2018 EDITION

LEARN TO PRODUCE VIDEO

WITH  FFMPEG

IN 30 MINUTES OR LESS

BY JAN OZER

Hey There.

Jan Ozer here, thanks for downloading Chapter 4 from the ***2018 Edition of Learn to Produce Video With FFmpeg in 30 Minutes or Less.*** This chapter covers bitrate control, and details how to encode to CBR, VBR, CRF and capped CRF with FFmpeg.

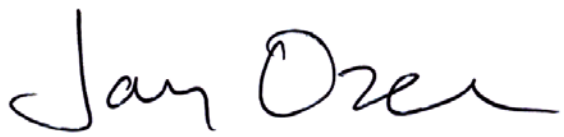
Here's a link to the batch files associated with the chapter, which includes the media files (bit.ly/ch4dolo). It's about 500 MB in size which includes the file you'll need for testing and to duplicate the results from the chapter.

Like all chapters, it's short, focused, and to the point. It includes some streaming-related instruction so you understand the theory underlying the configurations, and you get the batch files and test files you need to verify the result and quickly apply them to your own projects.

I've also included the Table of Contents behind the chapter so you can see what's covered in the book.

You know the drill; I'm hoping this chapter is so useful that you decide to buy the entire book. If so, you can find it at www.streaminglearningcenter.com/learnffmpeg. The book is available now in PDF format ($29.95) and should be available in Paperback on Amazon by September 10 or so ($34.95). You'll get the batch files for all chapters right after registering. If there's an issue, contact me at janozer@gmail.com

Thanks again, and I hope you find the chapter useful.



Chapter 4: Bitrate Control

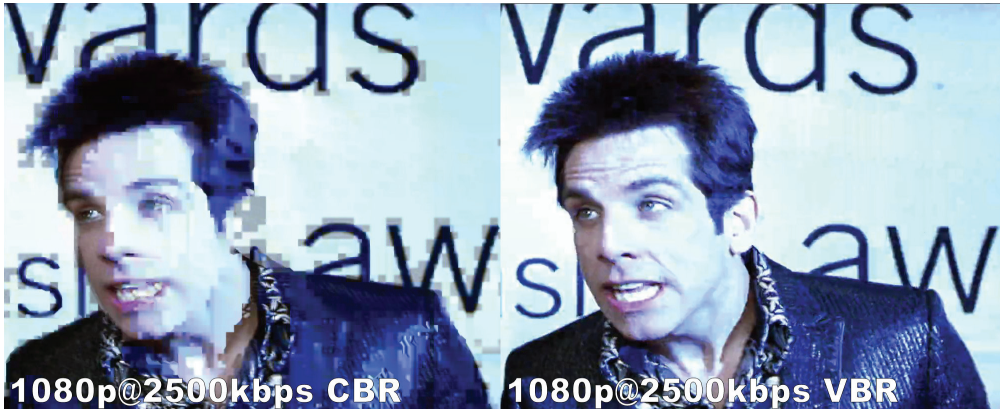


Figure 4-1. Sometimes CBR-encoded video exhibits transient quality glitches like this one in the movie Zoolander.

Whenever you encode a file, you must choose both the bitrate and the bitrate control technique, or how the video data rate is allocated within the file. For most producers, this means a choice between constant bitrate (CBR) or variable bitrate (VBR) encoding. While these choices have been available since the dawn of H.264 (and MPEG-2 and before it, for that matter) there still is no consensus as to which is best to use.

In general, the CBR-versus-VBR decision involves a debate between quality and deliverability. It's generally accepted that VBR produces better quality than CBR, although it probably doesn't make as big a difference as you might think. Despite the quality advantage, many producers use CBR over concerns that variances in the VBR bitrate will make their files harder to deliver, particularly over constrained bitrate connections like 3G and 4G. As you'll learn later in this chapter, these concerns are appropriate. Otherwise, in this chapter, you will learn:

- how VBR and CBR work
- differences in overall frame quality
- how both techniques affect deliverability
- what the Video Buffering Verifier (VBV) is and how it affects bitrate control and quality
- best practices for encoding with CBR and VBR
- how to encode CBR and VBR files in FFmpeg
- what CRF encoding and capped CRF encoding are and how to use them.

