A Comparative Study of Android and iOS for Accessing Internet Streaming Services

Yao Liu @ George Mason University
Fei Li @ George Mason University
Lei Guo @ Ohio State University
Bo Shen @ Vuclip
Songqing Chen @ George Mason University

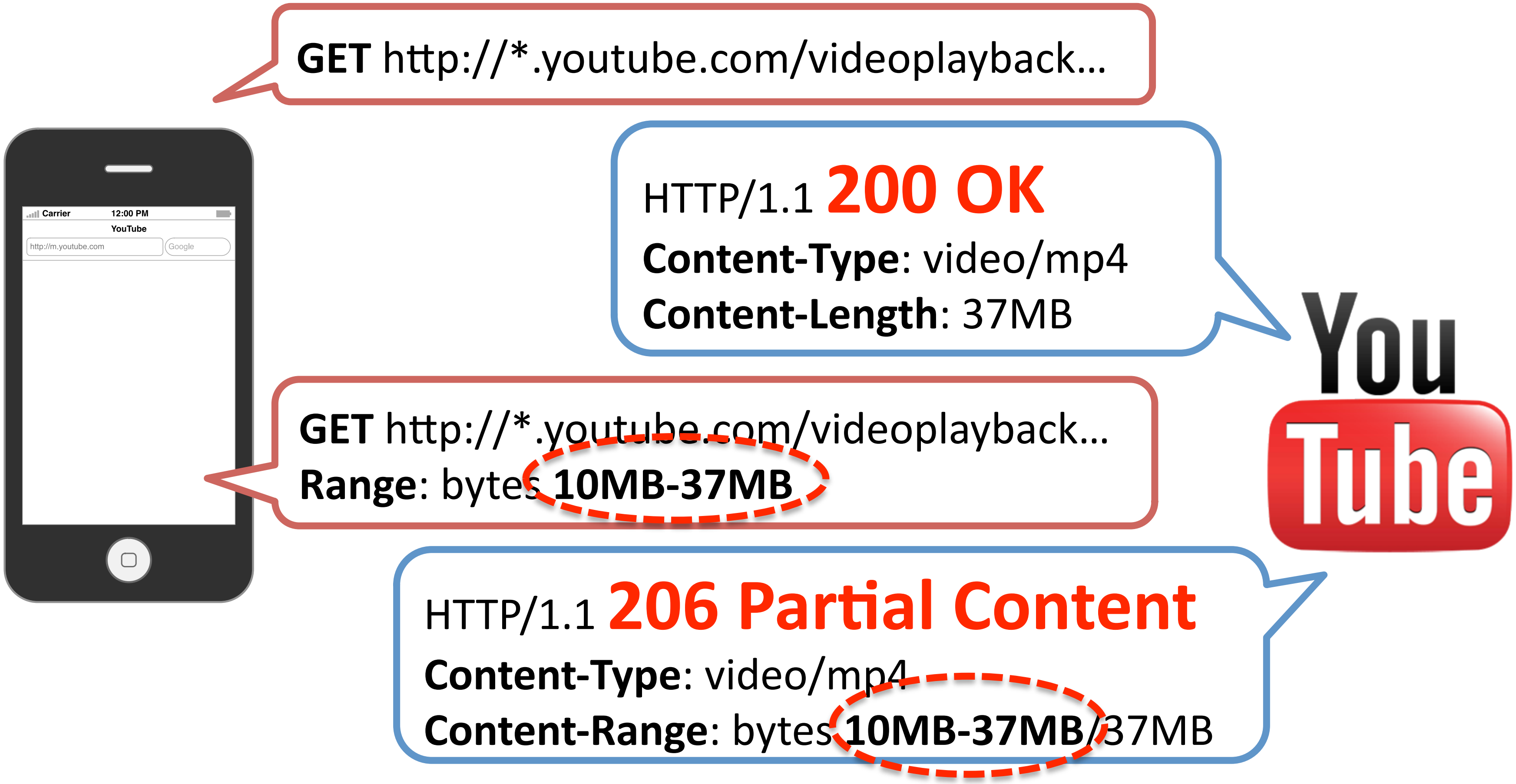
Internet streaming to mobile devices

- Internet streaming services receive an increasing number of access from mobile devices
  - 25% of YouTube views are from mobile devices[1]

- Today, 50% of mobile network traffic is video[2]
- By 2016, this number will grow to 66%[2]

[1] Data source: YouTube
[2] Data source: Cisco

Pseudo streaming is used in mobile Internet streaming



GET http://*.youtube.com/videoplayback…

HTTP/1.1 **200 OK**
Content-Type: video/mp4
Content-Length: 37MB

GET http://*.youtube.com/videoplayback…
Range: bytes **10MB-37MB**

HTTP/1.1 **206 Partial Content**
Content-Type: video/mp4
Content-Range: bytes **10MB-37MB**/37MB

Server-side observations (from VUCLIP)

- Server-side log during February 2011
- 26,713,708 HTTP requests
- 15,725 video clips
- 27.4 TB video traffic

