TOSHIBA

Leading Innovation

Multicast Video-Streaming on Embedded Linux Environment

Daichi Fukui, Toshiba Corporation

Japan Technical Jamboree 63

Dec 1st, 2017

Background

- **Multicast networking**
  - Use single IP address to send packets to multiple clients
  - 224.0.0.0 to 239.255.255.255 for IPv4
- **Video streaming**
  - Play video via networking
  - No need to save a video on disks
- **Multicast streaming on Linux aimed for IoT**
  - Multicast networking + Video streaming
  - Useful for playing a video simultaneously on multiple clients
  - Demand for multicast-streaming videos in a closed network

Purpose

- **Research how to multicast-stream video on Linux**
  - Find software for multicast-streaming in an embedded board
  - Give appropriate parameters to streaming software
- **Build video-streaming system using Raspberry pi**
  - Multicast streaming of videos using Raspberry pi
  - Low-cost embedded board to stream/play videos
  - Reduced hardware development cost
- **Evaluate capability to stream/play video**
  - Video-streaming load on a server according to # of clients
  - Video-playing load on clients
    - Comparison of between PC/embedded board

Video streaming applications

S/W name	License	Architecture	Available on boards?	debian package	Language	Supported media format	User Interface (server)	User Interface (client)
icecast	GPLv2	x86, ARM	○	○	C, JS	(audio only)	Command line	Web browser
gstreamer	LGPL	x86, ARM	○	○	C	FLV, MP4, …	Command line	X server, Command line
VLC	LGPLv2.1	x86, ARM	○	○	C/C++	FLV, MP4, …	Command line, Qt	Qt
FFmpeg	LGPL2.1 + GPLv2+	x86, ARM	○	△	C	MP4, …	Command line	X server, command line
Mist Server	aGPLv3	x86, ARM, MIPS	○	×	C, JS	FLV, MP3, OGG	Web browser	Web browser
red5	Apache	x86, ARM	×	×	Java	FLV, MP4, …	Web browser, Flash	Web browser, Flash

Video streaming applications

S/W name	License	Architecture	Available on boards?	debian package	Language	Supported media format	User Interface (server)	User Interface (client)
icecast	GPLv2	x86, ARM	○	○	C, JS	(audio only)	Command line	Web browser
gstreamer	LGPL	x86, ARM	○	○	C	FLV, MP4, …	Command line	X server, Command line
VLC	LGPLv2.1	x86, ARM	○	○	C/C++	FLV, MP4, …	Command line, Qt	Qt
FFmpeg	LGPL2.1 + GPLv2+	x86, ARM	○	△	C	MP4, …	Command line	X server, command line
Mist Server	aG							eb owser
red5	Ap							eb owser, sh

FFmpeg/VLC are suitable because of …
- many projects using FFmpeg/VLC,
- adequate documentation,
- low-memory requirement,
- easier parameter setting

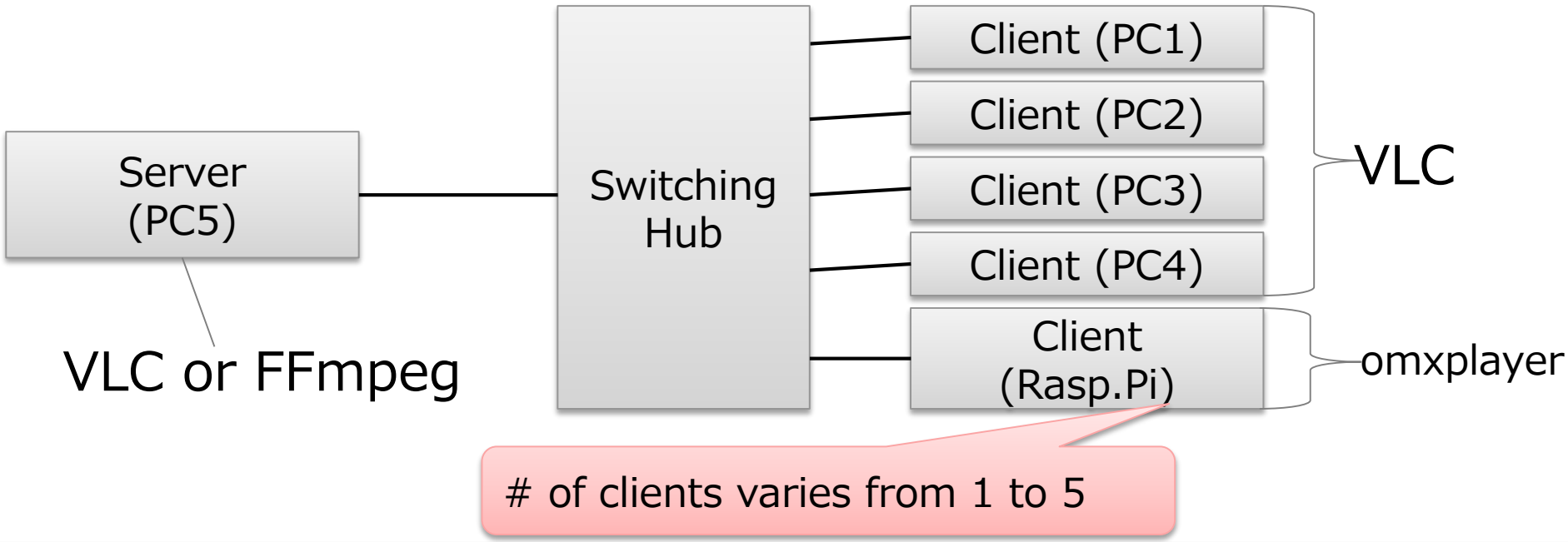
Evaluation environment

Purpose	Name	CPU	NIC	Mem.	Monitor	OS
Client	PC1	1.4GHz x2	1Gbps	4GB	1280x800	Ubuntu 16.04
	PC2	2.7GHz x2(4)	1Gbps	8GB	1366x768	Ubuntu 16.04
	PC3	1.2GHz x2	1Gbps	3GB	1280x800	Ubuntu 16.04
	PC4	2.7GHz x2(4)	1Gbps	4GB	1600x900	Ubuntu 14.04
	RaspberryPi V1 Type B+ (*)	0.7GHz x1	100Mbps	400MB	1280x800	Raspbian 8.0
Server	PC5	1.7GHz x2(4)	1Gbps	4GB	1366x768	Ubuntu 16.04

* Low-price and small single board computer produced by Raspberry Pi Foundation
Processor: ARMv6 (ARM1176JZF-S)
Price: 40USD

Multicast networking

- **224.0.0.0/4 as IPv4 multicast address**
- **Streaming software**
  - Server (send streaming): VLC, FFmpeg
  - Clients (receive streaming): VLC, omxplayer (RPi only)
- **Network**



omxplayer

- **Hardware-accelerated video/audio player**
- **Other players are not HW-accelerated by default**
  - So it is hard to play videos on RPi without omxplayer
  - Need to rebuild the players to exploit GPU power on RPi
- **Available as raspbian package**

Evaluation of network bandwidth

- **Details of evaluation environment**
  - S/W: iperf
  - Sender: send UDP packets to receiver
    - Bitrate increased gradually starting from 10Mbps
    - 10, 20, 30, ..., 100, 200, 400, 800, 1000 Mbps
  - Receiver: receive UDP packets sent from sender
  - Value: Network bandwidth of PC/RPi
- **Commands to obtain data**
  - sender

