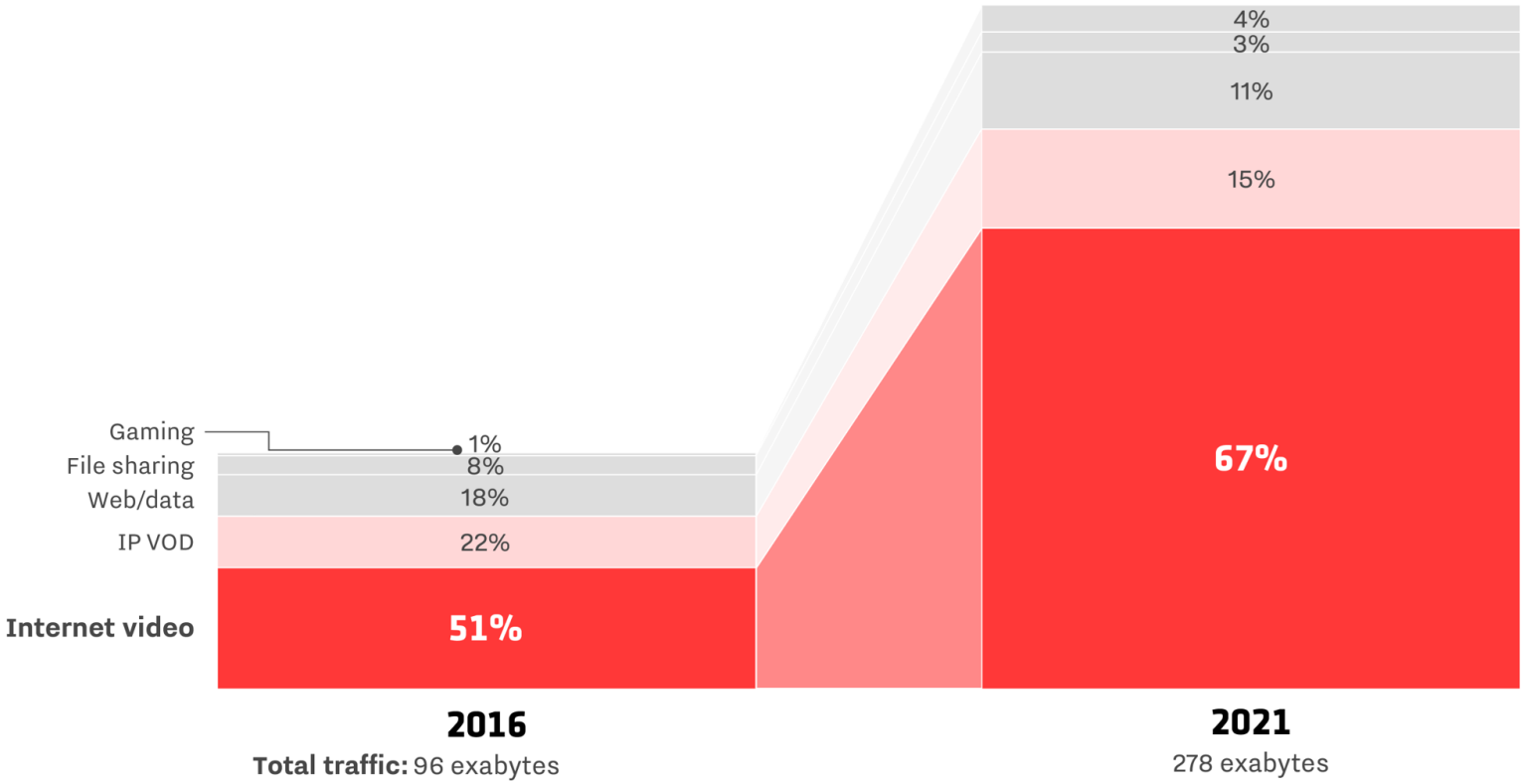
End-to-End Transport for Video QoE Fairness

Vikram Nathan, Vibhaalakshmi Sivaraman, Ravichandra Addanki,
Mehrdad Khani, Prateesh Goyal, Mohammad Alizadeh

