آزمایشگاه سخت‌افزار

گزارش نهایی

نیمسال اول ۰۱-۰۲

موضوع پروژه:

ارتباط رایانه و بورد رزبری‌پای با ماژول‌های 4/4.5/5G و مقایسه آنها با هم

شماره گروه: **۴**

اعضای گروه:

عرشیا اخوان ۹۷۱۱۰۴۲۲

مهدی صادق شبیری ۹۷۱۱۰۱۴۴

غزل شناور ۹۷۱۰۱۸۹۷

۱ شرح پروژه

این پروژه، یک پروژه تحقیقاتی است که هدف اصلی آن، بررسی و مقایسه عملی میزان تاخیر بسته‌ها در بسترهای ارتباطی 3G و 4G است.

برای اجرای پروژه، از آردوینو و شیلد آردوینو SIM7000C استفاده کردیم. راهکار مورد استفاده ما، راهکار پیشنهادی دوم، یعنی ارسال از طریق برد به سرور و سپس دانلود اطلاعات از سرور به رایانه بود. متاسفانه بدلیل مشکلات این قطعه، پروژه در نهایت با استفاده از آردوینو وESP8266 انجام شد.

بررسی پروتکل‌ها در دو لایه انجام شد. در لایه چهارم، TCP، UDP و در لایه پنجم، پروتکلHTTP را بررسی کردیم.

۲ توضیح برخی اصطلاحات

۲,۱. پروتکل TCP

این پروتکل، یک پروتکل لایه چهارم (لایه انتقال) است. TCP اطلاعات را به ترتیب و با چک کردن برای خطا انتقال می‌دهد؛ از همین رو، پروتکل قابل اعتمادی است. این پروتکل، connection-oriented است و نیاز به برقراری ارتباط بین سرور و کلاینت توسط یک handshake سه مرحله‌ای دارد. این ویژگی پروتکل، امکان تشخیص و تصحیح خطا را فراهم می‌کند اما تاخیر را نسبت به روش UDP افزایش می‌دهد. از این پروتکل در email، انتقال فایل و بسیاری موارد دیگر استفاده می‌شود.

۲,۲. پروتکل UDP

این پروتکل، یک پروتکل لایه چهارم (لایه انتقال) است. UDP سرعت را به تصحیح خطا ترجیح می‌دهد. این پروتکل، نیازی به برقراری ارتباط اولیه میان سرور و کلاینت ندارد. این روش هیچ تضمینی درباره ترتیب داده و یا رسیدن آن به مقصد نمی‌دهد. این پروتکل در مواردی استفاده می‌شود که سرعت از تشخیص و تصحیح خطا مهمتر است و یا توسط بقیه اپلیکیشن‌ها انجام می‌شود.

۲,۳. پروتکل QUIC

این پروتکل، یک پروتکل لایه چهارم (لایه انتقال) است. QUIC با هدف بهبود عملکرد اپلیکیشن‌های connection-oriented طراحی شده است و این کار را با برقراری چند ارتباط بر پایه UDP انجام می‌دهد. هدف دیگر این پروتکل کاهش تاخیر است. این پروتکل در 2021 توسط IETF استاندار شد. این پروتکل توسط Chrome، Edge، Firefox، و Safari پشتیبانی می‌شود.

۲,۴. پروتکل HTTP

این پروتکل، یک پروتکل لایه پنجم (لایه اپلیکیشن) است. HTTP پایه داده‌ها در world wide web است. این روش، از مدل درخواست-پاسخ پیروی می‌کند.

۲,۵. دستورهای AT

دستورهای AT مجموعه دستوراتی است برای کنترل کردن ارتباط با مودم (یا دکل). علت این نامگذاری این است که با دستورات AT قرار است attention مودم گرفته شود.

این دستورات ۴ نوع دارند:

۲,۵,۱. دستورات تست Test commands

این دستورات برای بررسی پشتیبانی مودم از یک دستور است.

فرمت دستور به این شکل است:

AT<command name>=?

برای مثال

ATD=?

۲,۵,۲. دستورات خواندن Read command

این دستورات برای گرفتن تنظیمات گوشی یا مودم است

فرمت دستور به این شکل است:

AT<command name>?

برای مثال

AT+CBC?

۲,۵,۳. دستورات ست کردن Set commands

این دستورات برای مقدار دهی تنظیمات گوشی یا مودم است.

فرمت دستور به این شکل است:

AT<command name>=value1, value2, …, valueN

برای مثال

AT+CSCA=”+9876543210”, 120

۲,۵,۴. دستورات اجرا Execution commands

این دستورات برای اجرای یک عملیات است.

فرمت دستور به این شکل است:

AT<command name>=parameter1, parameter2, …, parameterN

برای مثال

AT+CMSS=1,”+ 9876543210”, 120

در جدول پایین تعدادی از این دستورات آمده است:

Call control :

Command	Description
ATA	Answer command
ATD	Dial command
ATH	Hang up call
ATL	Monitor speaker loudness
ATM	Monitor speaker mode
ATO	Go on-line
ATP	Set pulse dial as default
ATT	Set tone dial as default
AT+CSTA	Select type of address
AT+CRC	Cellular result codes

Data card Control :

Command	Description
ATI	Identification
ATS	Select an S-register
ATZ	Recall stored profile
AT&F	Restore factory settings
AT&V	View active configuration
AT&W	Store parameters in given profile
AT&Y	Select Set as power up option
AT+CLCK	Facility lock command
AT+COLP	Connected line identification presentation
AT+GCAP	Request complete capabilities list
AT+GMI	Request manufacturer identification
AT+GMM	Request model identification
AT+GMR	Request revision identification
AT+GSN	Request product serial number identification (IMEI)