- 397,940 unique video sessions from iOS devices
- 884,648 unique video sessions from Android devices

iOS and Android use different content requesting approaches

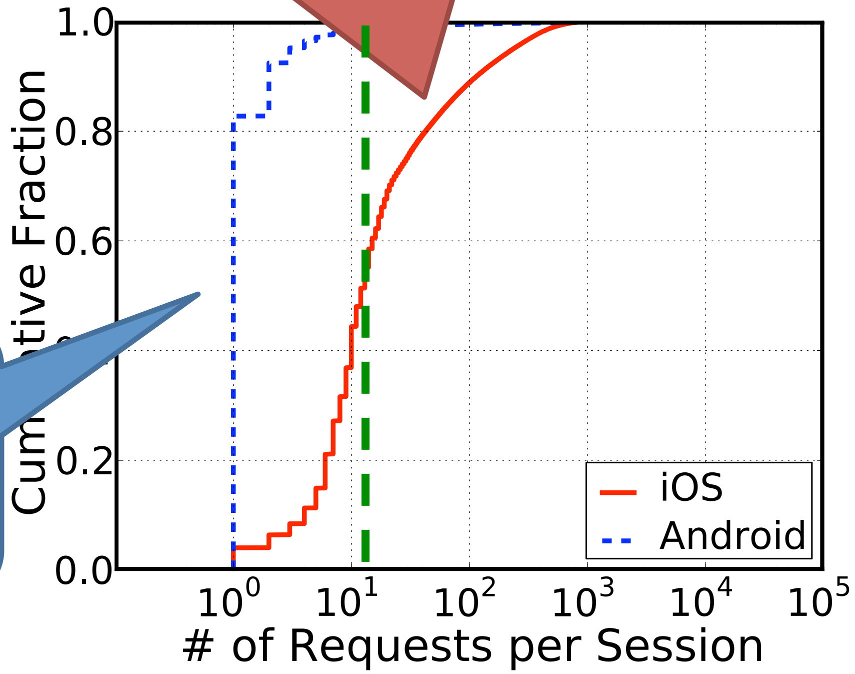
iOS		
	HTTP 200	HTTP 206
% of Requests	0.01	99.99
% of Traffic Amount	0.001	99.999

Almost all iOS traffic is delivered using HTTP partial content response (206)

Android		
	HTTP 200	HTTP 206
% of Requests	27.30	72.70
% of Traffic Amount	80.594	19.406

80% of Android traffic is delivered using standard HTTP responses (200)

More requests are sent out by iOS devices

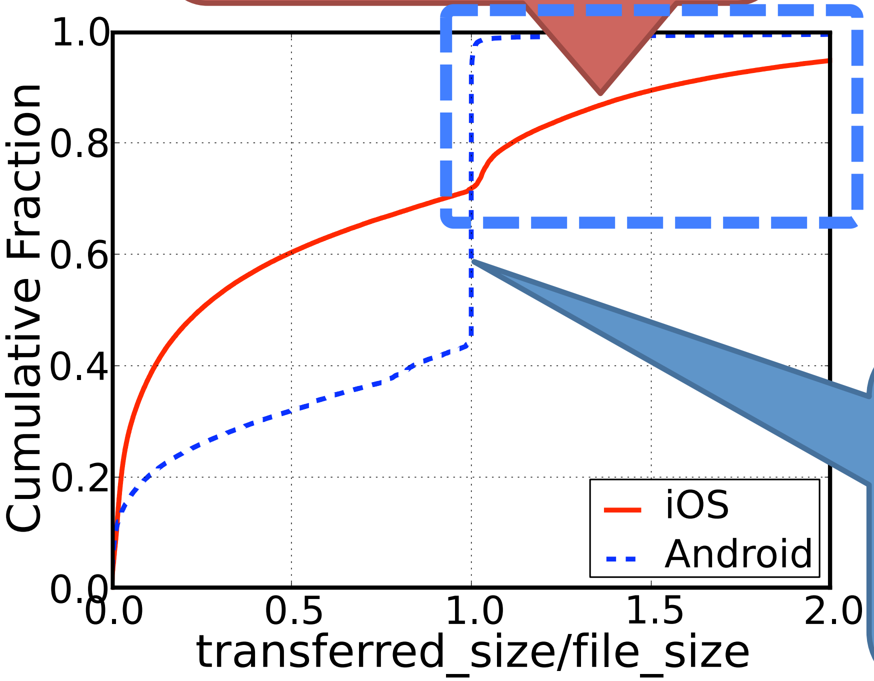


50% iOS sessions issued more than 12 HTTP requests per session

80% of Android sessions use only one single HTTP request

More traffic is received at iOS devices

28% iOS sessions received more traffic than the file size



55% Android sessions downloaded exact the same amount of traffic as the file size

Highlights of server-side observations

- iOS uses **HTTP range requests**, while Android uses **standard HTTP requests**

- **Multiple** HTTP requests are issued when iOS devices are watching streaming video, while 80% Android sessions use **only one** HTTP request

- 28% iOS sessions received **more traffic** than the video file size, while only 2% for Android

Devices used for client-side experiments

Name	OS version	Memory size
iPod Touch	iOS 3.1.2	128 MB
iPhone 3G	iOS 4.2.1	128 MB
iPhone 3GS	iOS 5.0.1	256 MB
iPhone 4S	iOS 5.1	512 MB
Nexus One	Android 2.3.4	512 MB
Kindle Fire	Android 2.3.4	512 MB