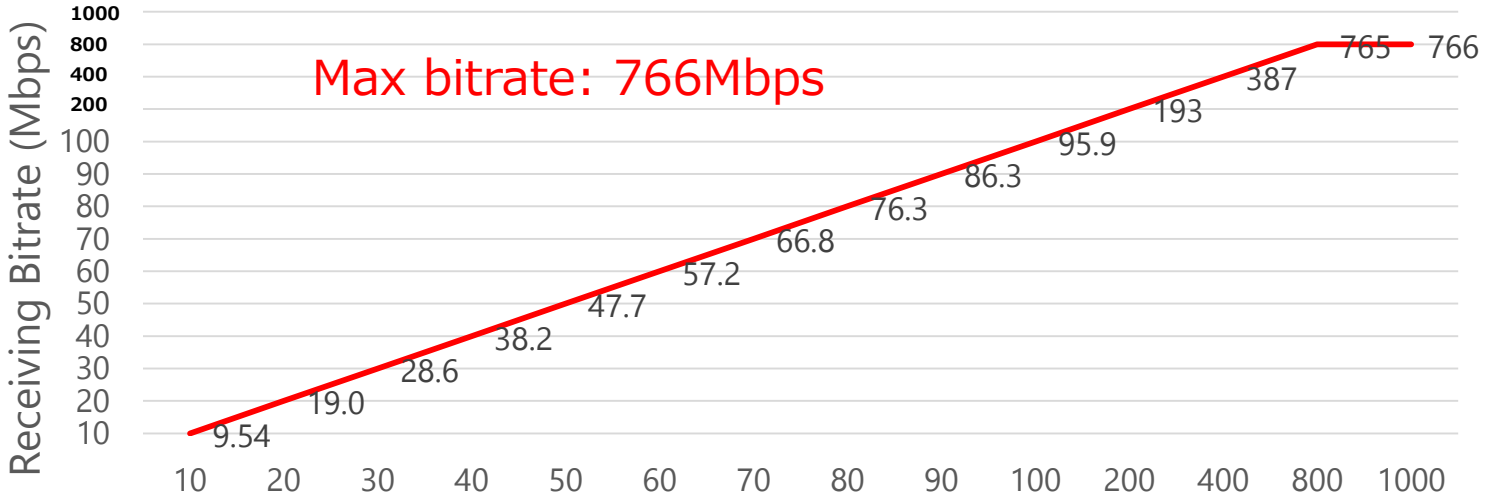
```
$ iperf -c ${ipaddr} -y C -u -f m -b ${size}M -i 1 -t 11
```

  - receiver

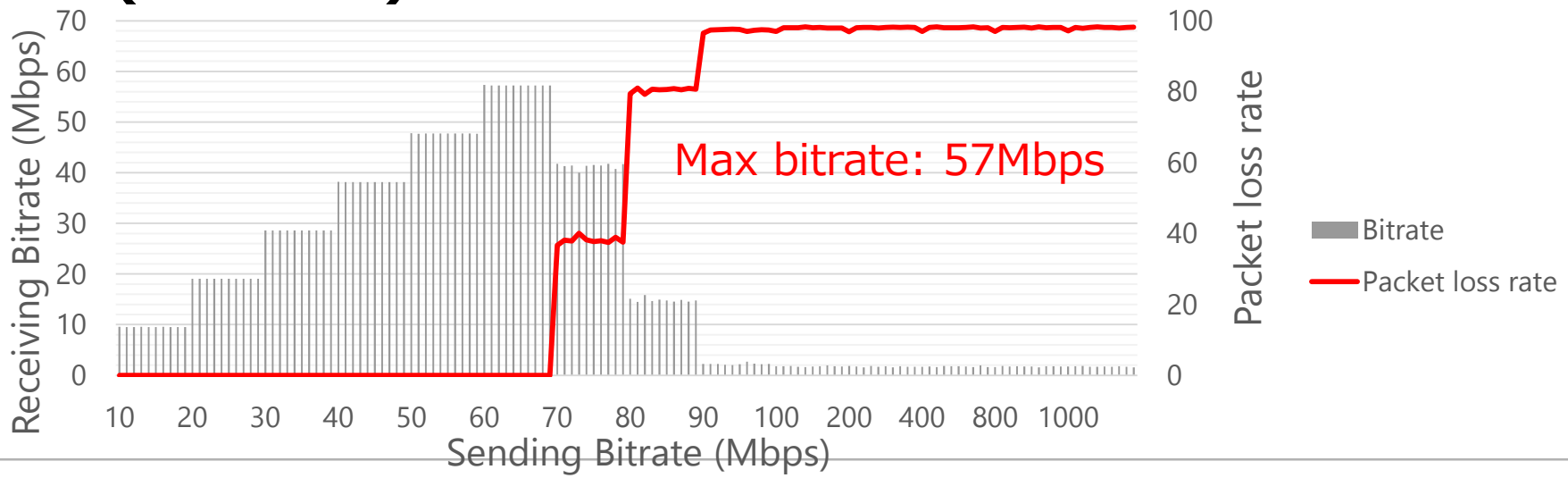
```
$ iperf -s -y C -u -f m -i 1 > logfile
```

Network bandwidth

- **PC (receiver) – Packet loss rate is almost zero**



- **RPi (receiver)**



Evaluation of video multicast-streaming

- **Details of evaluation environment**
  - Measuring duration: 120 seconds
  - Server: VLC/FFmpeg used to multicast-stream video
  - Client : VLC, omxplayer (RPi only) to play video
  - Value: CPU usage, network load
    - Calculate median for each data
- **Commands to obtain data**
  - CPU usage

```
$ top -d1 -n120 -b > logfile
```

  - Network load

```
$ timeout 120 dstat --noheader --net-packet --output logfile
```

Details of sample video

- **Underwater video**
  - Duration: 30 seconds
  - File size: 57.6 MB
  - Video bitrate: 15377 kbps
  - Resolution: 1920x1080 (played as full screen)
  - Frame rate: 30fps
  - Video codec: H264
  - Audio: None
  - Media format: MPEG4
  - Source: orangeHD.com

Parameter setting for streaming

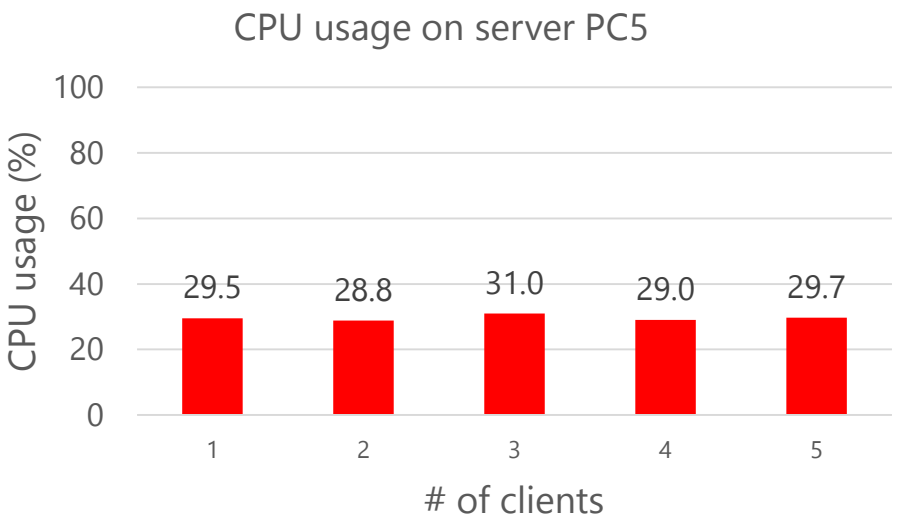
- **VLC**
  - Bitrate limited to 2M bps due to low VLC capability of streaming

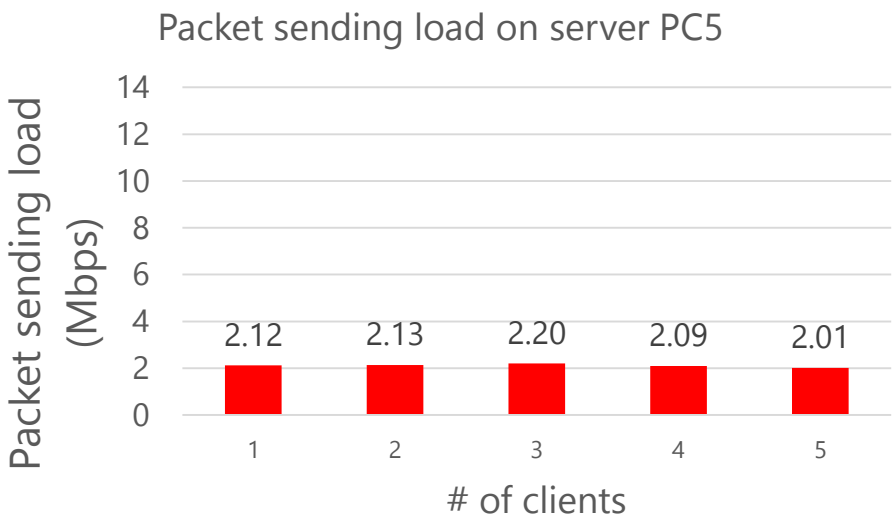
```
vlc -v underwater-video.mp4 ¥
--sout '#transcode{vcodec=h264,venc=x264'¥
'{scenecut=20,bframes=0},vb=2048,scale=1.0}'¥
':rtp{access=udp,mux=ts,dst=224.0.0.1,port=1234,'¥
'sap,name="water"}' ¥
:sout-all
```