Phone control :

Command	Description
AT+CBC	Battery charge
AT+CGMI	Request manufacturer identification
AT+CGMM	Request model identification
AT+CGMR	Request revision identification
AT+CGSN	Request product serial number identification
AT+CMEE	Report mobile equipment error
AT+CPAS	Phone activity status
AT+CPBF	Find phone book entries
AT+CPBR	Read phone book entry
AT+CPBS	Select phone book memory storage
AT+CPBW	Write phone book entry
AT+CSCS	Select TE character set
AT+CSQ	Signal quality

Computer data interface :

Command	Description
ATE	Command Echo
ATQ	Result code suppression
ATV	Define response format
ATX	Response range selection
AT&C	Define DCD usage
AT&D	Define DTR usage
AT&K	Select flow control
AT&Q	Define communications mode option
AT&S	Define DSR option
AT+ICF	DTE-DCE character framing
AT+IFC	DTE-DCE Local flow control
AT+IPR	Fixed DTE rate

Service :

Command	Description
AT+CLIP	Calling line identification presentation
AT+CR	Service reporting control
AT+DR	Data compression reporting
AT+ILRR	DTE-DCE local rate reporting

Network Communication parameter :

Command	Description
ATB	Communications standard option
AT+CBST	Select bearer service type
AT+CEER	Extended error report
AT+CRLP	Radio link protocol
AT+DS	Data compression

Miscellaneous :

Command	Description
A/	Re-execute command line
AT?	Command help
AT*C	Start SMS interpreter
AT*T	Enter SMS block mode protocol
AT*V	Activate V.25bis mode
AT*NOKIATEST	Test command
AT+CESP	Enter SMS block mode protocol

SMS Text mode :

Command	Description
AT+CSMS	Select message service
AT+CPMS	Preferred message storage
AT+CMGF	Message format
AT+CSCA	Service centre address
AT+CSMP	Set text mode parameters
AT+CSDH	Show text mode parameters
AT+CSCB	Select cell broadcast message types
AT+CSAS	Save settings
AT+CRES	Restore settings
AT+CNMI	New message indications to TE
AT+CMGL	List messages
AT+CMGR	Read message
AT+CMGS	Send message
AT+CMSS	Send message from storage
AT+CMGW	Write message to memory
AT+CMGD	Delete message

SMS PDU mode :

Command	Description
AT+CMGL	List Messages
AT+CMGR	Read message
AT+CMGS	Send message
AT+CMGW	Write message to memory

Testing :

Command	Description
AT	Checking communication between the module and computer.

۲.۶. نرخ خرابی

خرابی[1] هنگامی رخ می‌دهد که بسته‌های داده در یک شبکه کامپیوتری به مقصد خود نمیرسند. این اتفاق به دلیل مشکلات موجود در سیستم، و یا ترافیک[2] رخ می‌دهد. نرخ خرابی[3] درصد بسته‌های از دست رفته به کل بسته‌های ارسالی است. نرخ خرابی تاثیر قابل توجهی بر تجربه کاربری دارد به نحوی که ۵ تا ۱۰ درصد نرخ خرابی به طرز قابل توجهی این تجربه را تحت‌الشعاع قرار می‌دهد.

[1] Packet loss
[2] Congestion
[3] Drop rate

۲,۷. پهنای باند

پهنای باند به حداکثر نرخ انتقال داده بر روی یک مسیر اطلاق می‌شود. این ویژگی معمولا بر اساس تعداد bit های منتقل شده بر ثانیه اندازه گرفته می‌شود.

۲,۸. تاخیر[۴]

تاخیر در شبکه، به زمانی اطلاق می‌شود که بسته در مسیر عبور از فرستنده به گیرنده قرار دارد. تاخیر نیز تاثیر قابل توجهی بر تجربه کاربری دارد.

۳ لیست آزمایش‌ها

آزمایش‌های ما شامل ۴ مورد می‌شوند:

- بررسی پروتکل TCP
- بررسی پروتکل UDP
- بررسی QUIC بر پایه UDP
- بررسی HTTP بر پایه TCP

برای هر یک از چهار آزمایش، سه متغیر نرخ خرابی، تاخیر و پهنای باند با انجام چند تست و سپس میانگین‌گیری میان نتایج محاسبه می‌شوند.

در این مرحله، تصمیم ما بر این شده است که از packet generator ها استفاده نکنیم؛ چرا که هیچ‌کدام از آنها هر چهار آزمایش ما را پوشش نمی‌دهند و استفاده از چند ابزار مختلف می‌تواند باعث ایجاد ناهماهنگی در نتایج آزمایش شود.

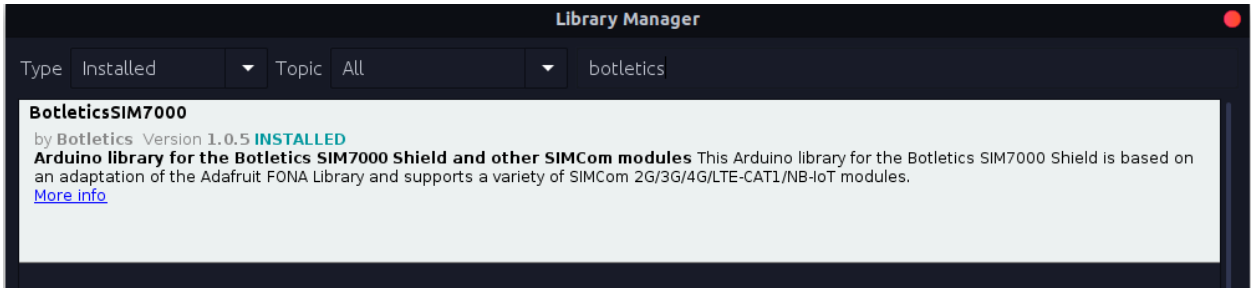
۴ آزمایش با Sim7000

۴,۱. کتابخانه‌های مورد استفاده

۴,۱,۱. کتابخانه‌ی Botletics-SIM7000

برای تست کردن این پروژه از کتاب‌خانه‌ی Botletics-SIM7000 استفاده شده است. (لینک به گیت‌هاب) این کتاب‌خانه بر اساس کتاب‌خانه‌ی Adafruit FONA طراحی شده است و از ماژول‌های SIM5320، SIM7000، SIM7070، و SIM7500 پشتیبانی می‌کند.