November 28, 2020

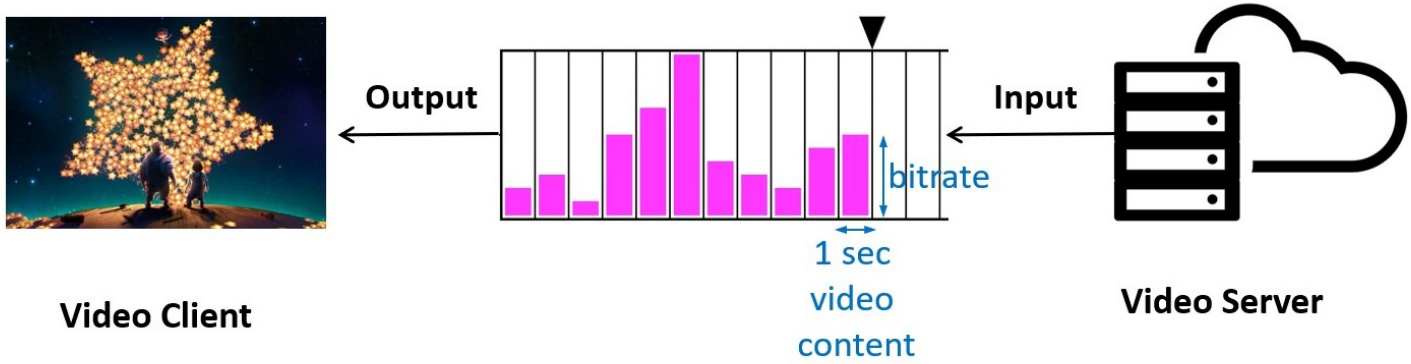
Overview

Global IP Traffic



Streaming video are more than 70% of global Internet traffic.

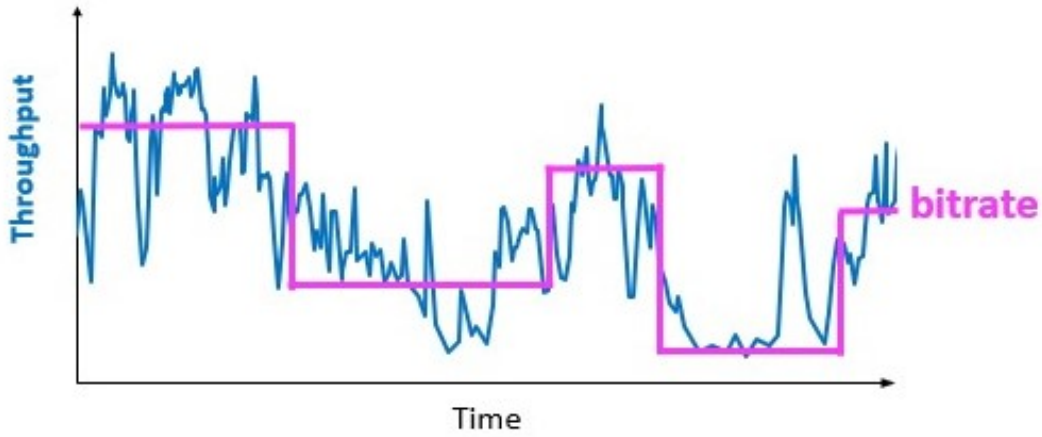
HTTP-based Video Streaming



In the past most video streaming technologies utilized streaming protocols such as RTP/RTSP, today are almost exclusively based on HTTP.

Adaptive Bitrate Streaming

1. MPEG-DASH
2. Adobe HTTP Dynamic Streaming
3. Apple HTTP Live Streaming
4. Microsoft Smooth Streaming



Videos value bandwidth differently!

94

58

Videos value bandwidth differently!