- **FFmpeg**
  - Bitrate limited to 2M/8M bps due to low RPi capability of playing

```
ffmpeg -i "underwater-video.mp4" ¥
-preset ultrafast ¥
-vcodec libx264 ¥
-tune zerolatency ¥
-b 2048k (or 8192k)¥
-f mpegts udp://224.0.0.1:1234
```

Load on server/VLC (bitrate=2048k)

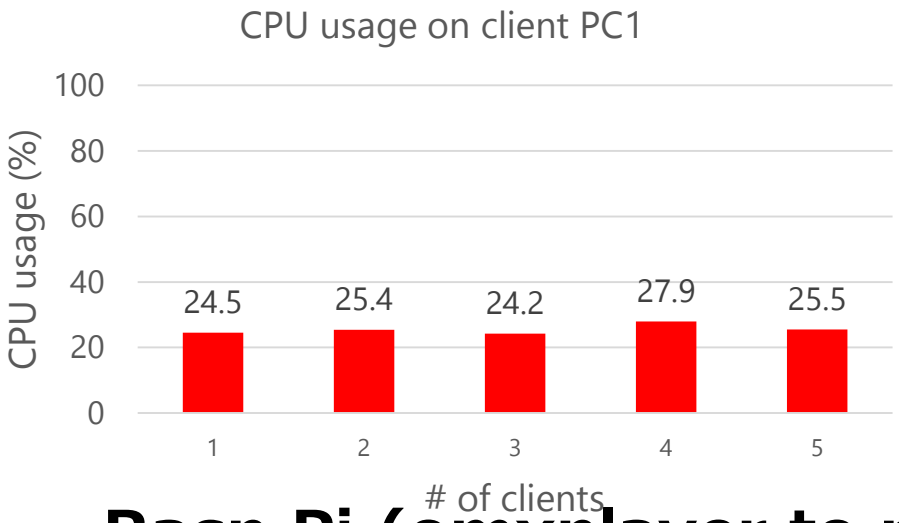


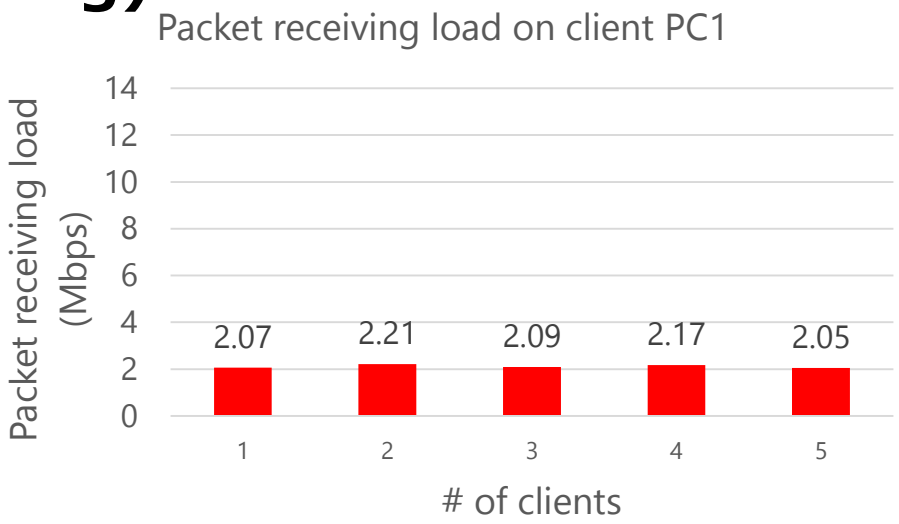


Constant load regardless of # of clients

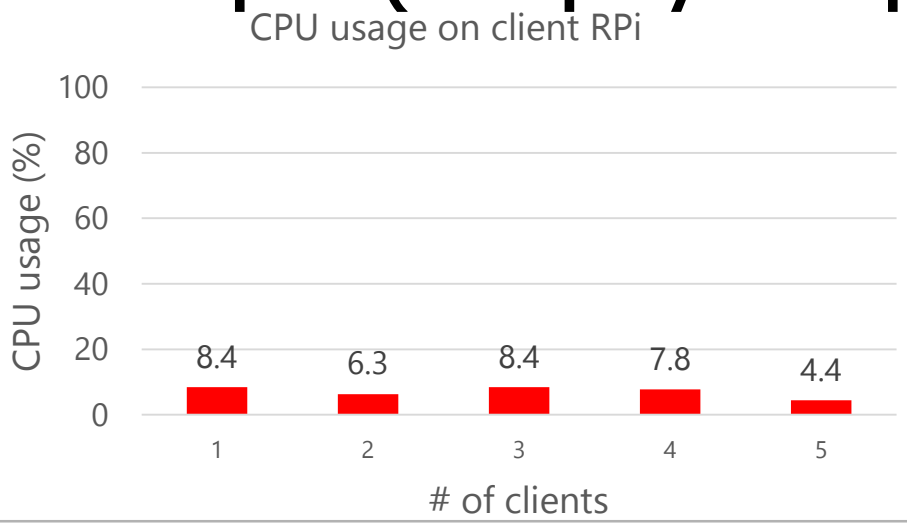
Load on clients (bitrate=2048k)

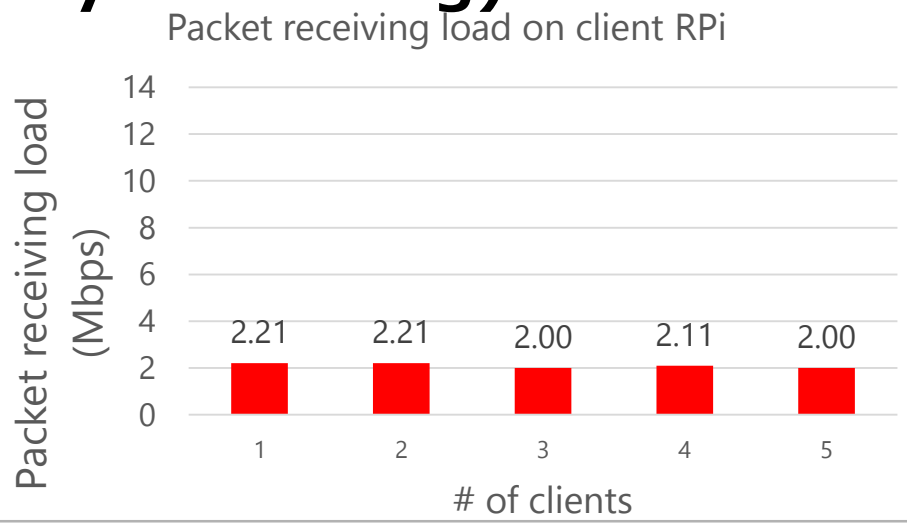
- **PC1 (VLC to play streaming)**





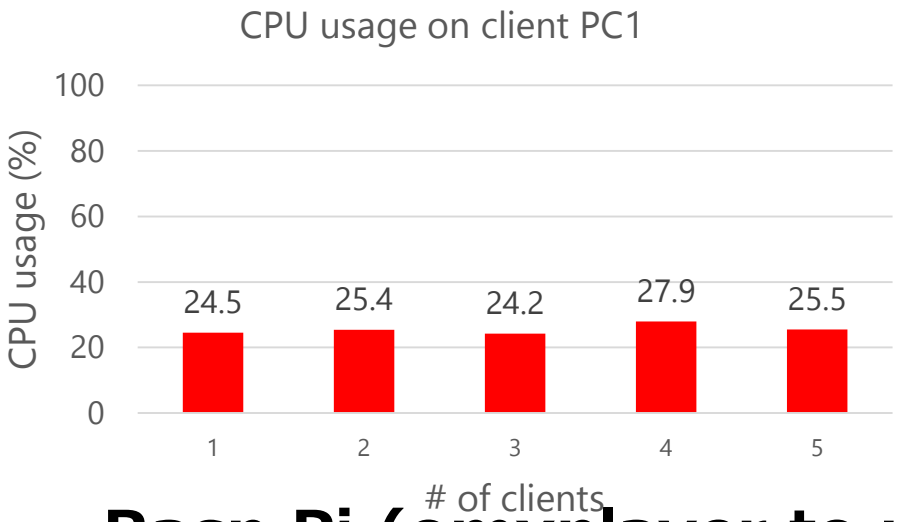
- **Rasp.Pi (omxplayer to play streaming)**