[4] latency



شکل ۱: کتابخانه استفاده شده برای اتصال به اینترنت

۴,۲. تست کارکرد قطعه

برای تست کردن ماژول از مثال SIM7XXX_LTE_Demo از کتاب‌خانه‌ی گفته شده استفاده شده است.
برای بارگذاری این کد باید از مسیر File <- Examples <- BotleticsSIM7000 <- SIM7XXX_LTE_Demo آن را انتخاب کرد.
سپس باید define های TX و RX را روی ۸ و ۷ قرار داد.

```
#define TX 8 // Microcontroller RX
#define RX 7 // Microcontroller TX
```

با توجه به اینکه ماژول ما SIM7000 است باید این مدل را انتخاب کرد و ۴ define دیگر را کامنت کرد.

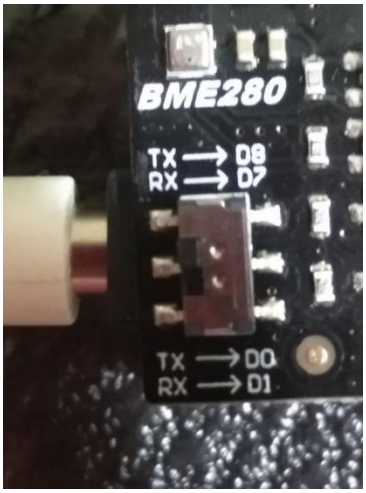
```
// Define *one* of the following lines:
#define SIMCOM_7000
// #define SIMCOM_7070
// #define SIMCOM_7500
//#define SIMCOM_7600
```

در نهایت باید APN مناسب با سیمکارت را انتخاب کرد

```
// Configure a GPRS APN, username, and password.
// You might need to do this to access your network's GPRS/data
// network.  Contact your provider for the exact APN, username,
// and password values.  Username and password are optional and
// can be removed, but APN is required.
//modem.setNetworkSettings(F("your APN"), F("your username"), F("your password"));
//modem.setNetworkSettings(F("m2m.com.attz")); // For AT&T IoT SIM card
//modem.setNetworkSettings(F("telstra.internet")); // For Telstra (Australia) SIM card - CAT-M1 (Band 28)

modem.setNetworkSettings(F("RighTel")); // For Hologram SIM card
// modem.setNetworkSettings(F("mcinet")); // For Hologram SIM card
// modem.setNetworkSettings(F("mtnirancell")); // For Hologram SIM card

// Optionally configure HTTP gets to follow redirects over SSL.
// Default is not to follow SSL redirects, however if you uncomment
// the following line then redirects over SSL will be followed.
//modem.setHTTPSRedirect(true);
```

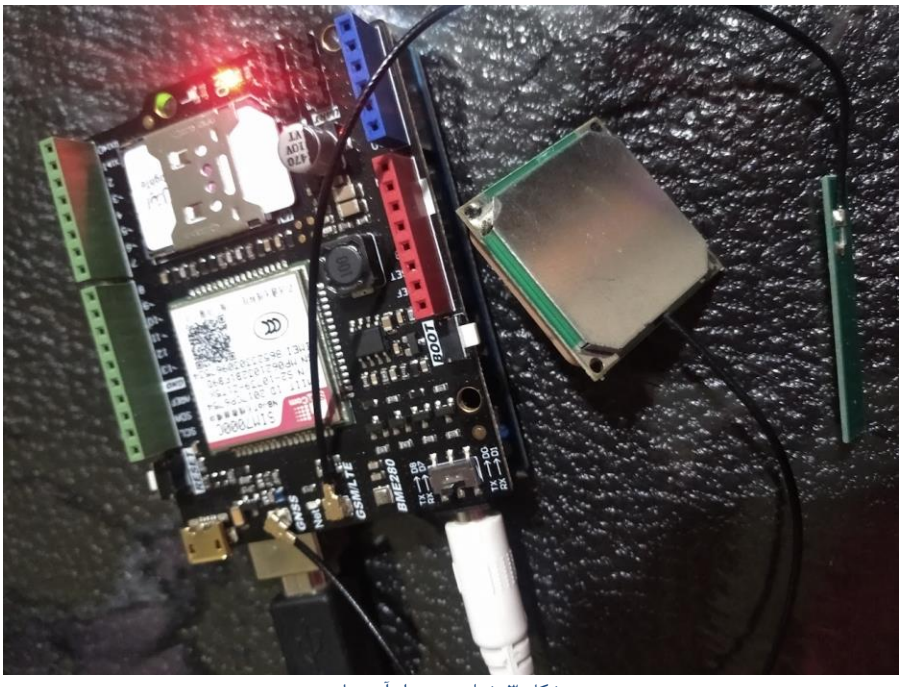
که برای رایتل RighTel، همراه اول mcinet ،و ایرانسل mtnirancell است.