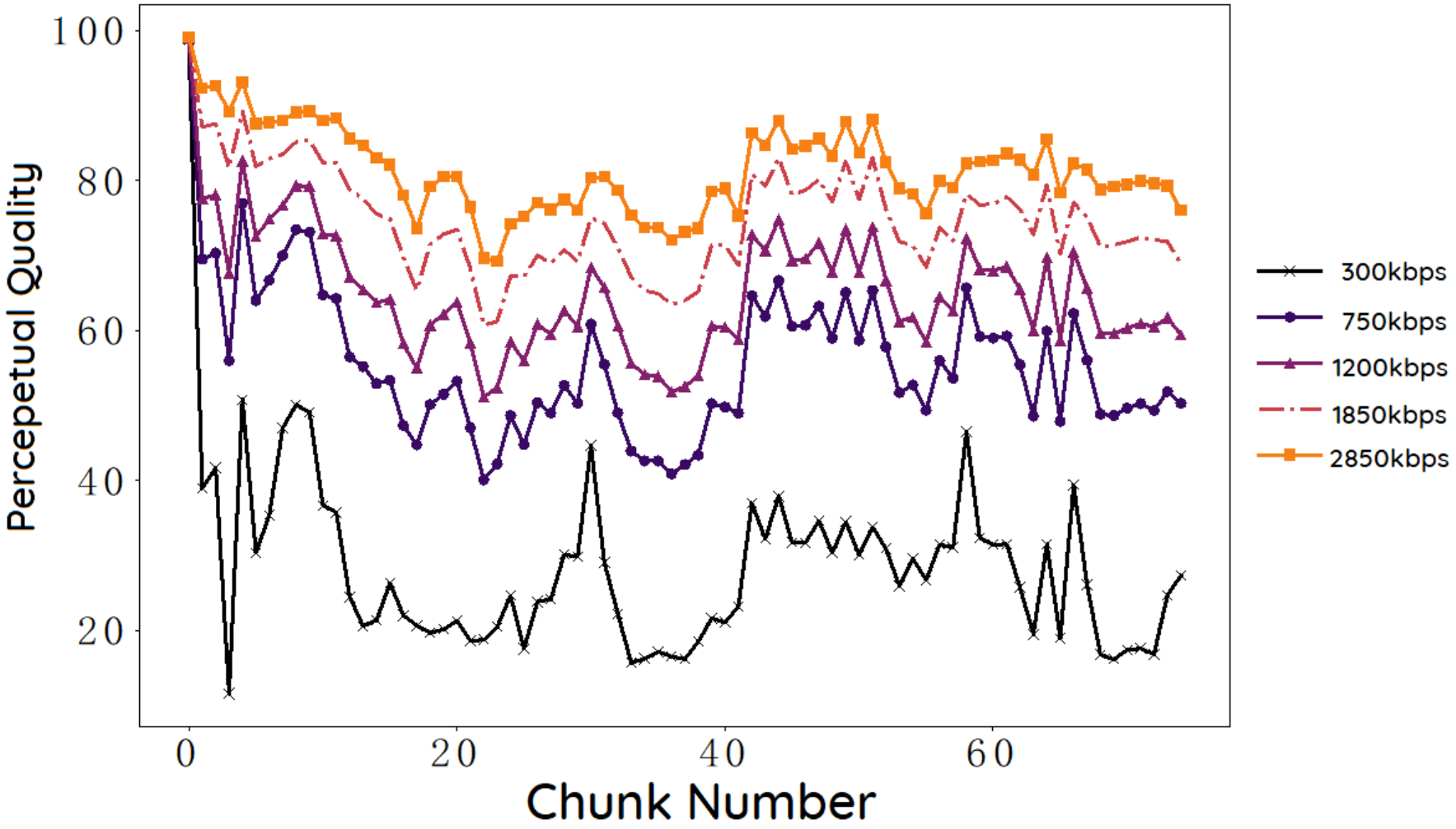
82

6

Perceptual Quality Function

- VMAF(Video Multimethod Assessment Fusion)
- PSNR(Peak Signal To Noise Ratio)
- SSIM(Structural Similarity Index)

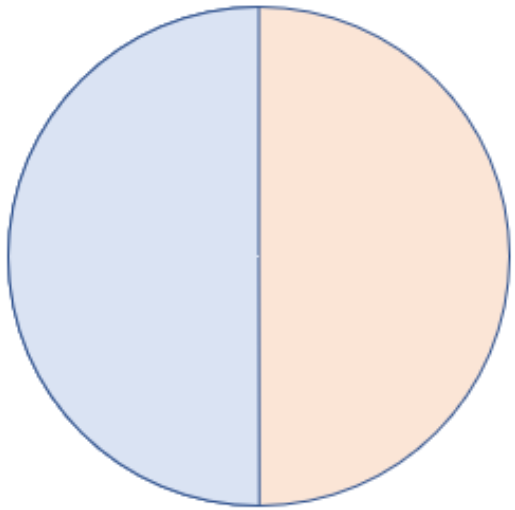
Perceptual Quality Function



Session State

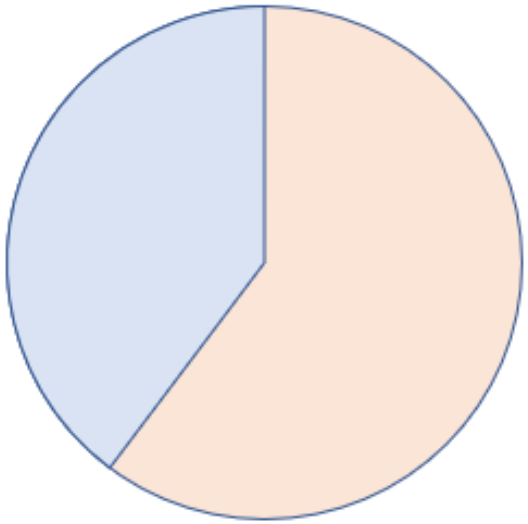
1. Buffer Size
2. Client require
3. User platform