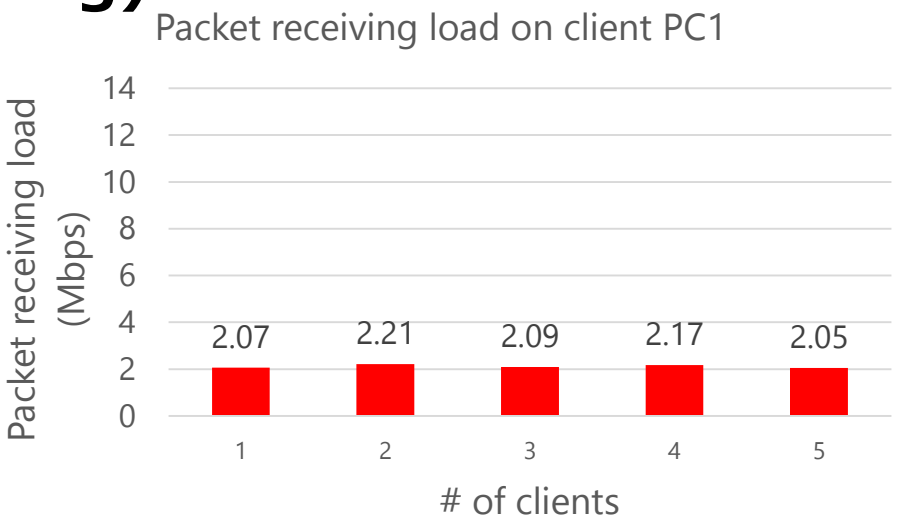




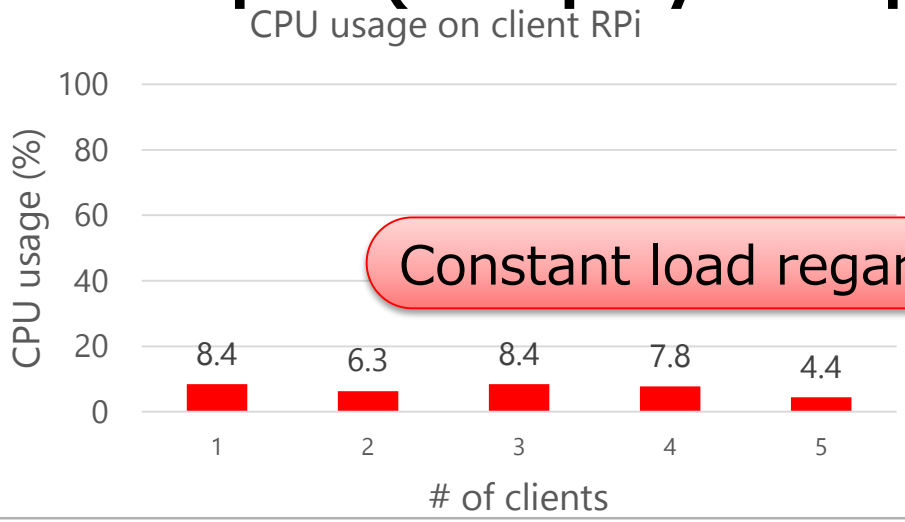
Load on clients (bitrate=2048k)

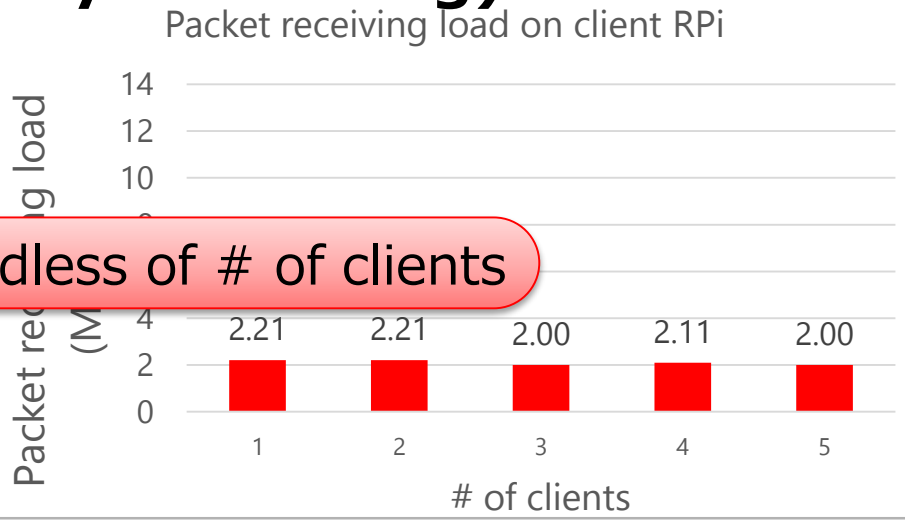
- **PC1 (VLC to play streaming)**





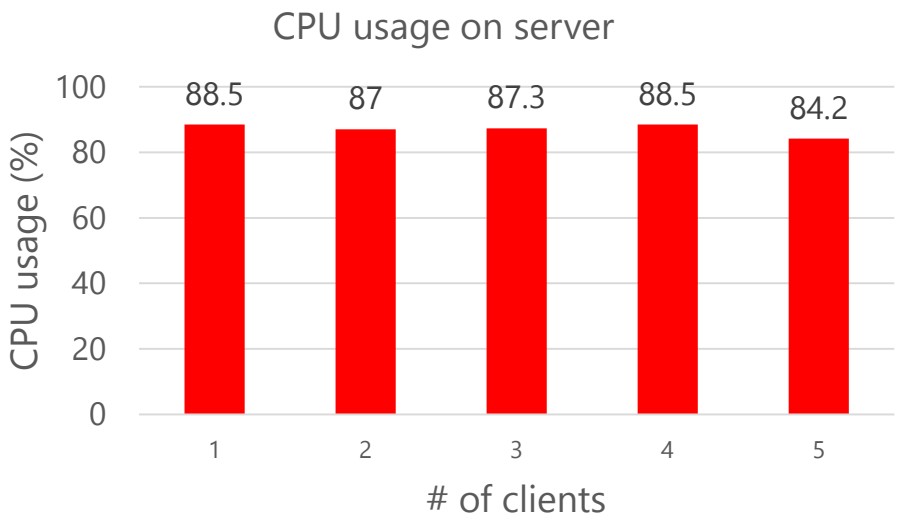
- **Rasp.Pi (omxplayer to play streaming)**