چون TX و RX را پایه ۸ و ۷ قرار دادیم سوئیچ پاور را باید روی ۷ و ۸ تنظیم کنیم. مانند شکل زیر:



شکل ۲: تنظیمات سوییچ پاور

شیلد SIM7000 شامل یک آنتن برای اینترنت و یک آنتن برای GPS، یک منبع تغذیه (سیم سفید) ،و پورت سریال (سیم سیاه) است.



شکل ۳: شیلد به همراه آنتن‌ها

بعد از اجرای کد ابتدا به مودم (دکل) وصل می‌شود که لاگ‌های آن در زیر مشاهده می‌شود.

```
17:54:07.925 -> modem> SIM7XXX Demo
17:54:23.231 -> Configuring to 9600 baud
17:54:24.195 -> Attempting to open comm with ATs
17:54:24.261 ->         ---> AT
17:54:24.261 ->         <--- OK
17:54:24.294 ->         ---> ATE0
17:54:24.328 ->         <--- OK
17:54:24.460 ->         ---> ATE0
17:54:24.460 ->         <--- OK
17:54:24.626 ->         ---> AT+GMR
17:54:25.158 ->         <--- Revision:1351B06SIM7000C
17:54:25.191 ->
17:54:25.191 -> OK
17:54:25.191 ->
17:54:25.191 ->         ---> AT+CPMS="SM","SM","SM"
17:54:25.257 ->         <--- +CPMS: 15,15,15,15,15,15
17:54:25.291 -> Modem is OK
17:54:25.291 -> Found SIM7000
17:54:25.324 ->         ---> AT+GSN
17:54:25.324 ->         <--- 865233030967534
17:54:25.357 -> Module IMEI: 865233030967534
17:54:25.390 ->         ---> AT+CFUN=1
17:54:25.390 ->         <--- OK
17:54:25.457 ->         ---> AT+CGDCONT=1,"IP","mtnirancell"
17:54:25.490 ->         <--- OK
17:54:25.523 -> ------------------------------------
```

سپس منوی زیر مشاهده خواهد شد.

```
17:54:25.523 -> ------------------------------------
17:54:25.556 -> [?] Print this menu
17:54:25.589 -> [a] Read the ADC, 0V-VBAT for SIM7000
17:54:25.623 -> [b] Read supply voltage
17:54:25.656 -> [C] Read the SIM CCID
17:54:25.656 -> [U] Unlock SIM with PIN code
17:54:25.689 -> [i] Read signal strength (RSSI)
17:54:25.722 -> [n] Get network status
```

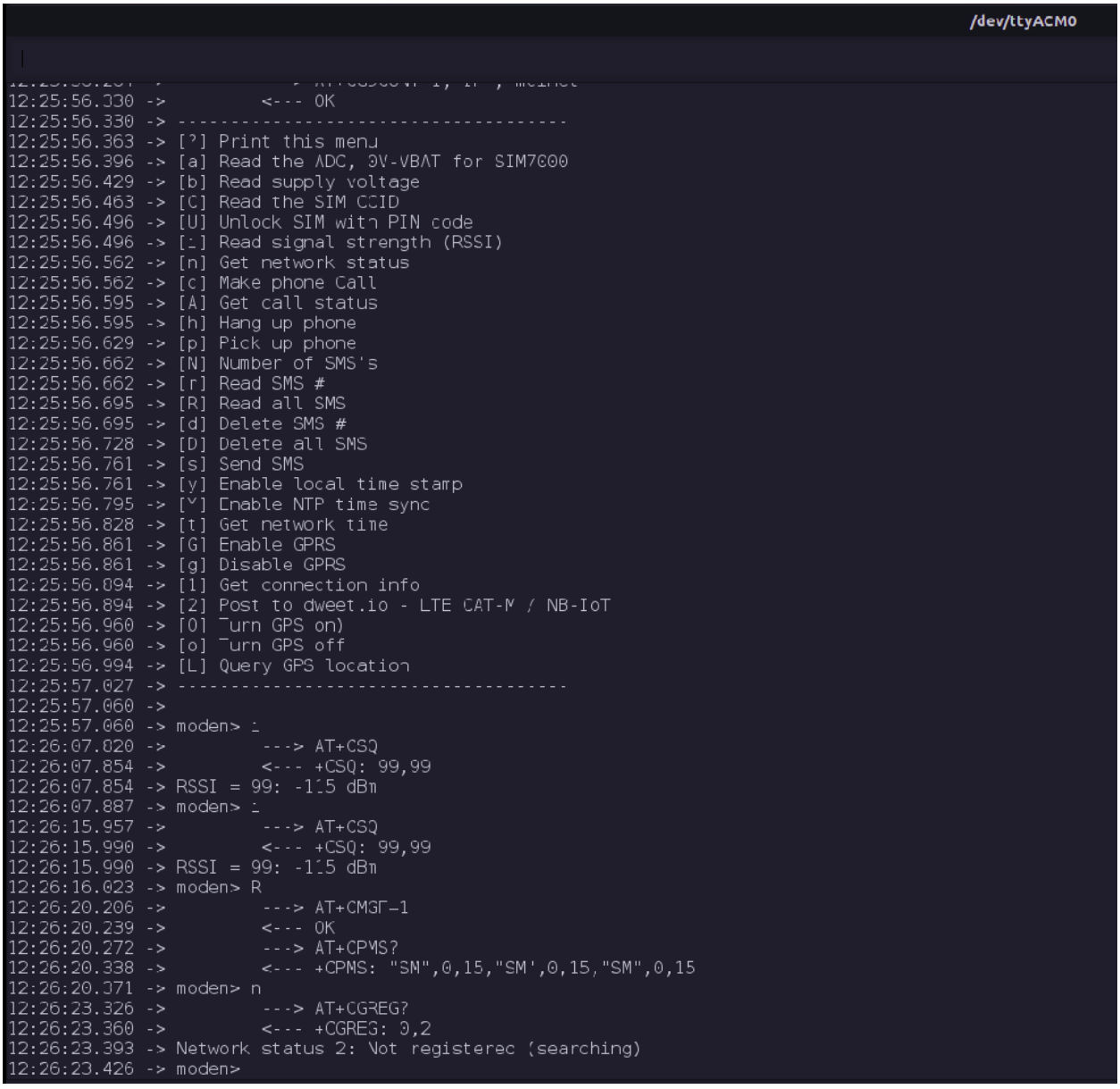
```
17:54:25.755 -> [c] Make phone Call
17:54:25.789 -> [A] Get call status
17:54:25.789 -> [h] Hang up phone
17:54:25.822 -> [p] Pick up phone
17:54:25.855 -> [N] Number of SMS's
17:54:25.855 -> [r] Read SMS #
17:54:25.888 -> [R] Read all SMS
17:54:25.888 -> [d] Delete SMS #
17:54:25.921 -> [D] Delete all SMS
17:54:25.955 -> [s] Send SMS
17:54:25.955 -> [y] Enable local time stamp
17:54:25.988 -> [Y] Enable NTP time sync
17:54:26.021 -> [t] Get network time
17:54:26.021 -> [G] Enable GPRS
17:54:26.054 -> [g] Disable GPRS
17:54:26.054 -> [1] Get connection info
17:54:26.087 -> [2] Post to dweet.io - LTE CAT-M / NB-IoT
17:54:26.154 -> [O] Turn GPS on)
17:54:26.154 -> [o] Turn GPS off
17:54:26.187 -> [L] Query GPS location
17:54:26.187 -> ------------------------------------
17:54:26.253 ->
17:54:26.253 -> modem>
17:54:37.642 -> +CPIN: NOT READY
```