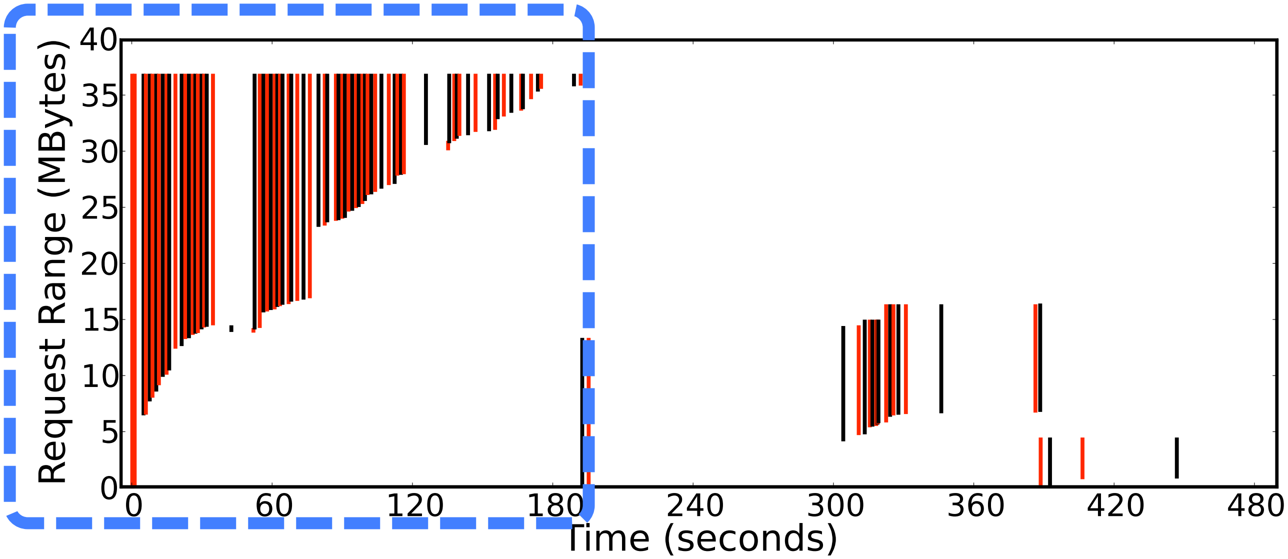
Watching an 8-minute YouTube video on iOS devices (1)

- 8-minute video, 360P
- 38,517,389 bytes (36.7 MBytes)

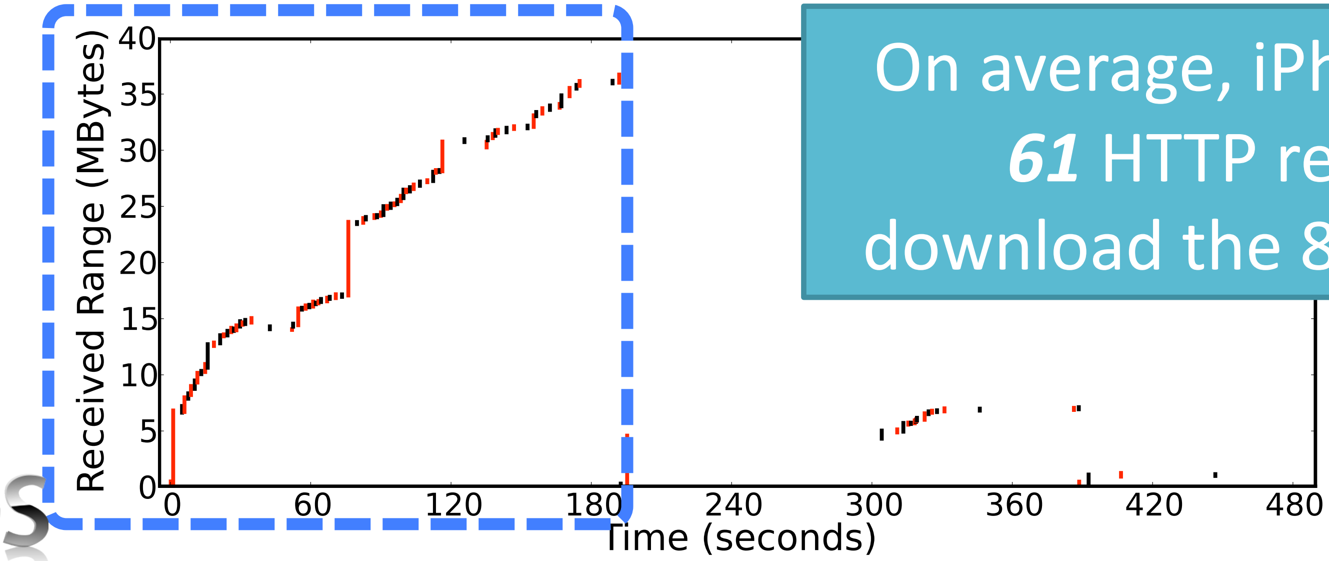
	# of HTTP requests
iPod Touch	261
iPhone 3G	301
iPhone 3GS	105
iPhone 4S	67