CBR and VBR Defined

Most encoding tools provide the bitrate control options shown in Figure 4-2, CBR or VBR. As you'll learn in this chapter, you can encode using both techniques with FFmpeg as well.

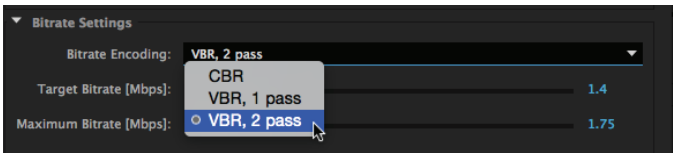


Figure 4-2. What will it be today, CBR or VBR?

Let's use the file shown in Figure 4-3 to illustrate the difference between CBR and VBR. As you can see, the file has fives scenes, as follows:

- ***Low motion.*** Talking head.
- ***Moderate motion.*** Woman cooking pita bread on an outdoor oven.
- ***Low motion.*** An integrated-circuit chip-cutting machine in operation.
- ***Moderate motion.*** A musician playing the violin.
- ***High motion.*** Walking holding the camcorder to my chest and panning side to side.

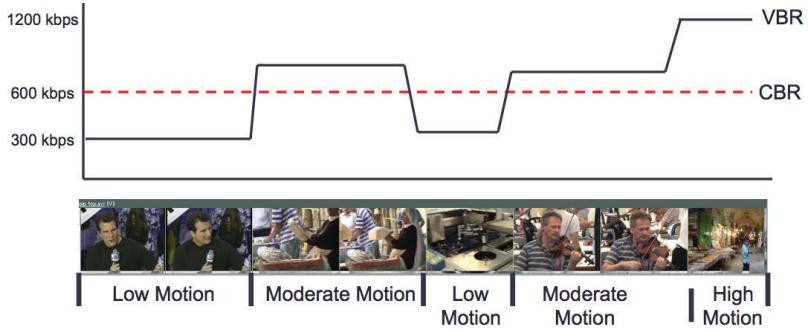


Figure 4-3. CBR applies the same data rate to the entire file, while VBR varies the data rate to match scene complexity.

CBR, the dotted red line, ignores the variances, and produces a constant 600 kbps throughout. In contrast, VBR varies the bitrate according to the complexity of the video—lower in the easy-to-compress talking-head sequence, and higher in the high-motion sequence at the end.

Note that you can produce CBR using either a single pass or multiple passes. With multiple passes, the encoder assesses complexity during the first pass, and then encodes and allocates bits during the second. Obviously, with live encodes, CBR is produced in a single pass.

Almost all VBR is produced using two passes, again, one for analysis, one for encoding. In addition, most VBR is "constrained," which means you assign a maximum data rate that the encoder won't exceed. So, 200% constrained VBR means a maximum data rate of 200% of the target, while 110% constrained VBR means a maximum rate of 110% of the target. You should constrain all VBR encodes produced for streaming because if data rate spikes get too high, you may experience problems delivering the files under constrained conditions like 3G and 4G.

When choosing between VBR and CBR, you should consider three elements, overall quality, transient quality, and file deliverability.

1. Overall quality

First is overall quality, which is shown in Table 4-1. As with most tables in this book, the number with the green background is the best quality, while the number with the red background is the worst. As you can see, 200% constrained VBR delivers the best quality in almost all cases, while one or two pass CBR delivers the worst in seven of eight files.

PSNR	200% VBR	150% VBR	110% VBR	CBR 2-Pass	CBR 1-Pass	Total Quality Delta	Delta - 110% to 200%
Tears of Steel	41.97	41.89	41.60	41.40	41.41	1.36%	0.88%
Sintel	41.34	41.13	40.67	40.56	40.17	2.83%	1.64%
Big Buck Bunny	41.73	40.98	40.00	39.70	40.07	4.88%	4.14%
Talking Head	44.23	44.22	44.17	44.12	44.15	0.25%	0.14%
Freedom	42.06	42.02	41.84	41.83	41.65	0.98%	0.53%
Haunted	42.07	42.07	42.01	41.90	42.06	0.40%	0.15%
Tutorial	46.81	46.56	45.27	45.08	44.71	4.49%	3.29%
Screencam	39.71	38.31	36.89	36.96	40.01	7.80%	7.11%
1080p Average	**42.49**	**42.15**	**41.56**	**41.44**	**41.78**	**2.46%**	**2.20%**
720p Average	**41.35**	**41.18**	**40.82**	**40.77**	**40.76**	**1.71%**	**1.28%**

Table 4-1. PSNR quality using different bitrate control techniques.