در صورتی که n یعنی Get network status را صدا بزنیم خروجی زیر را می‌دهد.

```
17:55:47.357 -> modem> n
17:55:47.855 ->          ---> AT+CGREG?
17:55:47.888 ->          <--- +CGREG: 0,0
17:55:47.921 -> Network status 0: Not registered
17:55:47.955 -> modem>
```

با وجود تلاش‌های بسیار تیم، و نظر به محدودیت‌های اعمال شده بر روی سامانه مخابرات کشور، پروژه‌ی این گروه با شکست مواجه شد. طبق بررسی‌های انجام شده توسط تیم توسعه پروژه، متاسفانه امکان اجرای طرح تحقیقاتی ارائه شده وجود ندارد.

به طور دقیقتر ماژول Sim7000 در ایران به درستی کار نمی‌کند و حتی خطاهای درستی نیز نمی‌دهد که بشود آن‌ها را رفع کرد. موقعی که سیمکارت داخل ماژول قرار می‌گیرد در شبکه Register نمی‌شود و این باعث می‌شود امکان استفاده از آن وجود نداشته باشد.



تصویر بالا از اجرای کد hardware.ino به دست آمده است. همانطور که دیده می‌شود وضعیت شبکه سیم کارت در حالت Not Registered می‌ماند.

برای همین به جای استفاده از ماژول sim7000 از ماژول ESP8266 استفاده کردیم.

۵ آزمایش ESP8266

۵,۱. سرور

قسمت سرور کاملا زده شده و الان از TCP, UDP, Quic, و HTTP پشتیبانی می‌کند و برای هر کدام Upload, Download, و Latency محاسبه می‌شود.

۵,۱,۱. TCP

کدهای مورد نظر در فایل tcp.py هستند.

```
def start(port: int) -> socket.socket:
    """
    Starts a tcp socket on given port

    Args:
        port: socket port
    """
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind(('0.0.0.0', port))  # Bind to 0.0.0.0:port
        s.listen()  # Waits for new client to connect
        conn, addr = s.accept()
        logger.info("New connecting %s", addr)
        return conn
```

تابع start یک سوکت از نوع TCP می‌سازد و در ورودی port مربوط به سوکت را می‌گیرد.

```python
def start_throughput(port: int, period: int, upload: bool, packet_size: int) -> None:
    """
    Start a tcp socket server to test throughput

    Args:
        port: socket port
        period: time of test in seconds
        upload: test upload or download
        packet_size: size of packets to send
    """
    conn = start(port)
    logger.info("Start tcp throughput with period: %d upload: %s", period, upload)
    now = datetime.datetime.now()  # test start time
    bytes_cnt = [0] * period  # Result of number of bytes sent or received per seconds
    while True:
        total = (datetime.datetime.now() - now).total_seconds()
        if total >= period:
            break
        if upload:
            # Receive `packet_size` bytes of data
            data = conn.recv(packet_size)
            bytes_cnt[int(total)] += len(data)
        else:
            # Sends `packet_size` bytes of data
            conn.sendall(b'*' * packet_size)
            bytes_cnt[int(total)] += packet_size
    logger.info("Result: %s", bytes_cnt)
    conn.close()  # Close connection
```

تابع start_throughput برای راه اندازی سرور upload, download است و در ورودی با period مشخص می‌شود که در چند ثانیه تست را انجام دهد و با بولین upload مشخص می‌شود هدف سنجش آپلود است یا دانلود.

```python
def start_latency(port: int, number_of_packets: int) -> None:
    """
    Starts latency test

    Args:
        port: socket port
        number_of_packets: number of packets to test latency
    """
    conn = start(port)
    logger.info("Start tcp latency with %d packets", number_of_packets)
    result = []  # Result of latency tests
    for i in range(number_of_packets):
        # Sends 1 byte of data and wait for 1 byte of data to receive
        now = datetime.datetime.now()
        conn.sendall(b'$')
        conn.recv(1)
        result.append(int((datetime.datetime.now() - now).total_seconds() * 1_000))
    conn.close()  # Close connection
    logger.info("Result: %s", result)
```

تابع start_latency برای شروع سرور محاسبه تاخیر است. در ورودی number_of_packets برابر تعداد پکتی است که می‌فرستد تا latency را حساب کند. این بخش به این صورت عمل می‌کند که سرور یک پکت به کلاینت می‌فرستد و کلاینت لحظه‌ای که پکت را دریافت کرد یک پکت به سرور می‌فرستد.