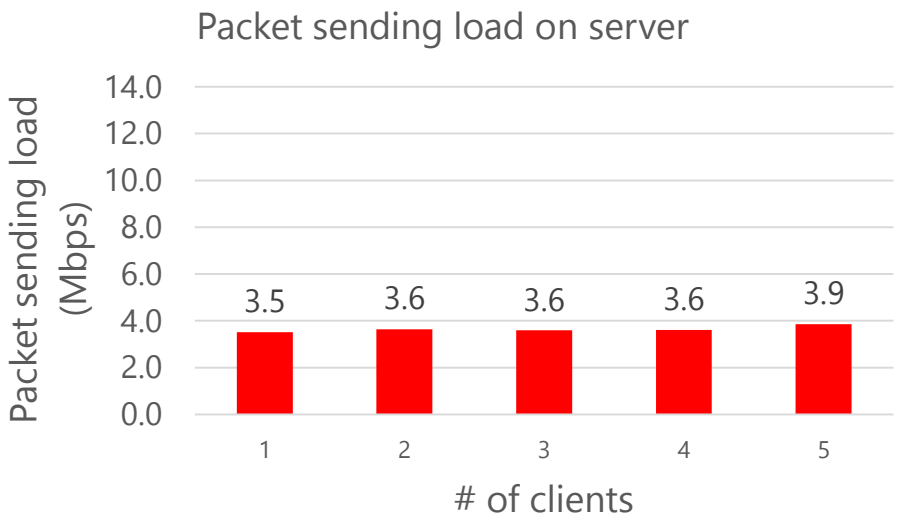




Constant load regardless of # of clients

Load on server/FFmpeg (bitrate=2048k)

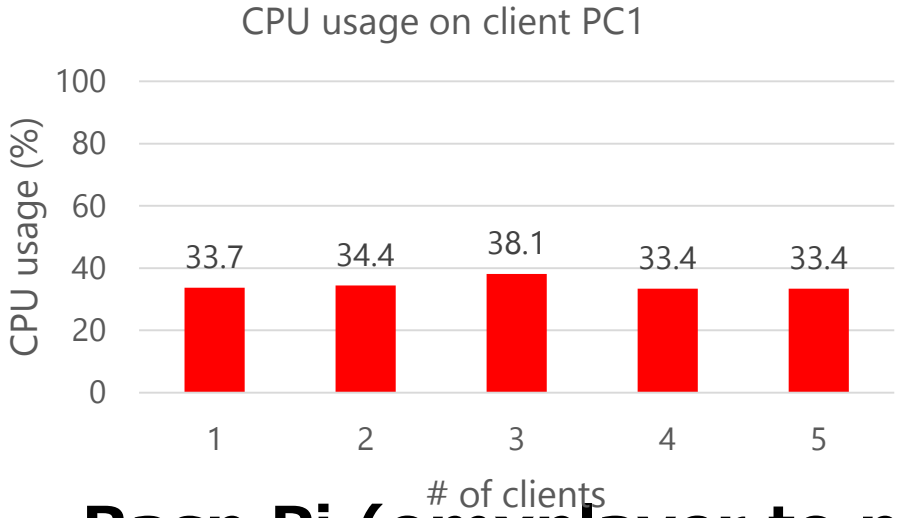


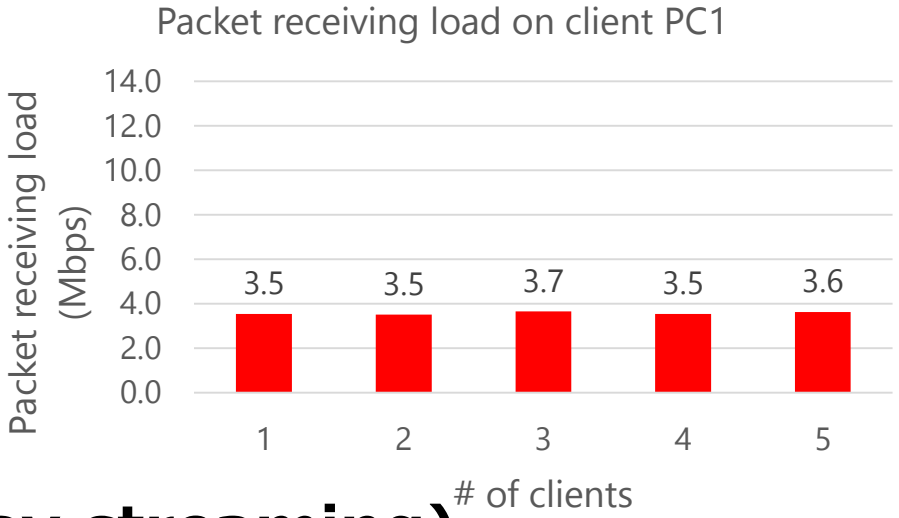


Constant load regardless of # of clients

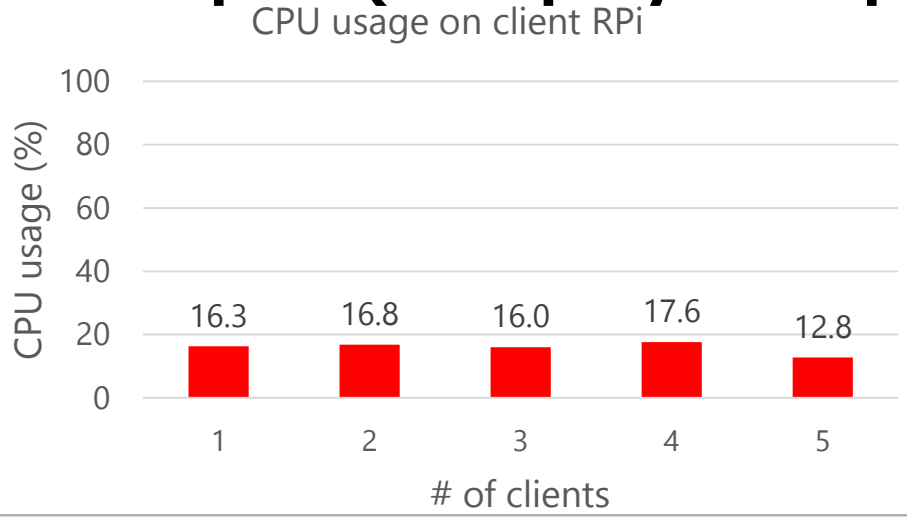
Load on client (bitrate=2048k)

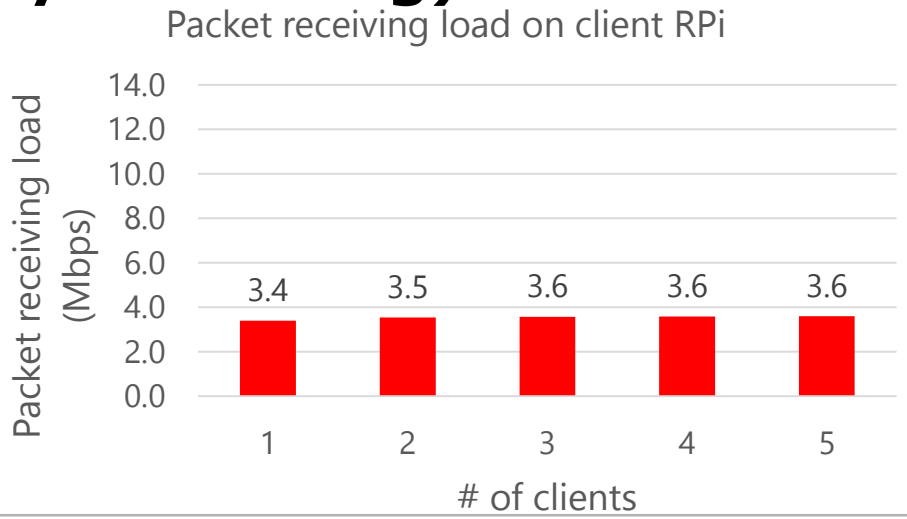
- **PC1 (VLC to play streaming)**





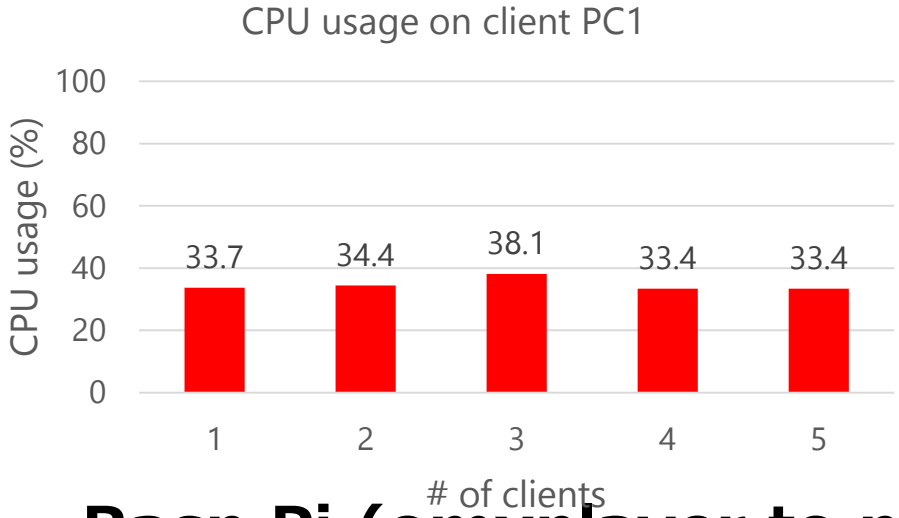
- **Rasp.Pi (omxplayer to play streaming)**