On the other hand, the quality difference in the Total Quality Delta column averages only 2.46% for 1080p video, and 1.71% for 720p video, which few, if any, viewers would notice. So, while VBR is widely (and accurately) touted as delivering the best possible quality, the difference isn't as dramatic as you might think.

2. Transient quality.

The big issue with CBR is that quality can drop precipitously for one or two frames in high-motion sequences (Figure 4-1). While this doesn't happen all that frequently, it's still the most important reason to avoid CBR.

3. Deliverability

The final consideration for choosing a bitrate control technique is the ability to deliver the file over constrained connections. This is shown in Figure 4-4, which shows two views of an application called Bitrate Viewer. The top view shows the bitrate of a CBR-encoded file, which is relatively flat, the bottom the bitrate of the VBR-encoded file, which shows significant variances in data rate. Which would you rather deliver over a 3G connection?

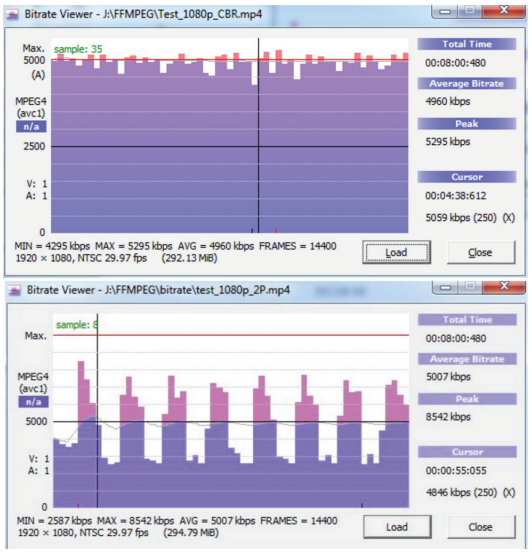


Figure 4-4. CBR file on top, VBR on the bottom.

I examined the impact of data rate control technique in an article on the *Streaming Learning Center* entitled Bitrate Control and QoE-CBR is Better, which you can read at bit.ly/vbr_cbr_qoe. Using a specially constructed file that contained 30 seconds of talking head followed by 30 seconds of high-motion ballet footage, I showed how 200% constrained VBR can degrade the quality of experience (QoE) of viewers on constrained connections.

Overall, the article recommends producing streaming files using between 110% and 150% constrained VBR, though there are some contrary views. For example, while Apple previously recommended not exceeding 110% constrained VBR for streams encoded for HLS, they boosted this to 200% constrained VBR in 2016, though they cited no QoE-related data when making this change (bit.ly/hls_spec_2017).

On the other side of the coin, there are still many producers who swear by CBR. Overall, given the lack of significant quality differences between 110% and 200% constrained VBR, and the deliverability risk proven by the aforementioned article, I think 110%-150% is the safer choice.

There's one more concept you need to understand before tackling data rate control using FFmpeg. That's VBV, and it's our next stop.

> ***Tip:*** *The application you see in Figure 4-4 is Bitrate Viewer, a free Windows app you can download at bit.ly/brv_dl. It's a great tool, but it only works with H.264 and MPEG-2 files, not HEVC or VP9. For a tutorial on the tool (and MediaInfo), check out bit.ly/videoanalyze.*

A Quick Word on VBV

VBV stands for Video Buffering Verifier, and it refers to how much video data is stored (or cached) in the player. As you'll see, when setting the bitrate with FFmpeg, you'll set the target bitrate, maximum bitrate, and VBV buffer size.

In general, the larger the buffer size, the higher the quality and the greater the variability in data rate within the stream. If you're encoding with CBR and need a really consistent data rate, you should keep the buffer small, usually the same size as a single second of video data, which will become crystal clear in a moment. If you're producing using 200% constrained VBR and don't necessarily care about data rate consistency, using 2 seconds of data is acceptable.

Bitrate Control and Buffer Size in FFmpeg

Implementing the bitrate control technique and buffer size in FFmpeg is simple. To illustrate how, and the effect of each technique, I'll use the test video file that I created for the QoE article mentioned above, which again, was eight minutes long, and alternates 30 seconds of talking head video with 30 seconds of high-motion ballet.