UDP ۵،۱،۲

با توجه به اینکه کد در زبان پایتون زده شده و سوکت UDP, TCP شبیه هم هستند کدهای مربوط به قسمت UDP نیز شباهت زیادی با TCP دارند.

```python
def start(port: int) -> Tuple[socket.socket, Tuple[str, int]]:
    """
    Starts a udp socket on given port

    Args:
        port: socket port
    """
    logger.info("Waiting for udp socket")
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s.bind(('0.0.0.0', port))  # Bind to 0.0.0.0:port
    # Waits for new client to connect and send 1 byte of data
    while True:
        data, addr = s.recvfrom(1)
        if data is not None:
            break
    logger.info("Found udp socket")
    return s, (addr[0], 10210)
```

```python
def start_throughput(port: int, period: int, upload: bool, packet_size: int) -> None:
    """
    Start an udp socket server to test throughput

    Args:
        port: socket port
        period: time of test in seconds
        upload: test upload or download
        packet_size: size of packets
    """

    conn, addr = start(port)
    logger.info("Start udp throughput with period: %d, addr: %s", period, addr)
    now = datetime.datetime.now()  # test start time
    bytes_cnt = [0] * period  # Result of number of bytes sent or received per seconds
    while True:
        total = (datetime.datetime.now() - now).total_seconds()
        if total >= period:
            break
        if upload:
            # Receive `packet_size` bytes of data
            data, _ = conn.recvfrom(packet_size)
            bytes_cnt[int(total)] += len(data)
        else:
            # Sends `packet_size` bytes of data
            conn.sendto(b'*' * packet_size, addr)
            bytes_cnt[int(total)] += packet_size
    logger.info("Result: %s", bytes_cnt)
    conn.close()  # Close connection
```

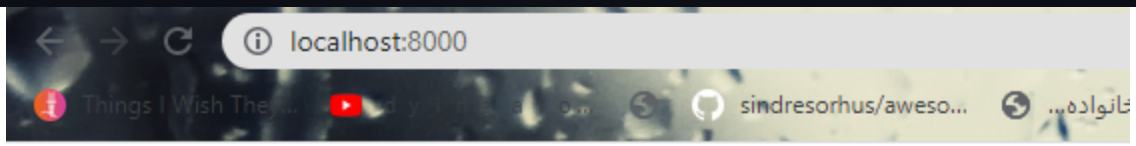
```python
def start_latency(port: int, number_of_packets: int) -> List[int]:
    """
    Starts latency test

    Args:
        port: socket port
        number_of_packets: number of packets to test latency
    """
    conn, addr = start(port)
    logger.info("Start udp latency with %d packets", number_of_packets)
    result = []  # Result of latency tests
    for i in range(number_of_packets):
        # Sends 1 byte of data and wait for 1 byte of data to receive
        now = datetime.datetime.now()
        conn.sendto(b'$', addr)
        conn.recv(1)
        result.append(int((datetime.datetime.now() - now).total_seconds() * 1_000))
    conn.close()  # Close connection
    logger.info("Result: %s", result)
```

به طور مشابه ۳ بخش TCP هر بخش کار گفته شده را انجام می‌دهد.

۵،۱،۳ HTTP

در این بخش با استفاده از لایبری uploadserver یک سرور http راه اندازی می‌کنیم. همچنین یک فایل حجیم برای تست Download و یک فایل خالی برای تست Latency می‌سازیم.

```
# We should remove arguments for uploadserver.main to work
sys.argv = sys.argv[:1]
sys.argv.append(str(PORT))
# Creating a huge file to test download
os.system(f'dd if=/dev/zero of={FILE_NAME} bs={FILE_SIZE}MB count=1')
# Creating an empty file to test latency
os.system(f'touch {EMPTY_FILE_NAME}')

# Clean dummy files when programm recived SIGINT
def sigint_handler(_, __):
    logger.info("Cleaning dummy file")
    os.system(f'rm -f {FILE_NAME} {EMPTY_FILE_NAME}')
    sys.exit(0)
signal.signal(signal.SIGINT, sigint_handler)

# Server http server
uploadserver.main()
```



Directory listing for /

- 3D Objects/
- ansel/
- AppData/
- Application Data/
- Contacts/
- Cookies/
- Desktop/
- Documents/
- Downloads/
- Favorites/

تصویر بالا از http server است.

۵,۲. توضیحات کد سخت افزار

برای این آزمایش نیاز به کتاب‌خانه‌های ESP8266HTTPClient, ESP8266WiFi و WiFiUdp داریم که هر ۳ کتاب‌خانه built-in هستند.