Multiple Video Streams Competing



Multiple Video Streams Competing

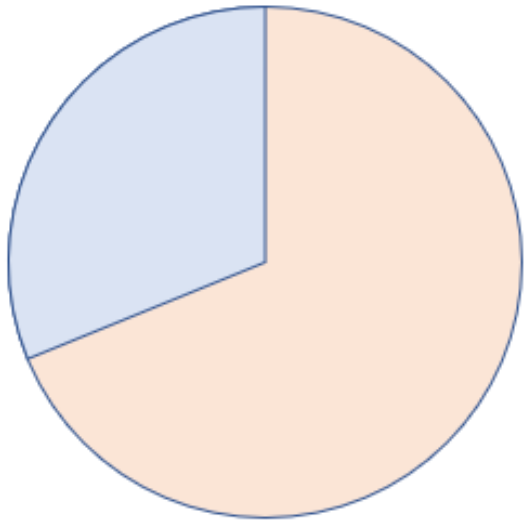
Doesn't need hi-res



Needs hi-res

Multiple Video Streams Competing

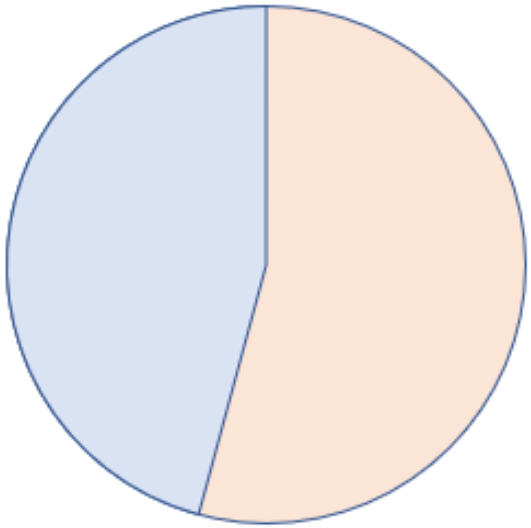
Doesn't need hi-res
Mobile Phone



Needs hi-res
HD-TV

Multiple Video Streams Competing

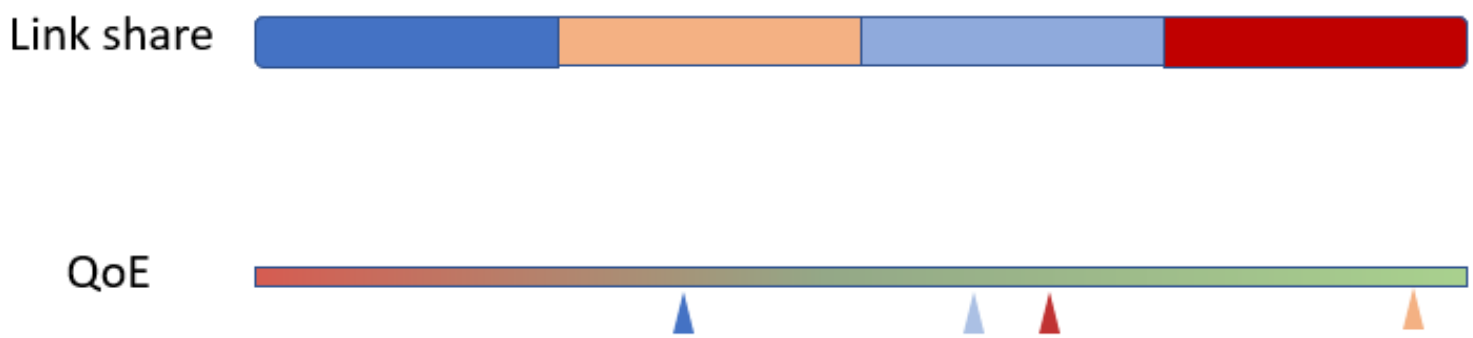
Doesn't need hi-res
Mobile Phone
Low buffer
(risk of stalling)