To set the bitrate target in FFmpeg, use the `-b:v` code (bitrate:video) below :

```
ffmpeg -i Test_1080p.MP4 -c:v libx264 -b:v 5000k Test_DR_5M.mp4
```

Batch 4-1. Producing a file using the `-b:v` *command and one-pass encoding.*

This produces a file that looks like Figure 4-5, where the data rate would vary according to content. The overall data rate of 5,062 kbps is pretty accurate, but you'd be concerned that data rate spikes in the file could hinder deliverability. The answer? Two-pass encoding.

How do you control data rate with two-pass encoding? Using two new controls, maximum bitrate and VBV buffer size. That is, you use:

`-b:v 5000k` as the target, as before.

`-maxrate 5000k` to set the maximum bitrate. So, 5000k would be CBR, 5500k would be 110 percent constrained VBR, and 10000k would be 200 percent constrained VBR.

`-bufsize 5000k` to set the size of the VBV. For this real-world video distributed via streaming, I'd use a VBV size equivalent to the data rate of one second of video (5000k).

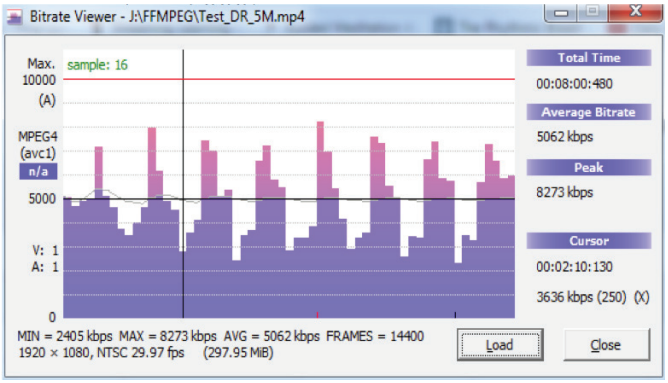


Figure 4-5. One-pass encoding at 5 Mbps.

***Tip**: Note that if you forget the "k" in the bitrate (5000**k**), FFmpeg will encode to bytes, not kilobytes. If you find your encoded files abnormally small, it's likely that you forgot the k.*

Two-Pass Encoding in FFmpeg

To implement two-pass encoding in FFmpeg, you define both passes, each in their own line. This is what the two lines would look like.

```
ffmpeg -y -i  test_1080p.mp4 -c:v libx264  -b:v 5000k -pass 1 -f mp4 NUL
&& \

ffmpeg -i  test_1080p.mp4 -c:v libx264  -b:v 5000k -maxrate 5000k
-bufsize 5000k -pass 2  test_1080p_CBR.mp4
```

Batch 4-2. Producing a CBR file with two-pass encoding.

Here is a description of the new controls added to the command line.

Line 1. During this pass, FFmpeg scans the file and records analysis data in a log file.

`-y` overwrites existing log file. If you encode multiple files using FFmpeg, this tells the program to overwrite the existing log file. Without `-y`, FFmpeg will stop the batch to ask if you want to overwrite the log file each encode. Or you can name the log file for each encode with the `-passlogfile` switch.

`-pass 1` completes the first pass and creates the log file but no output file.

`-f mp4` identifies the output format used in the second pass.

`NUL` creates the log file.

`&& \` tells FFmpeg to run a second pass if the first pass was successful.

Line 2. During this pass, FFmpeg uses the log created in the first pass to encode the file.

`-b:v 5000k` sets the overall target.

`-maxrate 5000k` sets the maximum bitrate. It's the same as the target, so this means CBR.

`-bufsize 5000k` sets the size of the VBV.

`-pass 2` finds and uses the log file for the encode.

`test_1080p_CBR.mp4` sets the output file name.

Figure 4-6 shows the CBR-encoded file in Bitrate Viewer. Although the file isn't a total flat line, there's much less data rate variability than in Figure 4-5, and the file would be much simpler to deliver. Of course, overall quality is slightly lower than VBR, and there's a risk of transient quality problems.