۵,۲,۱. تابع setup_wifi

این تابع برای اتصال به access point است. کد این تابع در صفحه بعد قابل دسترسی است.

```
/// Connect to wifi
bool setup_wifi() {
  Serial.printf("connecting to %s\n", WIFI_SSID);
  WiFi.begin(WIFI_SSID, WIFI_PASS);

  // Wait for the Wi-Fi to connect
  int tries_left = WIFI_CONN_TRY_COUNT;
  while (WiFi.status() != WL_CONNECTED && tries_left) {
    delay(WIFI_CONN_CHK_INTERVAL);
    Serial.printf("retrying [%d]\n", WIFI_CONN_TRY_COUNT - tries_left);
    tries_left--;
  }
  // Didn't make the connection.
  if (!tries_left) {
    Serial.println("Failed to connect");
    WiFi.disconnect();
    return false;
  }

  // Configure wifi connection.
  WiFi.mode(WIFI_STA);
  WiFi.setAutoReconnect(true);
  WiFi.persistent(true);

  Serial.println("Connection established!");
  Serial.print("IP address:\t");
  Serial.println(WiFi.localIP());
  return true;
}
```

۵،۲،۲. تابع setup

آردوینو هنگام اجرا تابع setup را اجرا می‌کند. سریال برای ورودی و خروجی را روی ۹۶۰۰ تنظیم می‌کنیم و LED را در صورتی که با موفقیت به Access point وصل شد روشن می‌کنیم.

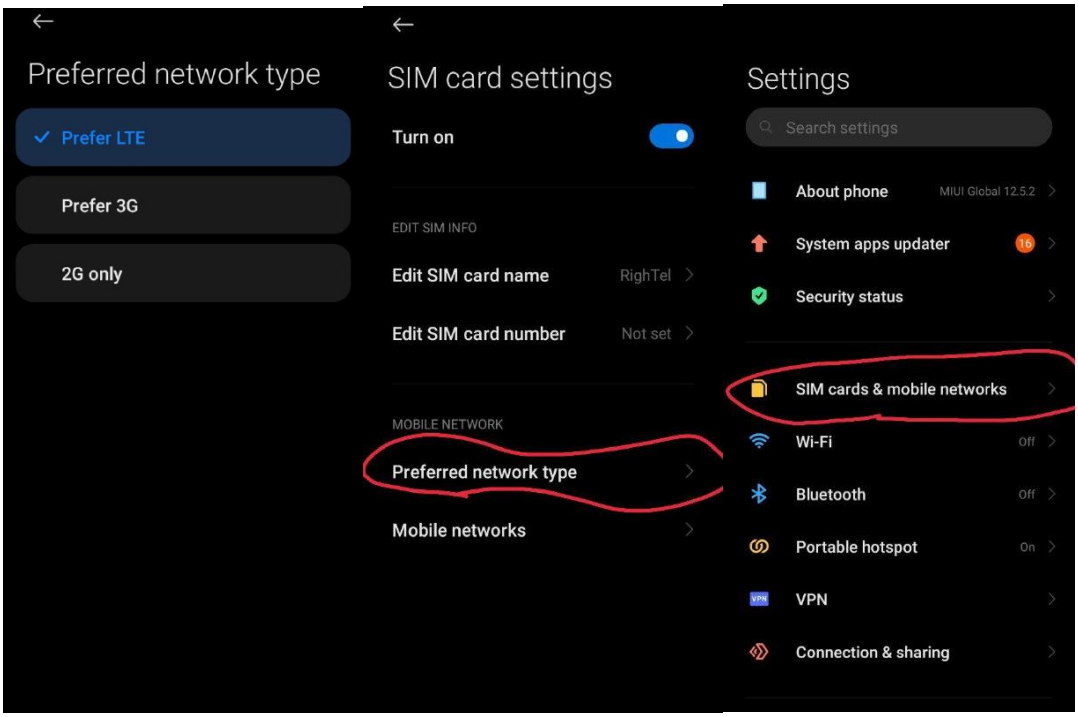
```
void setup() {
  // Make esp work Faster.
  ESP.eraseConfig();

  // Starting Serial input for user comminiucation.
  Serial.begin(9600);
  // Waits for Serial to begin.
  delay(10);

  // Turn the LED off.
  digitalWrite(BUILT_IN_LED, HIGH);

  // Fill upload_buffer with random data.
  generate_upload_data(upload_buffer, BUFF_SIZE);

  if (setup_wifi()) {
    Serial.println("Wifi connection is setup!");
    // Turn the LED on.
    // Show that the System is up and running.
    digitalWrite(BUILT_IN_LED, LOW);
  }
}
```

۵,۲,۳. توضیحات آزمایش

با استفاده از ماژول ESP8266 به hotspot وصل می‌شویم و سپس تست‌های لازم را انجام می‌دهیم.

۵,۲,۴. هات اسپات

برای مشخص کردن نوع اینترنت در گوشی اندروید باید به قسمت mobile network & SIM cardsd رفت سپس از قسمت Preferred network type نوع اینترنت را مشخص می‌کنیم.