Needs hi-res
HD-TV
Full buffer

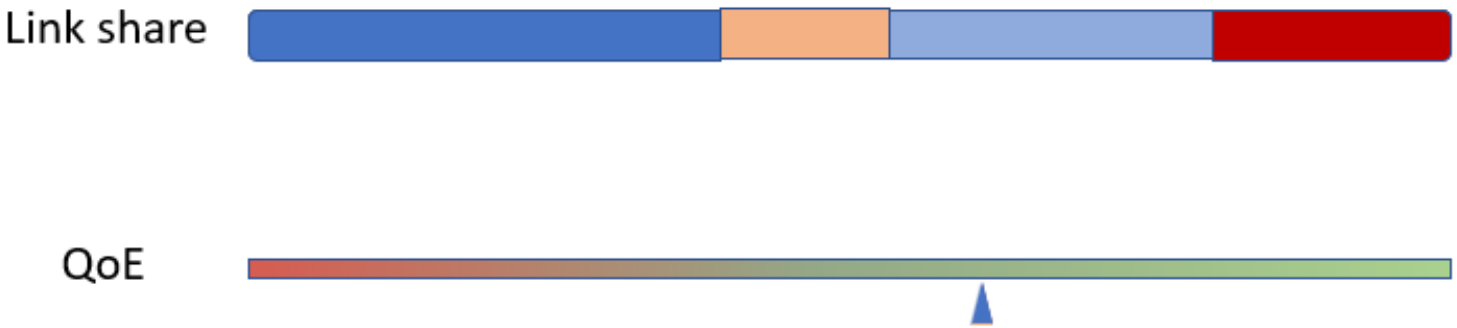
QoE Fairness

- What if video providers could allocate bandwidth for their videos based quality of experience ?
- Objective: max-min QoE fairness



QoE Fairness

- What if video providers could allocate bandwidth for their videos based quality of experience ?
- Objective: max-min QoE fairness



Goals

Achieve max-min QoE fairness on their videos.

- Requires changes only to client and server endpoints
- Clients do not communicate with each other or know of others' existence.
- Plays fairly with non-Minerva traffic on average.

QoE Function

$$\mathrm{QoE}(c_k, R_k, c_{k-1}) = P(c_k) - \beta R_k - \gamma ||P(c_k) - P(c_{k-1})||$$

- c_k: Bitrate of kth chunk.
- R_k: Rebuffering time.
- $P(e)$: quality gained from watching a chunk at bitrate e.

Utility Function

- QoE Function is difficult to optimize directly.
- QoE is not directly related to download rate.

Utility Functiom

- Utility captures how "good" the client's viewing experience is expected to be over the course of the video.
  - Low Utility $\rightarrow$ Need high bandwidth.
  - Measure the QoE for each video chunk.
- Client aware utility

$$U(r) = \frac{\varphi_1(\text{Past QoE}) + \varphi_2(\text{QoE from current chunk}) + V_h(r, b, c_i)}{1 + \varphi_1 + \varphi_2}$$

Measure Utility

$$U(r) = \frac{\varphi_1(\text{Past QoE}) + \varphi_2(\text{QoE from current chunk}) + V_h(r, b, c_i)}{1 + \varphi_1 + \varphi_2}$$

- Past QoE: Using saved QoE value.

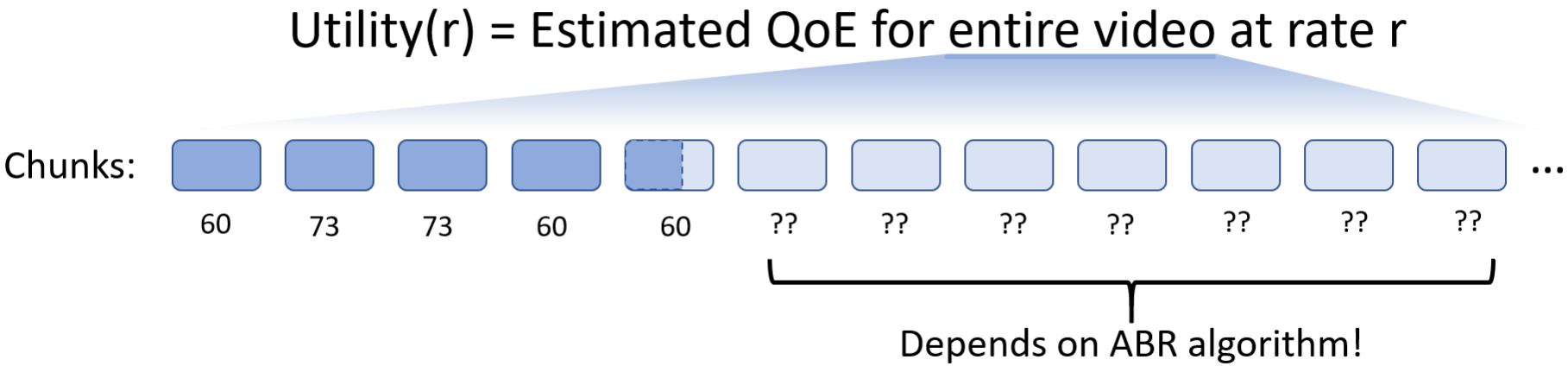
Measure Utility

$$U(r) = \frac{\varphi_1(\text{Past QoE}) + \varphi_2(\text{QoE from current chunk}) + V_h(r, b, c_i)}{1 + \varphi_1 + \varphi_2}$$