Multiple HTTP requests were issued to download the streaming data

iOS devices frequently abort HTTP connections

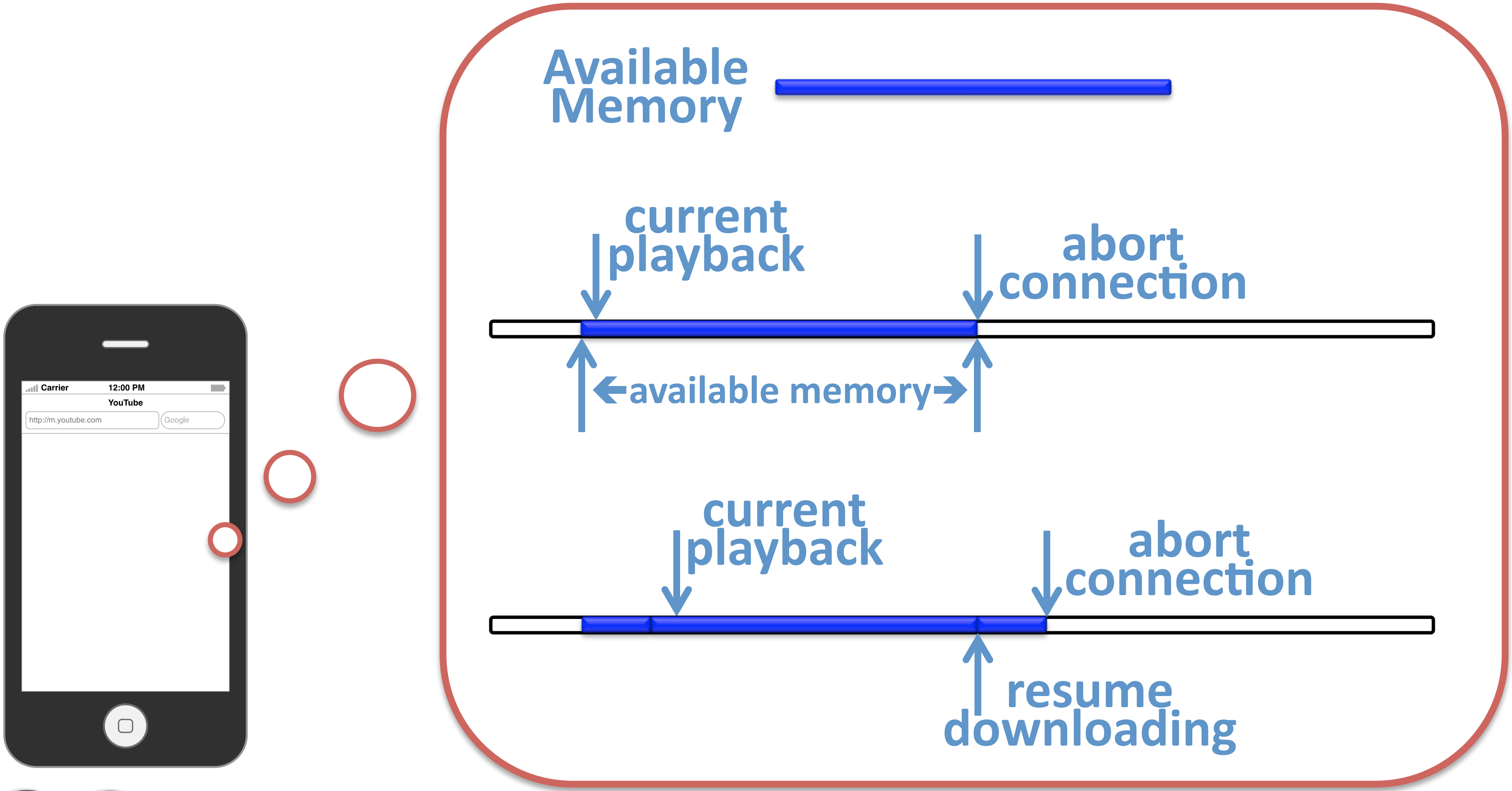


iPhone 3GS



On average, iPhone 3GS uses ***61*** HTTP requests to download the 8-minute video

Available memory is filled up, causing connections to be aborted



Available Memory

current playback

abort connection

←available memory→

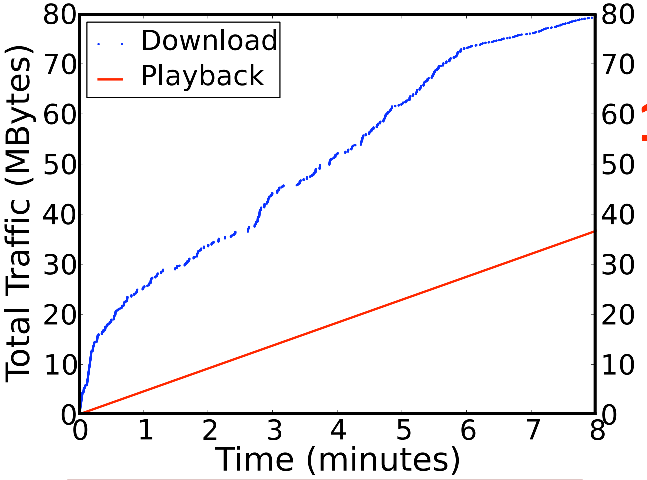
current playback

abort connection

resume downloading

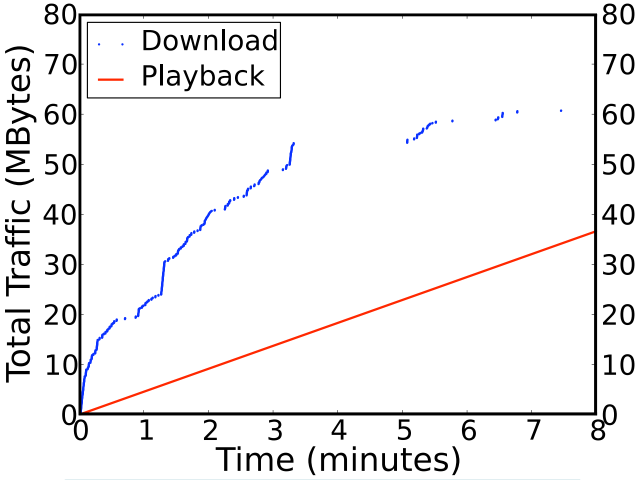
iOS

Watching an 8-minute YouTube video on iOS devices (2)