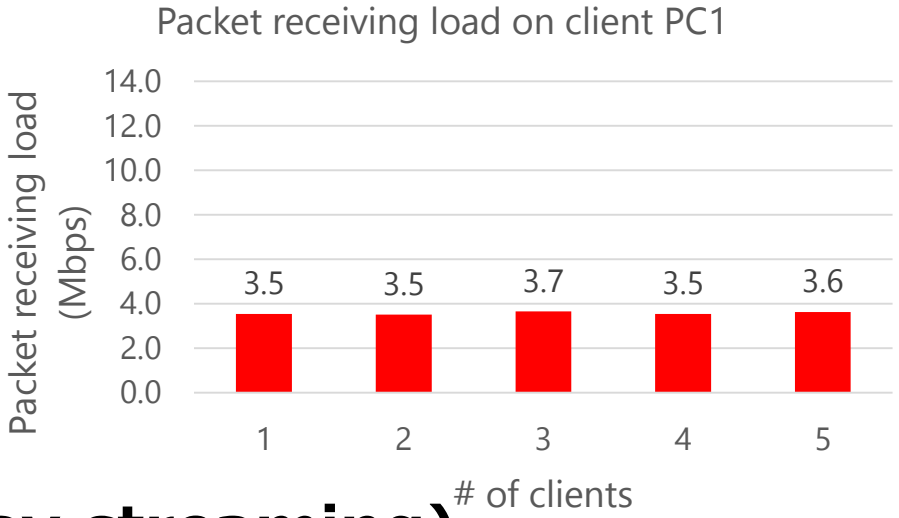




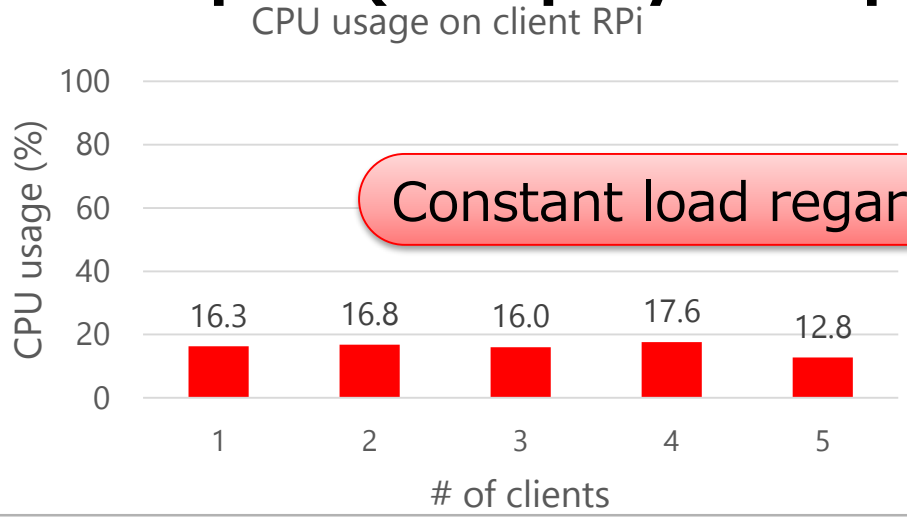
Load on client (bitrate=2048k)

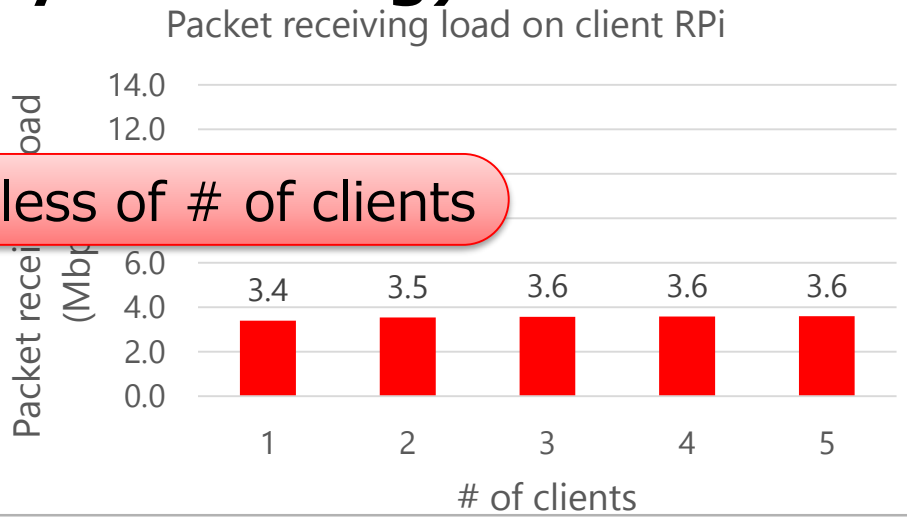
- **PC1 (VLC to play streaming)**





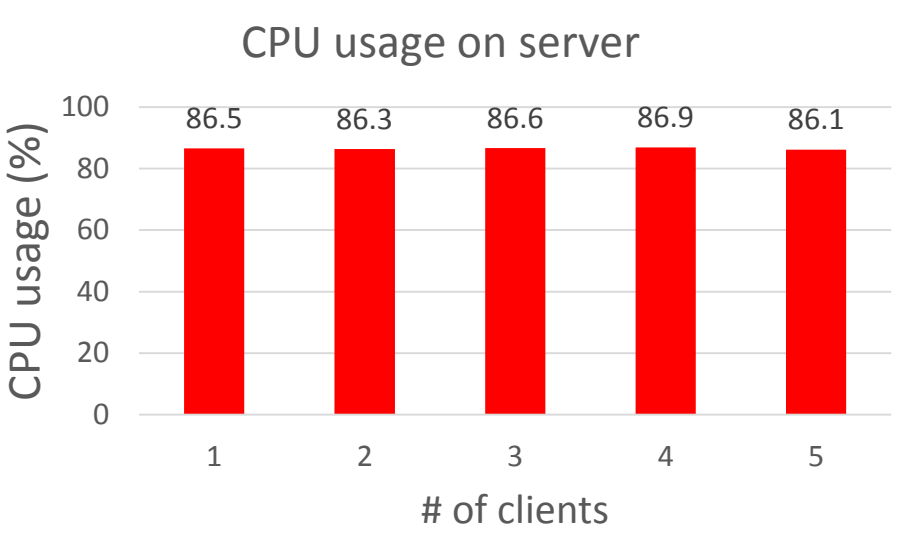
- **Rasp.Pi (omxplayer to play streaming)**

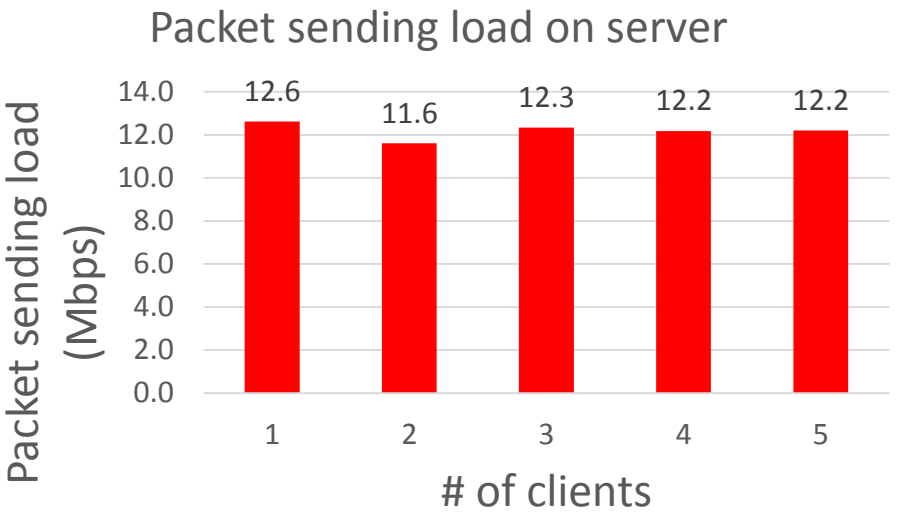




Constant load regardless of # of clients

Load on server/FFmpeg (bitrate=8192k)