- The QoE of the current chunk is estimated by using the current rate r to determine if the video stream will rebuffer. Suppose that a client with buffer level b is downloading a chunk c_i and has c bytes left to download, Minerva computes the expected rebuffering time $R = [c/r - b]_+$ and estimates the QoE of the current chunk as $QoE(c_i, R, c_{i-1})$,

$V_h(r, b, c_i)$: Future QoE

Computing the expected per-chunk QoE over the next h chunks.

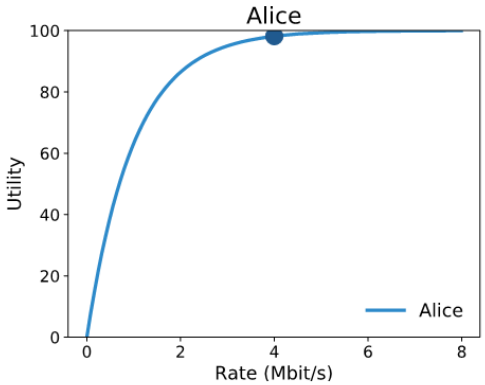


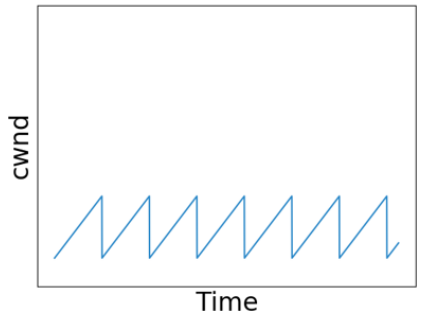
$V_h(r, b, c_i)$: Future QoE

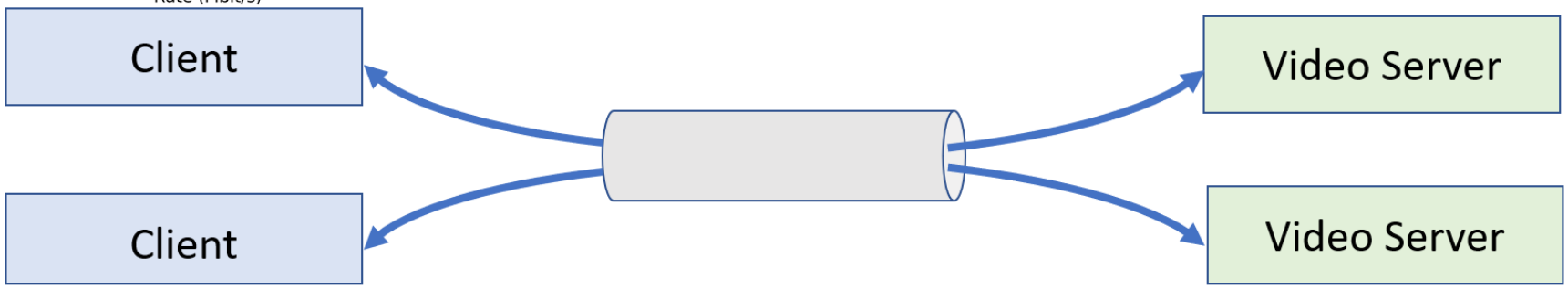
Computing the expected per-chunk QoE over the next h chunks.

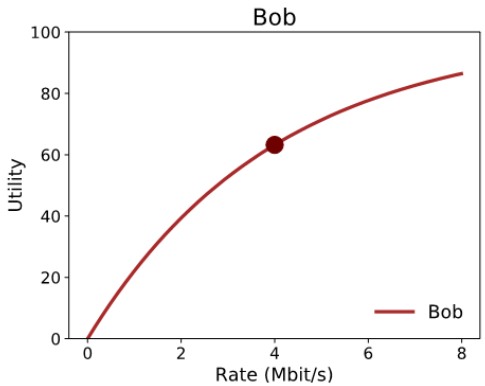
```
r ← Measured download rate
S ← client state
do h times:
  e = ABR(S)
  S, rebuf ← client state and
       rebuffer time after chunk
       e is downloaded at rate r
```