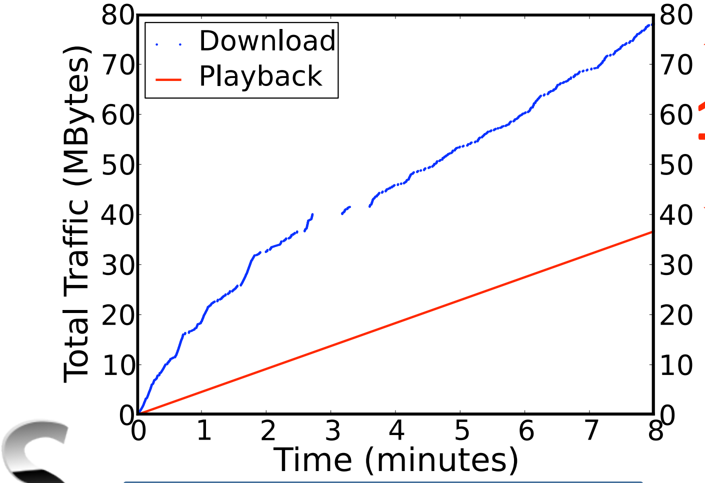
116%

iPod Touch



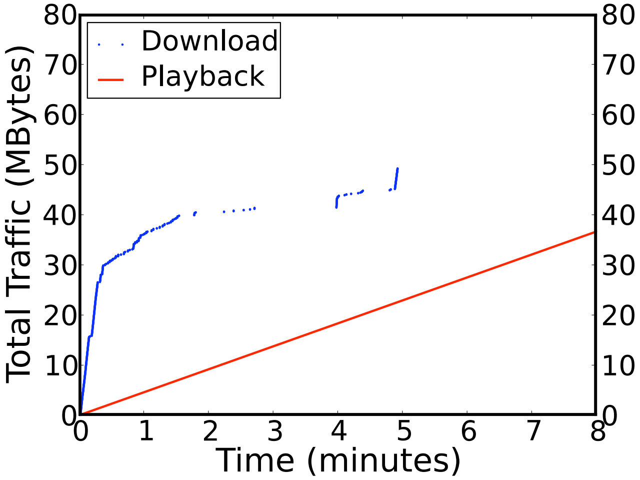
65%

iPhone 3GS



114%

iPhone 3G



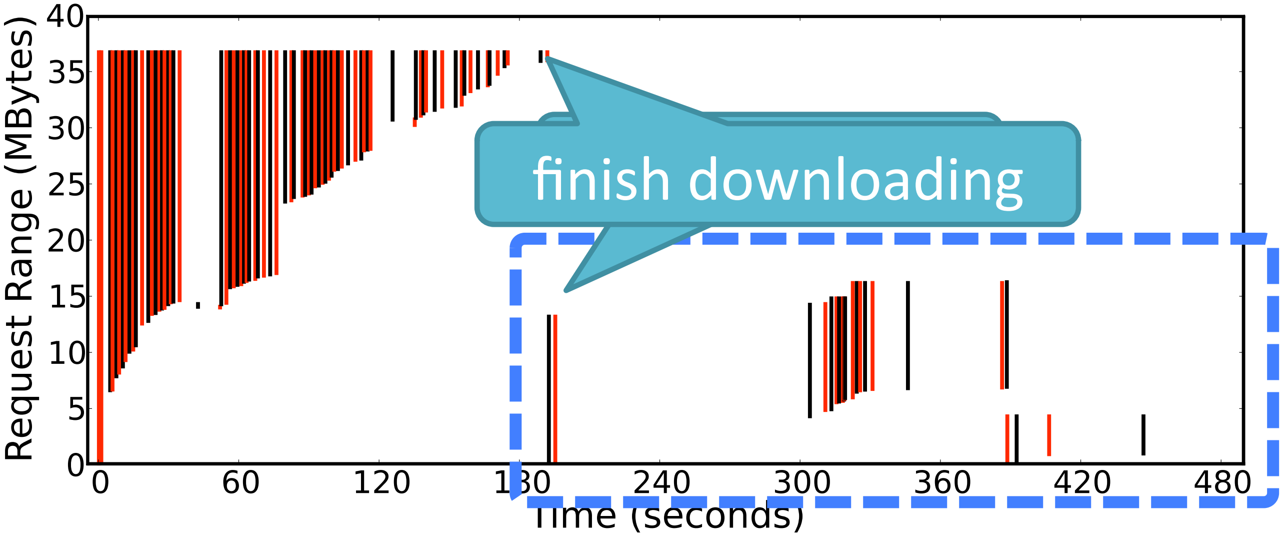
34%

iPhone 4S

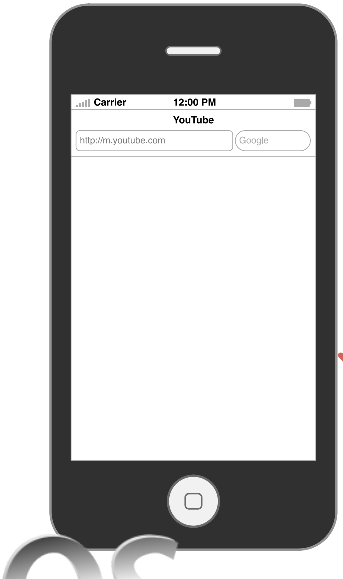
iOS

Re-downloading causes redundant traffic

iPhone 3GS



finish downloading



[FIN, ACK] (received full video content)