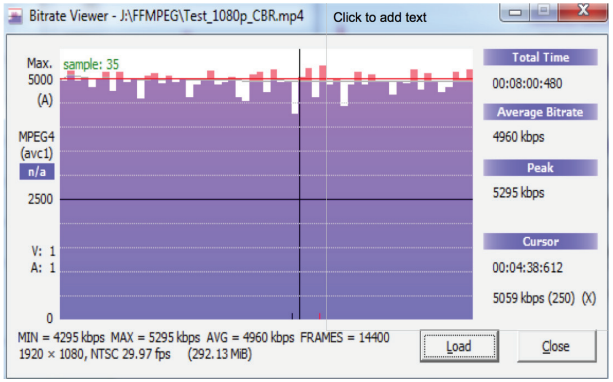


Figure 4-6. Two-pass CBR encoding with FFmpeg.

Quick Summary: Constant Bitrate Encoding

1. CBR delivers the lowest quality stream with occasional transient issues but is the easiest stream to deliver.

2. You produce a CBR stream by using the same value for target and maximum bitrate in either a single or two-pass encode.

3. When producing CBR files, you should use a VBV buffer of one-second of video.

200 Percent Constrained VBR Encoding in FFmpeg

Here's how to produce 200% constrained VBR in FFmpeg. The first line is the same, but I've boosted `-maxrate` to `10000k` in the second line.

```
ffmpeg -y -i  test_1080p.mp4 -c:v libx264  -b:v 5000k -pass 1  -f mp4 NUL
&& \

ffmpeg -i  test_1080p.mp4 -c:v libx264  -b:v 5000k -maxrate 10000k -buf-
size 5000k -pass 2  test_1080p_200p_CVBR.mp4
```

Batch 4-3. Producing a 200% constrained VBR file with two-pass encoding.

Figure 4-7 shows the 200% constrained VBR file in Bitrate Viewer. Quality would be optimal, and there should be no transient quality problems. Again, however, with this worst-case file with mixed high- and low-motion footage, deliverability might be a real issue.

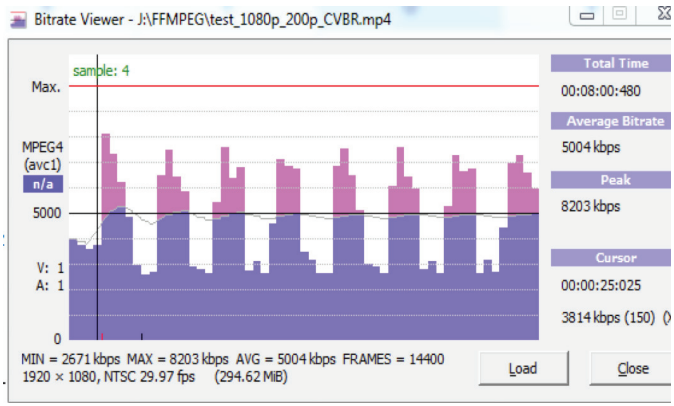


Figure 4-7. Two-pass 200 percent constrained VBR encoding with FFmpeg.

110 Percent Constrained VBR Encoding in FFmpeg

Here's how to produce 110% constrained VBR in FFmpeg. The first line is the same as the previous two, but `-maxrate` in pass two is limited to `5500k`.

```
ffmpeg -y -i  test_1080p.mp4 -c:v libx264  -b:v 5000k -pass 1  -f mp4 NUL
&& \

ffmpeg -i  test_1080p.mp4 -c:v libx264  -b:v 5000k -maxrate 5500k -bufsize
5000k -pass 2  test_1080p_110p_CVBR.mp4
```

Batch 4-4. Producing 110% constrained VBR file with two-pass encoding.

Figure 4-8 shows the 110% constrained VBR file in Bitrate Viewer. The data rate is very similar to the other two, of course—although the peak bitrate is 5,852 kbps compared with 5,295 for

CBR. While quality would be slightly less than 200 percent constrained VBR, there should be no transient quality problems, and the file should be pretty simple to deliver.