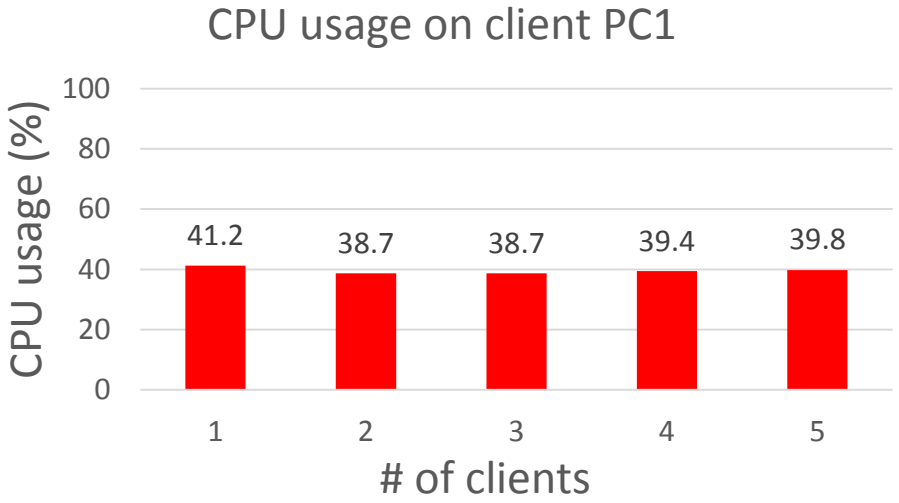


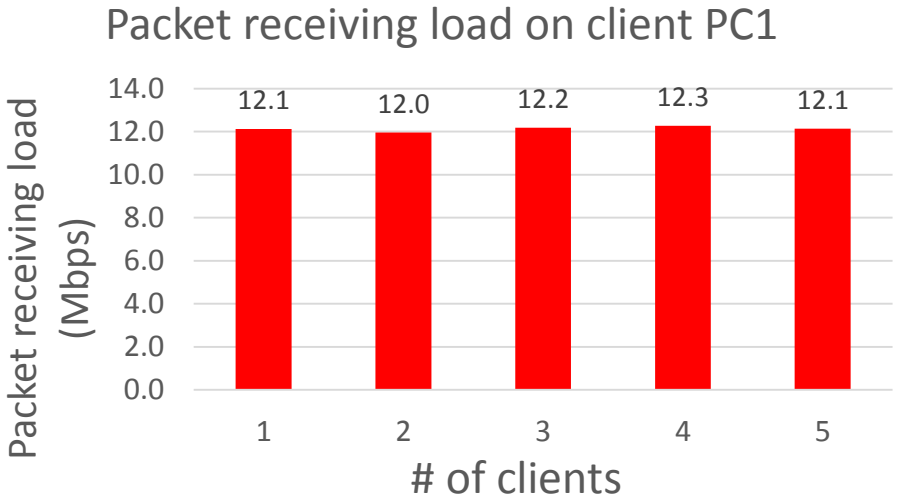


Constant load regardless of # of clients

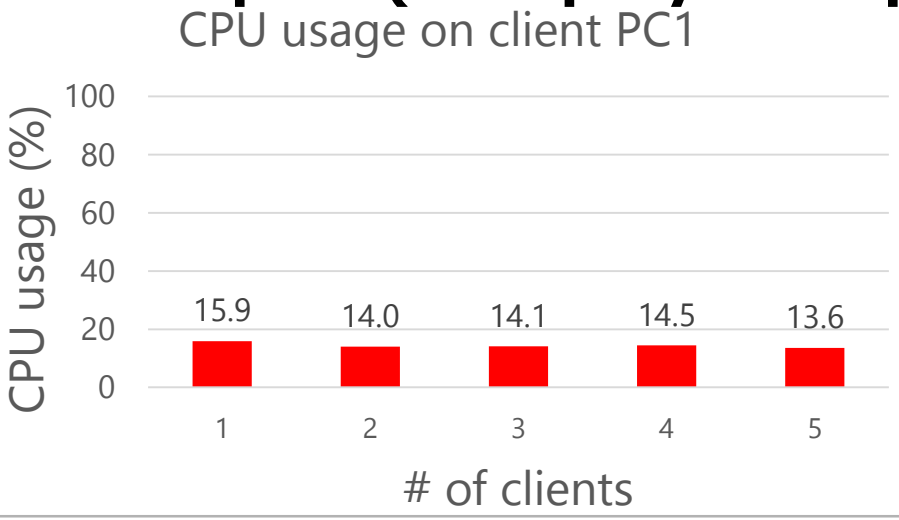
Load on clients (bitrate=8192k)