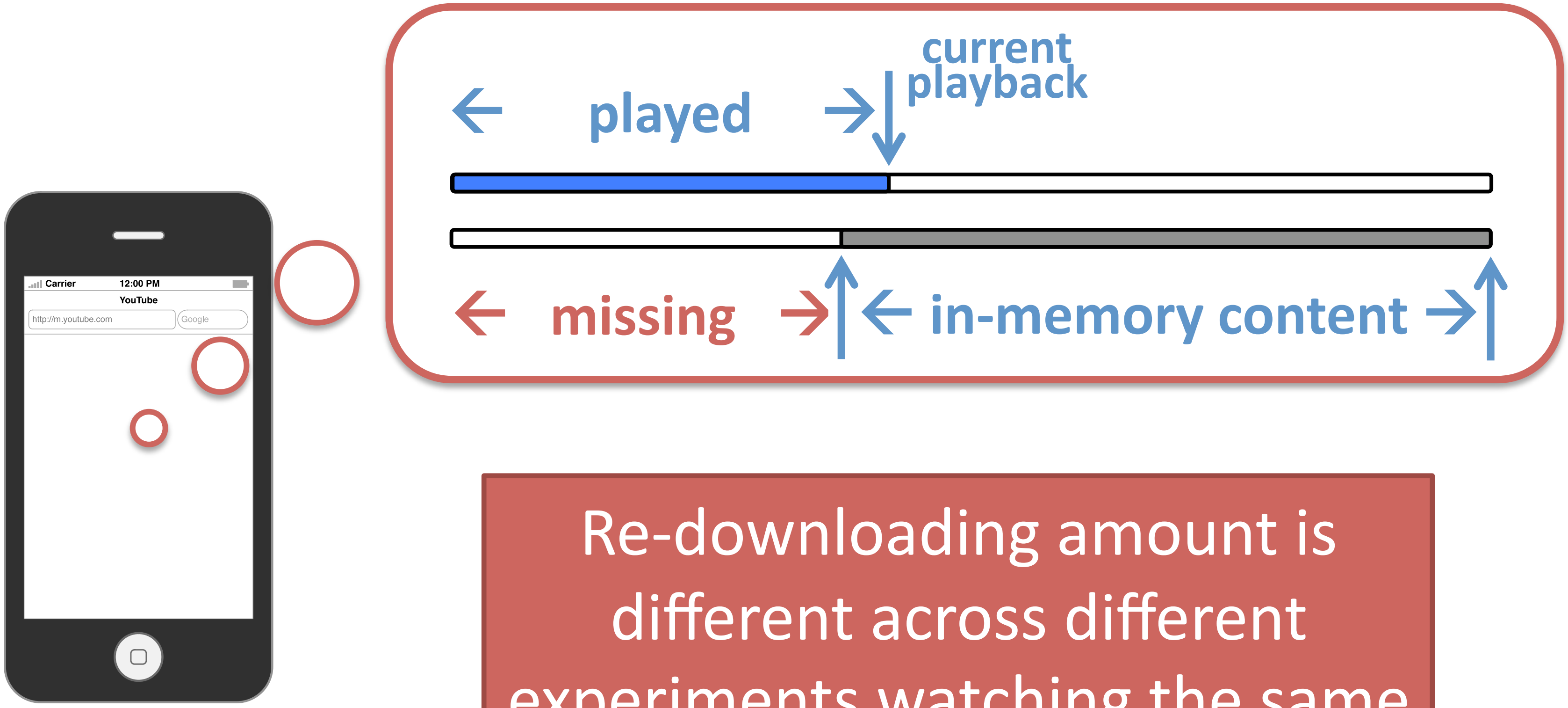
GET http://*.youtube.com/videoplayback…
Range: bytes **0-16MB**

Watching an 8-minute YouTube video on iOS devices

- 8-minute video, 360P
- 38,517,389 bytes (36.7 MBytes)

	# of HTTP requests	Received HTTP body (Bytes)	Re-downloaded (Bytes)
iPod Touch	261	83,410,351	26,450,851
iPhone 3G	301	82,616,828	37,449,911
iPhone 3GS	105	63,713,281	11,523,915
iPhone 4S	67	51,625,429	9,292,410

Limited available memory causes re-downloading



current playback

← played →

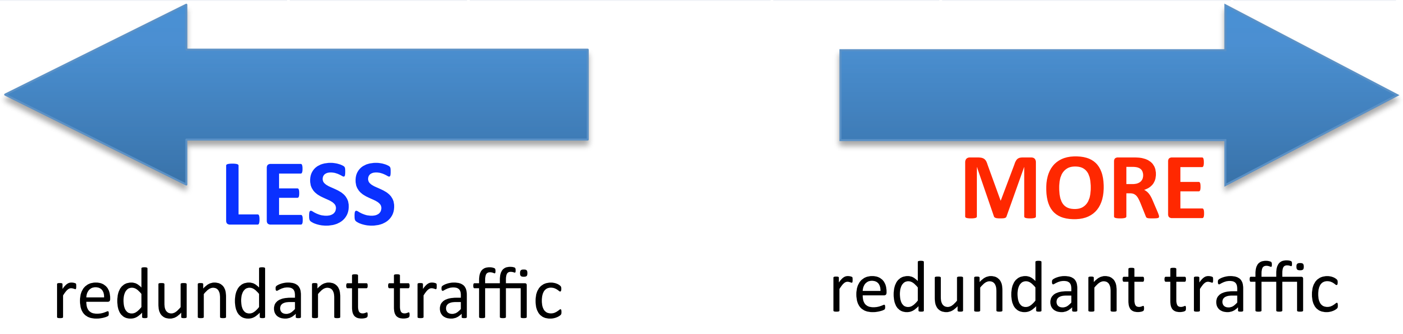
← missing →

← in-memory content →

Re-downloading amount is different across different experiments watching the same video on the same device

Video file size vs. redundant traffic amount

	Video 1		Video 2		Video 3	
Duration (sec)	*360*		*480*		*657*	
File size (Bytes)	*29,503,221*		*38,517,389*		*53,405,910*	
iPod Touch	42,379,164	**144%**	57,176,659	**148%**	90,445,044	**169%**
iPhone 3G	42,322,498	**143%**	74,442,375	**193%**	86,933,886	**163%**
iPhone 3GS	37,702,143	**128%**	47,460,396	**123%**	72,388,936	**136%**
iPhone 4S	32,248,384	**109%**	44,538,836	**116%**	61,731,408	**116%**