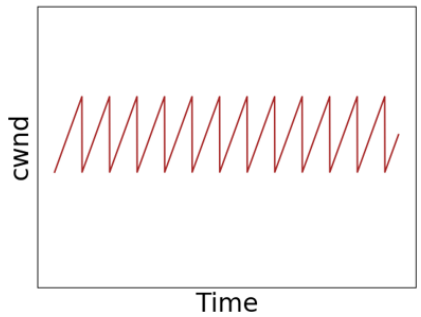
Share Bottleneck



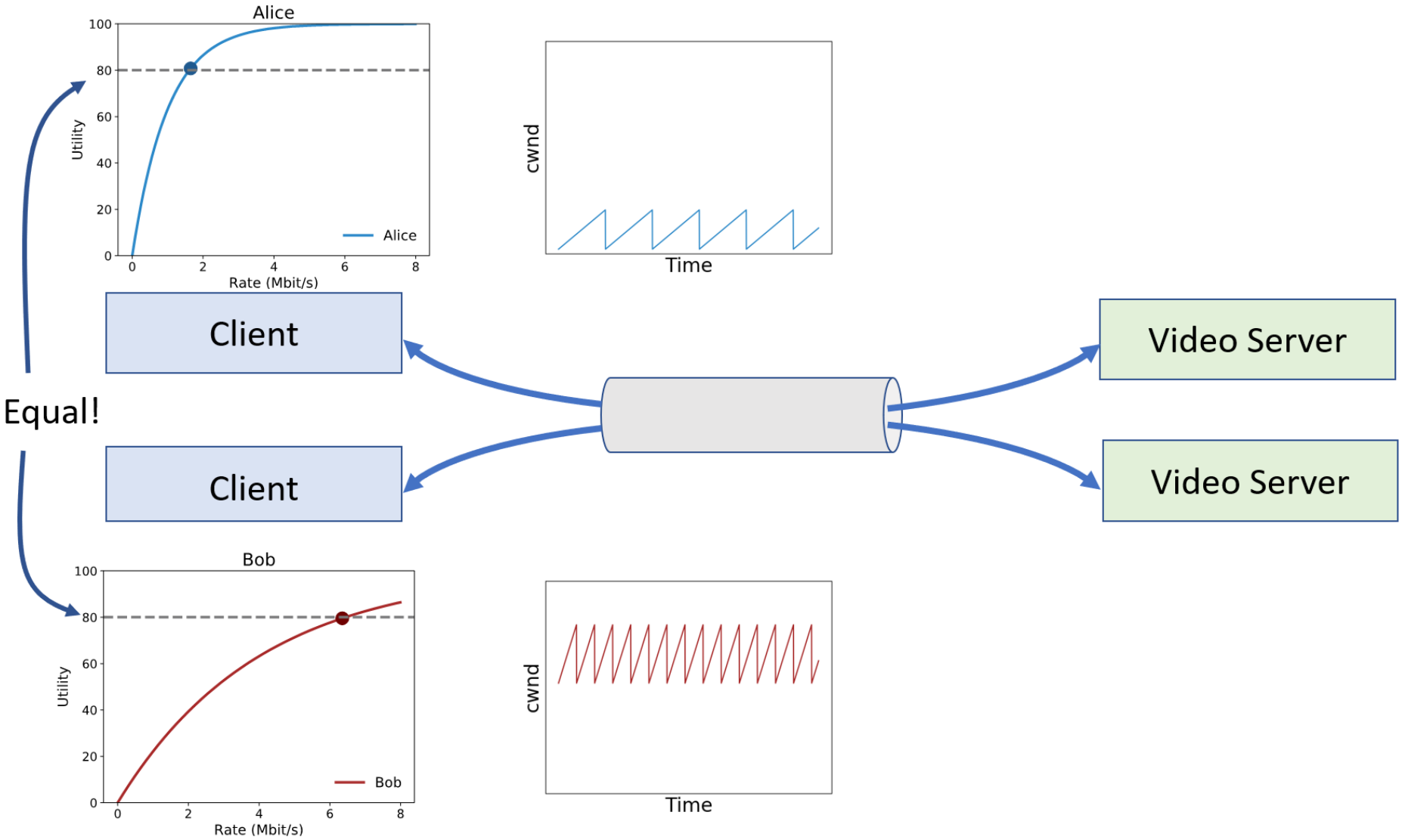








Share Bottleneck



Rate Update

Update the Utility function every T seconds.
Each client must find a weight w_i , such that the set of all client weights determines a relative bandwidth allocation.

Rate Update

Each client must find a weight w_i , such that the set of all client weights determines a relative bandwidth allocation.

$$W_i(r_i) = \frac{r_i}{U_i(r_i)}$$

All clients using this update rule will converge to max-min QoE fairness in a logarithmic number of steps.