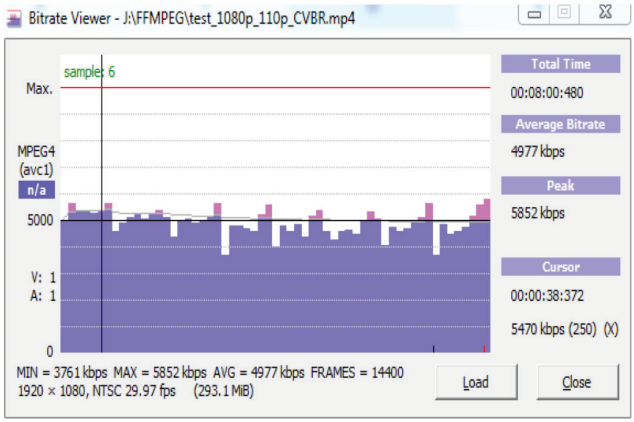


Figure 4-8. Two-pass 110 percent constrained VBR encoding with FFmpeg.

Tip: *Running multiple FFmpeg encodes simultaneously is a great way to speed up your multiple file encoding chores—particularly on a multiple-core computer. Be careful when encoding multiple files in the same folder using two-pass encoding, however, since you'll be creating multiple log files that will overwrite each other and ruin the second encode. You can separately name the log file using the* `-passlogfile` *switch, or simply run the different encodes from different folders, which is what I do.*

Quick Summary: Variable Bitrate Encoding

1. VBR delivers the highest quality stream with very few transient issues, but data rate swings can complicate delivery and degrade QoE.

2. All VBR encodes should be constrained by limiting the maximum bitrate. I recommend a maximum setting of 110-150% of the target.

3. All VBR encodes should be two-pass.

4. When producing VBR files for streaming, use a VBV buffer of between one and two seconds of video.

Constant Rate Factor (CRF) Encoding

When you encode using CBR and VBR, you choose a data rate and bitrate control technique, and FFmpeg attempts to meet that data rate using the selected bitrate control technique. With Constant Rate Factor (CRF) encoding, you choose a quality level and FFmpeg delivers that quality, adjusting the data rate up and down as needed. You get a file with a fixed quality level,

but unknown (in advance) data rate, and a file where the data rate varies significantly over the duration of the file, which may impact deliverability.

Using CRF

Figure 4-9 shows how CRF values affect quality; specifically, the lower the value, the higher the quality. It's counter-intuitive, but that's how it works.

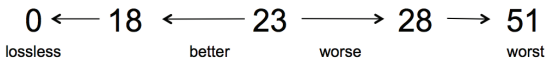


Figure 4-9. How CRF values affect video quality. From the CRF Guide.

How is CRF useful? Two ways. First, it's a measure of file complexity. That is, if you encode a talking head clip and a soccer clip using the same CRF value, the soccer clip will have a much higher data rate. That's because the increased motion and detail requires more data to achieve the same quality level.

The second way CRF is useful is as a bitrate control technique with a "capped" data rate, which is called capped CRF. It's almost easier to show than explain, so let's jump in with the FFmpeg controls for CRF and capped CRF encoding.

With plain CRF encoding, you insert a CRF value rather than a data rate as shown in Batch 4-5. Since the goal is quality, not a data rate target, all CRF (and capped CRF) encodes are single pass.

```
ffmpeg -i Test_1080p.MP4 -c:v libx264 -crf 23 Test_CRF23.mp4
```

Batch 4-5. Producing a file with a CRF value of 23.

> `-crf 23` tells FFmpeg to encode using a CRF value of 23. Note that in *Encoding by the Numbers*, we learned that a CRF value of 23 approximates the data rate (and quality) delivered by most Hollywood producers. That's why I used this value here.

So, you swap the `-crf` value for the data rate controls, producing the file shown in Figure 4-10.

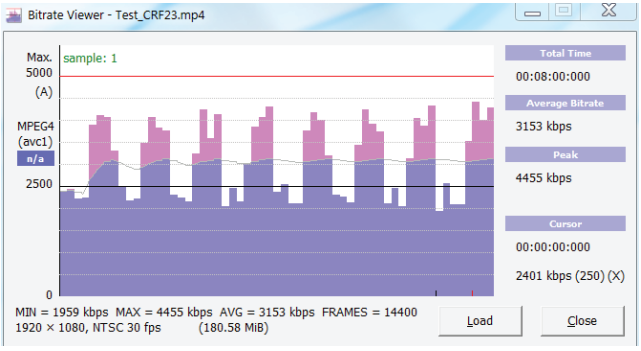


Figure 4-10. Test file encoded at CRF 23.