LESS

redundant traffic

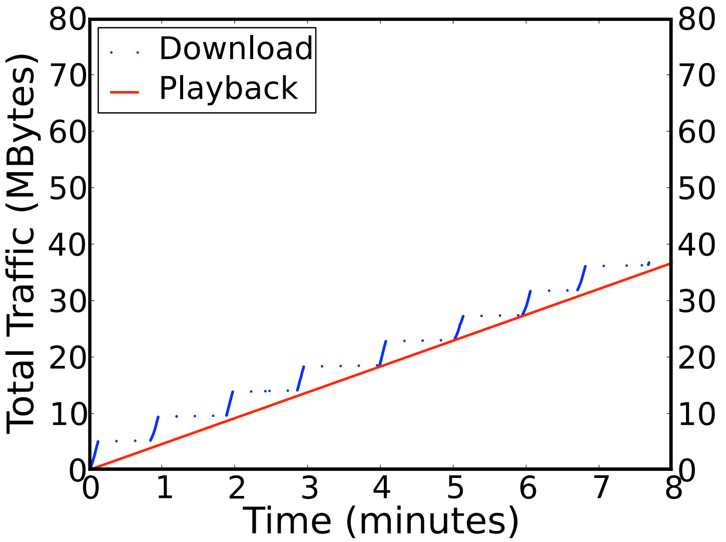
MORE

redundant traffic

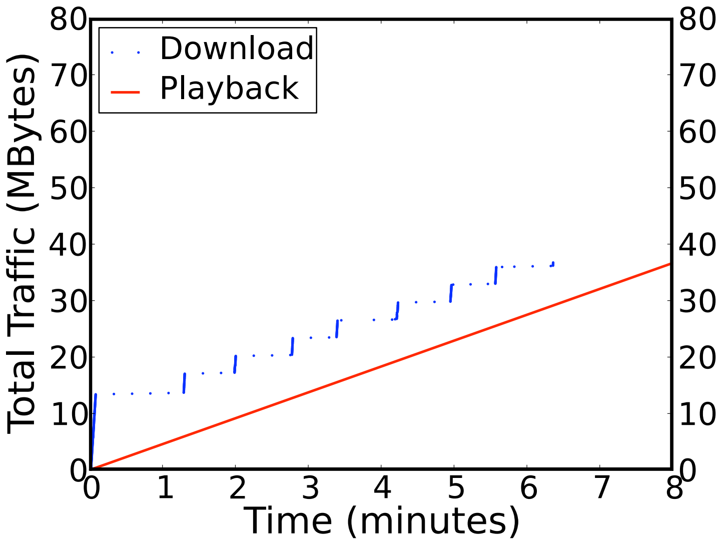
Findings with iOS

- **Dynamic** buffer management
  - The HTTP connection is aborted when the playback buffer fills up, causing **multiple** HTTP range requests to be used in one streaming session
  - Downloading resumes when more memory is supplied by the operating system
  - Uses **re-downloading** to fetch played content that has been evicted from the memory
  - Causes **redundant** traffic