Rate Update

If all clients at the optimal rates, $U_i(r_i)$ are equal for all clients,

$$\frac{r_i'}{r_j'} = \frac{r_i^* / U_i(r_i^*)}{r_j^* / U_j(r_j^*)} = \frac{r_i^*}{r_j^*}$$

Therefore, the rate ratios remain identical.We assumes the link capacity stays constant in the short term, so identical rate ratios will result in identical rates.

Rate Update

Each iteration moves the rates closer towards their optimal values.
Consider two clients

$$\rho = \frac{r_1}{r_2}, r_1 < r_1^* \rightarrow r_2 > r_2^*, U_1(r_1) < U_1(r_1^*) = U_2(r_2^*) < U_2(r_2)$$

the ratio in the next iteration is

$$\rho' = \rho \cdot \frac{U_2(r_2)}{U_1(r_1)} > \rho$$

Normalization

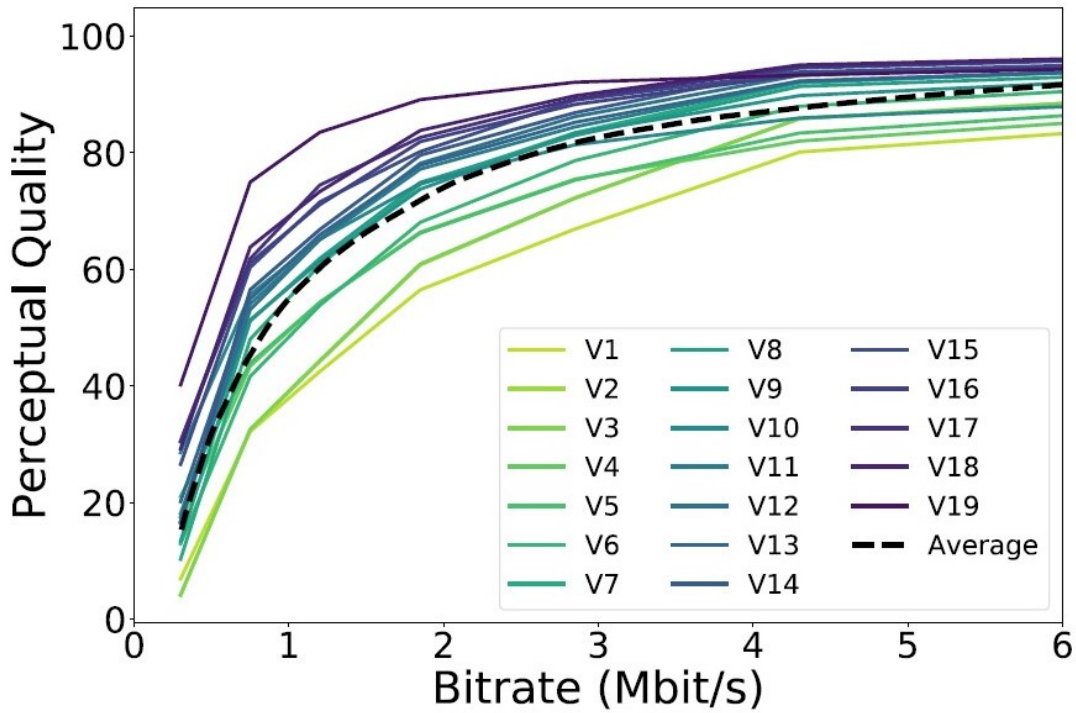
Plays fairly with non-Minerva traffic on average.
f is monotonically increasing

$$\frac{1}{N}\sum_i \frac{r_i}{f(U_i(r_i))} = \alpha \rightarrow f(u) = \frac{1}{\alpha N}\sum_i U_i^{-1}(u)$$

Normalization

$$W_i(r_i) = \frac{r_i}{\tilde{f}(U_i(r_i))} \text{ where } \tilde{f}(u) = \sum_{v \in V} p_v U_v^{-1}(u)$$

- U_v is the PQ function, the dotted line is the $\tilde{f}$



Normalization

$$W_i(r_i) = \frac{r_i}{\tilde{f}(U_i(r_i))} \text{ where } \tilde{f}(u) = \sum_{v \in V} p_v U_v^{-1}(u)$$

- If each video is drawn randomly from a popularity distribution $\{p_i\}$ then:

$$\mathrm{E}[f(u)] = \sum_{v \in V} p_v U_v^{-1}(u) = \tilde{f}(u)$$

Disadvantage

- Security: Client can easily cheat server.
- A lot of calculation.
- Cannot handle complex bottleneck.