As you can see, the data rate varies from around 2400 for the talking head sections to around 4500 kbps for the ballet sections. The average data rate is 3153, about 40% less than the 5 Mbps used in previous encodes. This is the attraction of CRF encoding; it applies the data rate necessary to preserve quality, and that's it. The problem is, of course, we need a maximum data rate to ensure file deliverability.

Capped CRF

If we capped the data rate at 5 Mbps, the file would look very similar to Figure 4-10 because there's no section where the data rate would be capped–it never exceeds 5 Mbps. So, let's cap it at 3500 kbps with the following command string.

```
ffmpeg -i Test_1080p.mp4 -c:v libx264 -crf 23 -maxrate 3500k -bufsize
3500k Test_CRF23_3500.mp4
```

Batch 4-6. Producing a file with a CRF value of 23 and a cap of 3500 kbps.

`-crf 23` sets the CRF level.

`-maxrate 3500k` sets the maximum data rate.

`-bufsize 3500k` sets the buffer size.

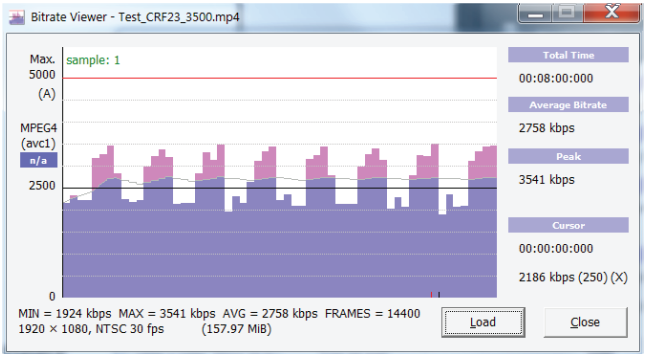


Figure 4-11. Test file encoded at CRF 23 and capped at 3500 kbps.

If you compare Figure 4-10 and 4-11, you'll see that the talking head sections are about the same data rate, but the maximum data rate has been restricted to around 3,500 kbps, as requested, and the overall data rate dropped from 3153 kbps to 2758 kbps.

Note that some companies use capped CRF for distribution, including online video platform vendor JWPlayer, who uses it for both H.264 and VP9. So, it's a proven, credible technique, despite the potential for data spikes within the file.

OK, that's it for data rate control, next up is setting resolution.

Contents

Acknowledgments	**3**
Introduction	**9**
Chapter 1: Video Boot Camp	**11**
Chapter 2: Installing FFmpeg and Batch File Operation	**20**
Chapter 3: Choosing Codecs and Container Formats	**30**
Chapter 4: Bitrate Control	**35**
Chapter 5: Setting Resolution	**46**
Chapter 6: Setting Frame Rate	**54**
Chapter 7: I-, B-, P-, and Reference Frames	**57**
Chapter 8: Encoding H.264	**66**
Chapter 9: Working with Audio	**77**
Chapter 10: Multipass Encoding	**81**
Chapter 11: Packaging HLS and DASH from H.264 Files	**86**
Chapter 12: Encoding HEVC	**106**
Chapter 13: Encoding VP9	**121**
Chapter 14: Miscellaneous Operations	**134**
Index	**151**

Acknowledgments 3

Introduction 9

About You 9
What You'll Get From This Book 10

Chapter 1: Video Boot Camp 11

What's a Codec? 12
Choosing a Codec 13

Container Formats 13

Configuration Basics 15

Video Resolution 16

Frame Rate 17

Bitrate (or Data Rate) 17

Compression and Ben and Jerry's Ice Cream 18

About Video Quality Metrics 18

Chapter 2: Installing FFmpeg and Batch File Operation 20

About FFmpeg 20

Installing FFmpeg 21

Installing on Ubuntu 21
Installing on Windows 21
Installing on the Mac 22

Working with Batch Files 23

Batch Files for Mac and Linux 24
Introduction to Batch File Creation and Operation 24
Essential Command Line Commands 26
Debugging Batch Files 27

Working with Continuation Characters 27

Chapter 3: Choosing Codecs and Container Formats 30

Designating the Codecs in FFmpeg 30

Other Codecs 32
Designating the Container Format in FFmpeg 33
Changing the Container Format in FFmpeg 33