Watching the same 8-minute YouTube video on Android devices



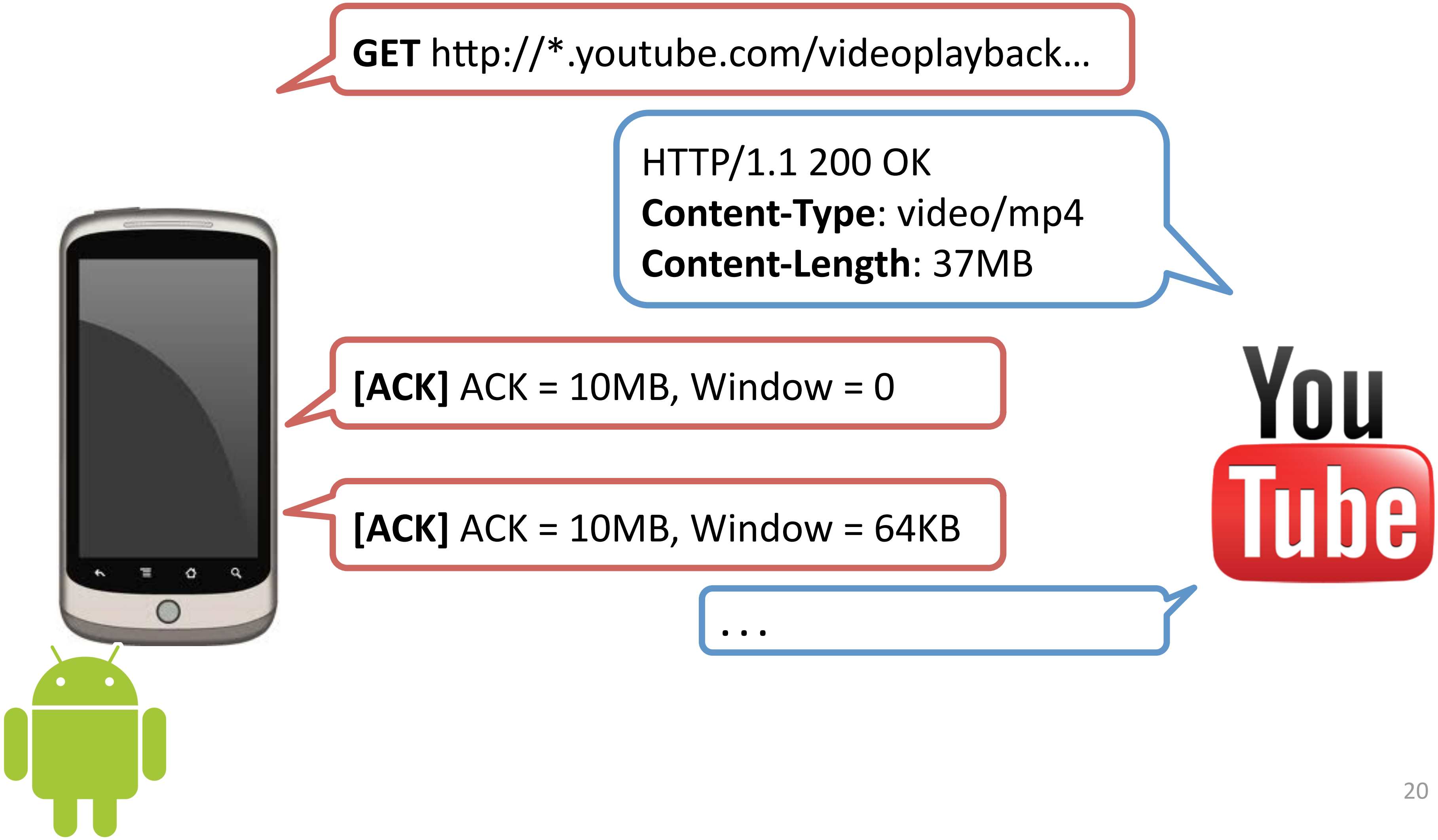
Nexus One



Kindle Fire

	# of HTTP requests	Received HTTP body (Bytes)	Re-downloaded (Bytes)
Nexus One	1	38,517,389	0
Kindle Fire	1	38,517,389	0

Android uses TCP window to control downloading



GET http://*.youtube.com/videoplayback…

HTTP/1.1 200 OK
Content-Type: video/mp4
Content-Length: 37MB

[ACK] ACK = 10MB, Window = 0

[ACK] ACK = 10MB, Window = 64KB

. . .

```
enum {
kPageSize = 65535,
kDefaultHighWaterThreshold = 20 * 1024 * 1024,
kDefaultLowWaterThreshold = 4 * 1024 * 1024,
kDefaultKeepAliveIntervals = 15000000,
};
```

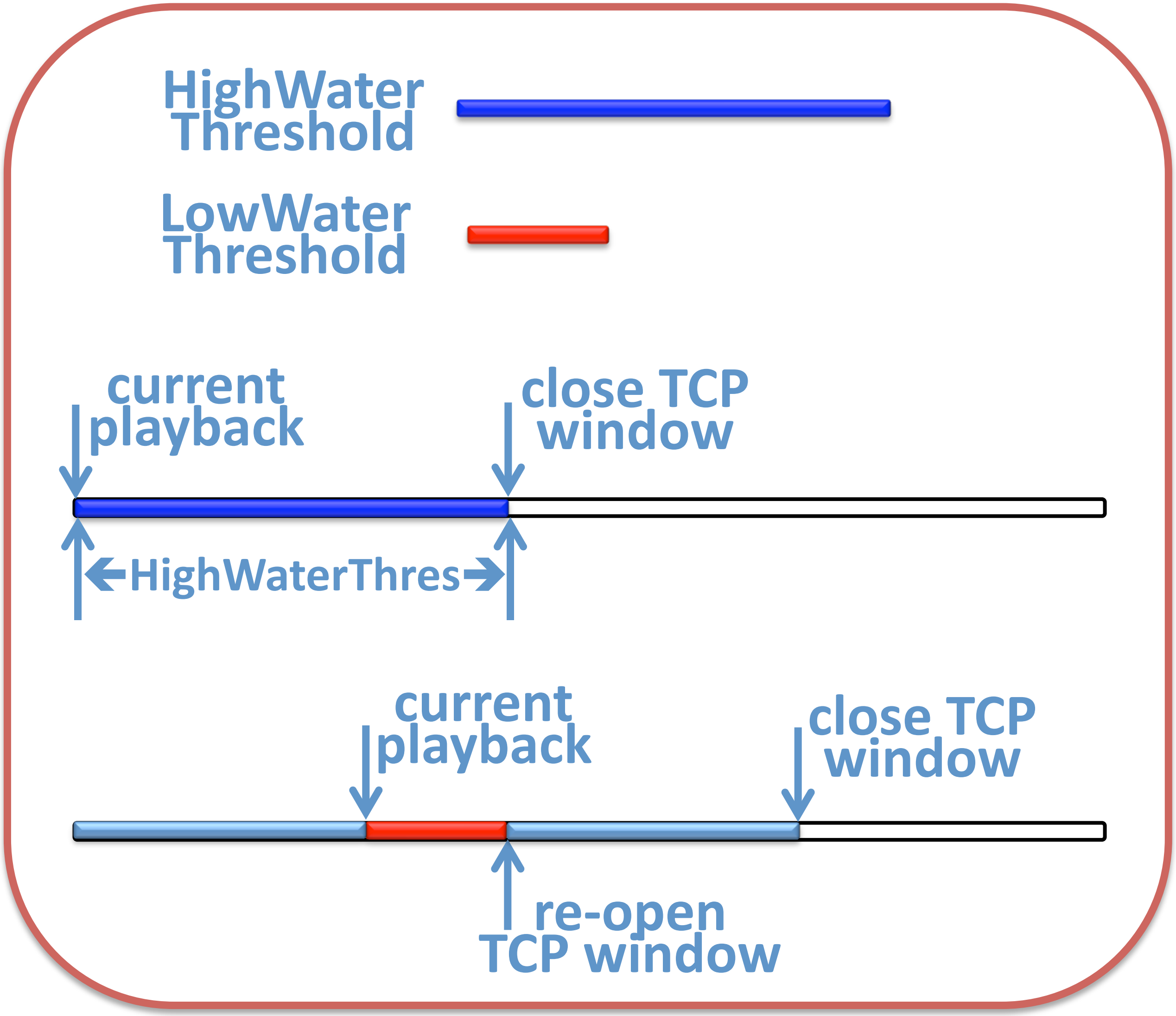
20 MB

4 MB

Code snippet from:

/libstagefright/include/NuCachedSource2.h

Static buffer management in Android



HighWater Threshold

LowWater Threshold

current playback

close TCP window

←HighWaterThres→

current playback

close TCP window

re-open TCP window

Findings with Android

- **Static** buffer management
  - Keeps a **fixed** amount of data in the buffer (HighWaterThreshold e.g., 20 MB)
  - Downloading is strictly synchronized with the playback progress (LowWaterThreshold)
  - **No redundant** traffic is transmitted

Conclusion

- Mobile devices have a **limited** amount of memory
- iOS uses a **dynamic** buffer management method
  - Multiple HTTP requests
  - Redundant traffic
- Android uses a **static** buffer management method
  - One single HTTP request
  - No redundant traffic