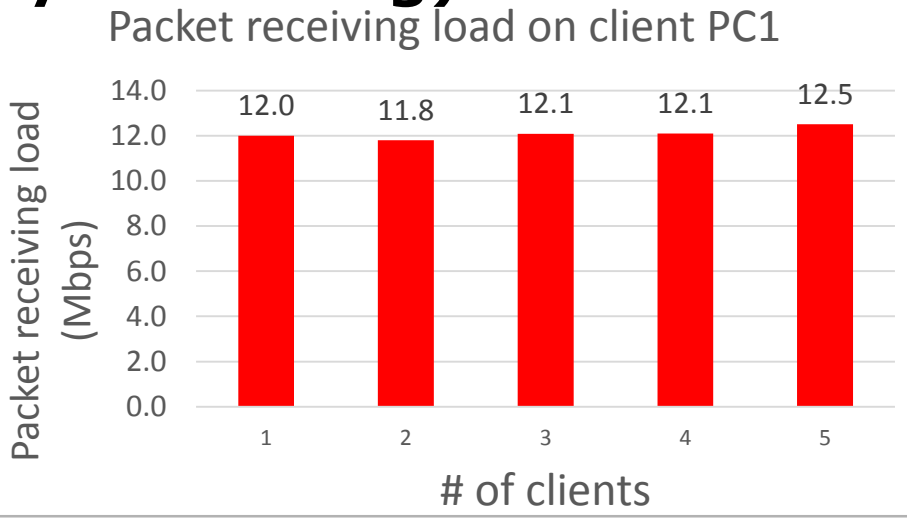
- **PC1 (VLC to play streaming)**



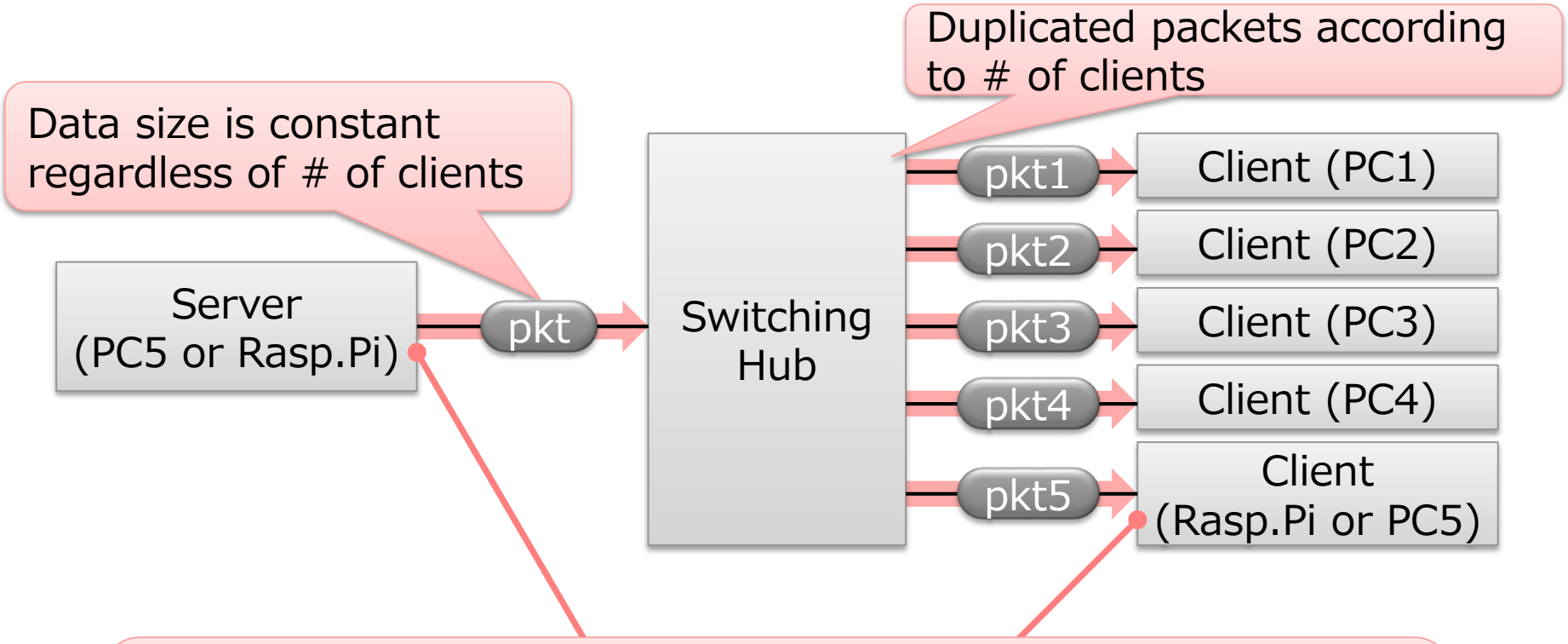


- **Rasp.Pi (omxplayer to play streaming)**





Duplicating packets on multicast network



No additional load for multicast thanks to packet duplication

Max streaming bitrate in different streaming S/W

- **VLC**
  - Skipping/lagging detected if the bitrate is more than **2048 kbps**
  - Quality of playing video is quite low; the image is coarse
- **FFmpeg**
  - PC1: No skipping/lagging even when bitrate = 15.4Mbps
  - RPi: Lagging detected if bitrate >= **8192 kbps**
    - Playing stops suddenly after few seconds if bitrate = 15.4Mbps

Max bitrate for streaming in different conditions (kbps)

S/W on server	Bitrate on PC	Bitrate on RPi
VLC	2048	2048
FFmpeg	15377	8192

Why skipping/lagging?

- **Narrow network bandwidth?**
  - Bandwidth of Raspberry pi = 57Mbps
  - Enough for streaming the video (bitrate = 15.4Mbps)
- **Small OS UDP buffer?**
  - Increased UDP buffer, but showed no improvement
- **PC/RPi's too low CPU/GPU power?**
  - The video played successfully on PC/RPi as a local file
  - "`vlc -v underwater-video.mp4`"
- **Difference between UDP and RTP?**
  - Changed btw. UDP and RTP, but showed no improvement
- **Missing parameters for VLC/FFmpeg on server?**
- **Missing parameters for VLC/omxplayer on client?**
  - Players on clients need more buffer

Conclusion

- **Studied how to multicast-stream on Linux**
  - FFmpeg or VLC to stream a video
- **Built Video-streaming system using Raspberry pi**
  - Multicast network for streaming
  - FFmpeg/VLC to stream a video
  - Parameters for multicast-streaming
  - Raspberry pi with omxplayer to play streaming
- **Evaluated capability to stream/play video**
  - Constant server/client load for streaming regardless of # of clients
  - Poor receiving capability in Raspberry pi
    - Lagging detected with FFmpeg if bitrate >= 8192kbps
  - Poor streaming capability of VLC on server
    - Skipping/Lagging detected with VLC if bitrate > 2048 kbps

Observation of multicast packets (Appx)

- **Duplicated ICMP packets in multicasting**
  - Multicast ping from PC5(server) to each client
  - PC5(server)

```
14:06:06.316506 IP 192.168.0.5 > 224.0.0.1: ICMP echo request, id 4045, seq 1, length 64
14:06:06.316722 IP 192.168.0.1 > 192.168.0.5: ICMP echo reply, id 4045, seq 1, length 64
14:06:06.316810 IP 192.168.0.3 > 192.168.0.5: ICMP echo reply, id 4045, seq 1, length 64
14:06:06.316821 IP 192.168.0.4 > 192.168.0.5: ICMP echo reply, id 4045, seq 1, length 64
14:06:06.316826 IP 192.168.0.2 > 192.168.0.5: ICMP echo reply, id 4045, seq 1, length 64
14:06:06.317017 IP 192.168.0.6 > 192.168.0.5: ICMP echo reply, id 4045, seq 1, length 64
```

  - PC1(client)

```
14:00:56.075160 IP 192.168.0.5 > 224.0.0.1: ICMP echo request, id 4045, seq 1, length 64
14:00:56.075194 IP 192.168.0.1 > 192.168.0.5: ICMP echo reply, id 4045, seq 1, length 64
```

  - PC2(client)

```
14:08:13.366629 IP 192.168.0.5 > 224.0.0.1: ICMP echo request, id 4045, seq 1, length 64
14:08:13.366680 IP 192.168.0.2 > 192.168.0.5: ICMP echo reply, id 4045, seq 1, length 64
```

  - PC3(client)

```
14:00:38.435625 IP 192.168.0.5 > 224.0.0.1: ICMP echo request, id 4045, seq 1, length 64
14:00:38.435711 IP 192.168.0.3 > 192.168.0.5: ICMP echo reply, id 4045, seq 1, length 64
```

  - PC4(client)

```
14:07:55.179511 IP 192.168.0.5 > 224.0.0.1: ICMP echo request, id 4045, seq 1, length 64
14:07:55.179570 IP 192.168.0.4 > 192.168.0.5: ICMP echo reply, id 4045, seq 1, length 64
```

  - Rasp. Pi(client)

```
14:30:05.724940 IP 192.168.0.5 > 224.0.0.1: ICMP echo request, id 4045, seq 1, length 64
14:30:05.725145 IP 192.168.0.6 > 192.168.0.5: ICMP echo reply, id 4045, seq 1, length 64
```

Observation of multicast packets (Appx)

- **Duplicated streaming packets in multicasting**
  - Multicast streaming from PC5(server) to each client
  - Captured by tcpdump
  - **PC5(server)**

18:46:29.993813 IP 192.168.0.5.40015 > 224.0.0.1.1234: UDP, length 1328
18:46:29.993863 IP 192.168.0.5.40015 > 224.0.0.1.1234: UDP, length 1328

  - **PC1(client)**

18:39:48.396436 IP 192.168.0.5.56971 > 224.0.0.1.1234: UDP, length 1328
18:39:48.396454 IP 192.168.0.5.56971 > 224.0.0.1.1234: UDP, length 1328

  - **PC2(client)**

18:49:02.429645 IP 192.168.0.5.40015 > 224.0.0.1.1234: UDP, length 1328
18:49:02.429676 IP 192.168.0.5.40015 > 224.0.0.1.1234: UDP, length 1328

  - **PC3(client)**

18:40:15.275972 IP 192.168.0.5.56971 > 224.0.0.1.1234: UDP, length 1328
18:40:15.276036 IP 192.168.0.5.56971 > 224.0.0.1.1234: UDP, length 1328

  - **PC4(client)**

18:47:09.161909 IP 192.168.0.5.56971 > 224.0.0.1.1234: UDP, length 1328
18:47:09.161952 IP 192.168.0.5.56971 > 224.0.0.1.1234: UDP, length 1328

  - **Rasp. Pi(client)**

19:10:07.008172 IP 192.168.0.5.40015 > 224.0.0.1.1234: UDP, length 1328
19:10:07.008177 IP 192.168.0.5.40015 > 224.0.0.1.1234: UDP, length 1328