Chapter 4: Bitrate Control 35

CBR and VBR Defined 36

A Quick Word on VBV 39

Bitrate Control and Buffer Size in FFmpeg 39

Two-Pass Encoding in FFmpeg 40
200 Percent Constrained VBR Encoding in FFmpeg 42
110 Percent Constrained VBR Encoding in FFmpeg 42
Constant Rate Factor (CRF) Encoding 43
Using CRF 44
Capped CRF 45
Chapter 5: Setting Resolution 46
Setting Resolution in FFmpeg 46
-s for Simple 46
Pixel Aspect Ratio and Display Aspect Ratio 47
Target Width and Compute Height (1280 x ?) 49
Target Height Compute Horizontal (? x 720) 50
Target 1280x720 resolution and Crop Excess Pixels 51
Target 1280x720 resolution and Letterbox 52
Chapter 6: Setting Frame Rate 54
Overview 54
When to Cut the Frame Rate? 55
Other Considerations in the HLS Specification 56
Chapter 7: I-, B-, P-, and Reference Frames 57
Frame Overview 57
I-frames and Single Files 58
I-frames and Adaptive Streaming 59
Inserting I-frames at Specified Intervals and Scene Changes 60
Working with B-frames 61
Inserting B-frames in FFmpeg 62
Reference Frames 63
Reference Frames in FFmpeg 64
Chapter 8: Encoding H.264 66
What Is H.264? 66
Basic H.264 Encoding Parameters 67
Profiles and Levels 67
Comparative Quality—Baseline, Main, and High Profiles 68
Choosing Profiles in FFmpeg 68
H.264 Levels 69
Levels and Computers/OTT 69
Setting Levels in FFmpeg 69
Entropy Coding 70
Setting Entropy Encoding in FFmpeg 71
x264 Presets and Tuning 71

x264 Presets 72
Choosing an x264 Preset 74
Tuning Mechanisms **75**
Animation Tuning 75
Film and Grain Tuning 76
Choosing an x264 Tuning Mechanism **76**
Chapter 9: Working with Audio **77**
Which Audio Codec? **77**
Dolby Digital 77
Opus Audio for VPX 78
Controlling Audio Parameters **78**
Bitrate 78
Sample Rate 79
Channels 79
Putting it All Together **80**
Chapter 10: Multipass Encoding **81**
Multiple-File Encoding in FFmpeg **81**
Extracting Audio or Video 84
Putting it All Together 85
Chapter 11: Packaging HLS and DASH from H.264 Files **86**
Packaging Existing MP4 Files 88
Creating HLS Output from Scratch 89
Creating the Master Playlist File 90
Working with Bento4 **93**
Creating HLS Output with Bento4 mp4hls 96
Creating DASH /HLS Output with Bento4 mp4dash **98**
Convert to fMP4 Format with mp4fragment 99
Creating the Manifest Files with mp4dash 100
Working with Apple Tools **102**
Media File Segmenter 102
Media Stream Validator 105
Which Stream First? 105
Chapter 12: Encoding HEVC **106**
What is HEVC **106**
HEVC Profiles 107
x265 Presets 108
Our HEVC Encoding Ladder 109
x265 and FFmpeg **110**

1080p Conversion 111
4K Scaling Exercise 112
Producing HLS and DASH from HEVC Files 113
HEVC in HLS 113
Introducing mp4dash 115
Convert to fMP4 Format with mp4fragment 116
Creating the Manifest Files with mp4dash 118
Chapter 13: Encoding VP9 121
About VP9 121
Basic VP9 Encoding Parameters 122
Other Configuration Options 122
Advice from the Stars 127
Our VP9 Encoding Ladder 128
VP9 and FFmpeg 128
1080p Conversion 128
4K Scaling Exercise 129
Deploying VP9 in DASH 130
Encoding VP9 Files for DASH Distribution 130
Packaging VP9 Files for DASH 131
Chapter 14: Miscellaneous Operations 134
Working With YUV/Y4M Files 134
Converting with FFmpeg 135
Scaling in FFmpeg 135
Computing PSNR with FFmpeg 136
Concatenating Multiple Files 137
Extract Files Without Re-encoding. 138
Burning Text into a Video File 139
How to Deploy Multiple Video Filters 140
Encoding with AV1 141
AV1 Encoding Modes 142
Other AV1 Encoding Controls 144
Live H264 Transcoding with FFmpeg 145
Choosing the Preset 147
Live Transcoding with HEVC 148
Live Transcoding with VP9 149