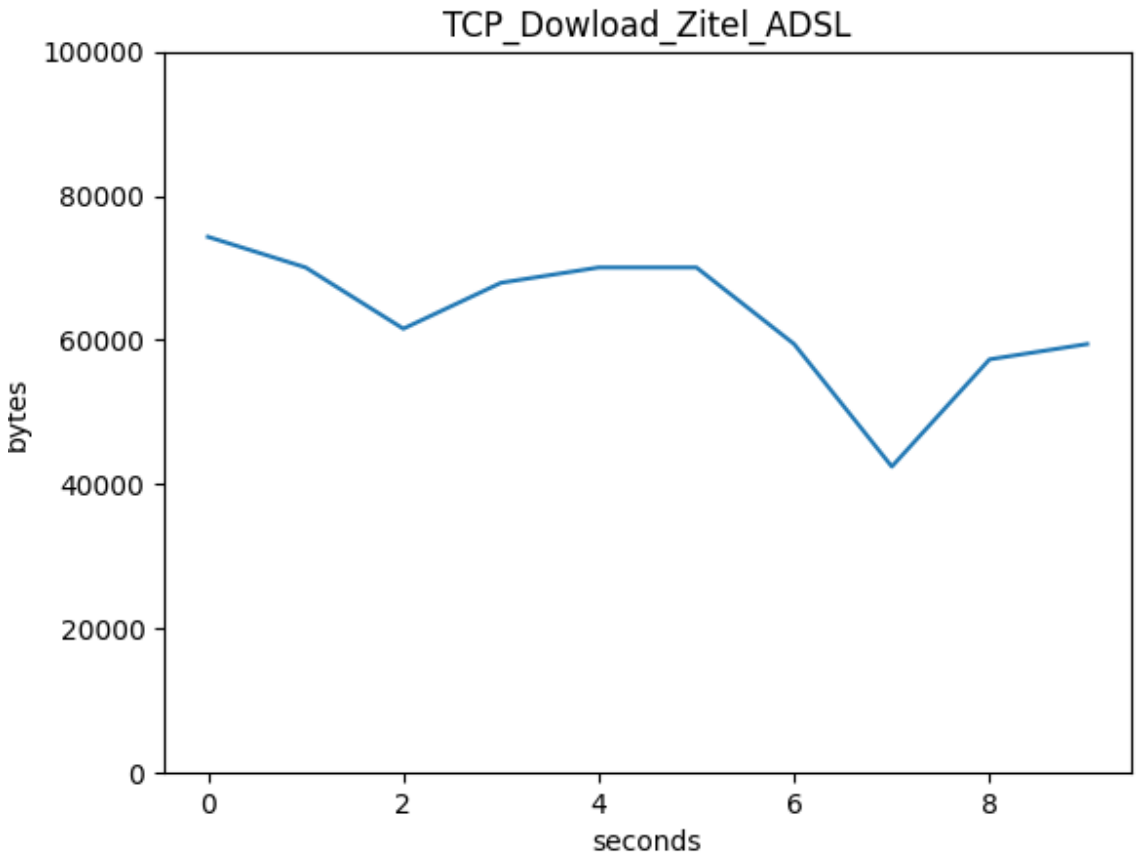


۶ نتایج و نمودارها

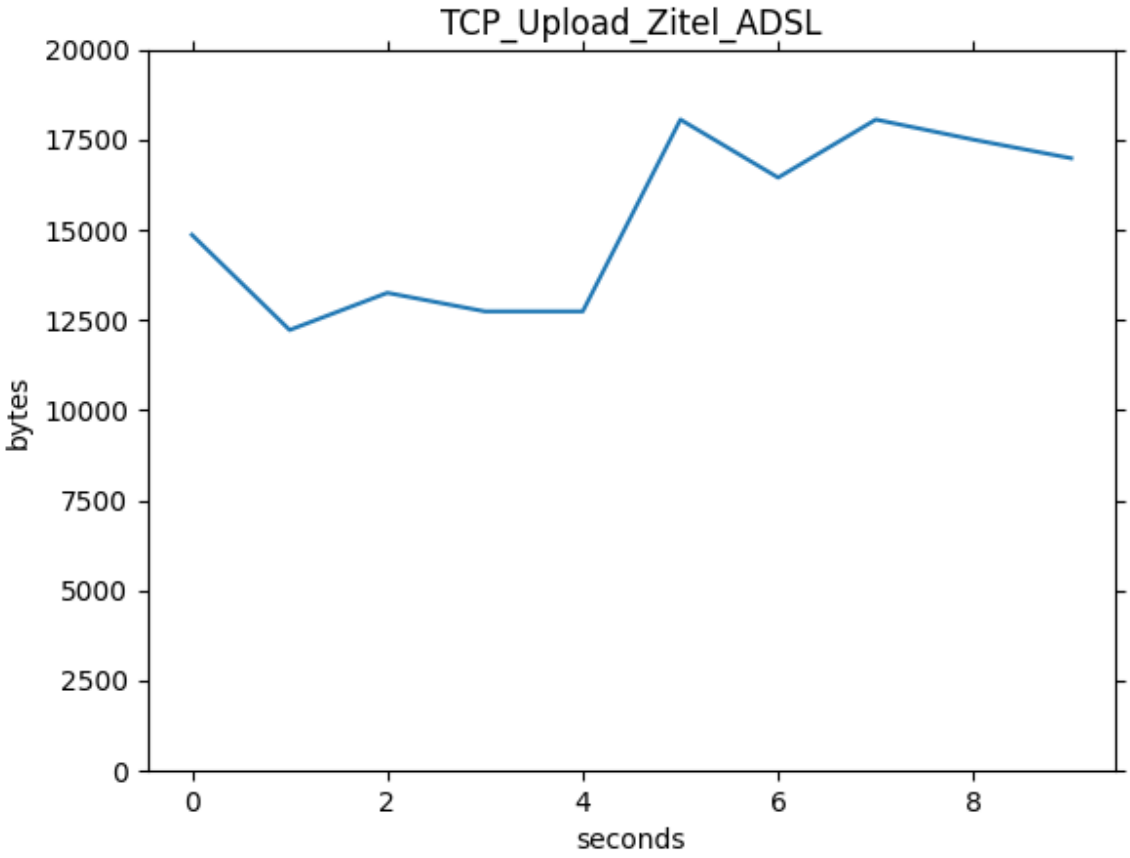
حدود ۳۵ آزمایش مختلف، بر روی سه ارایه دهنده اینترنت رایتل، همراه اول و زیتل انجام شد. این آزمایش‌ها شامل دانلود و آپلود با استفاده از سه پروتکل TCP، UDP و HTTP بودند. همچنین برای همراه اول، نسل‌های چهارم، سوم و دوم و برای رایتل، نسل‌های چهارم و سوم آزمایش شدند. نتایج این آزمایش‌ها را در نمودارهای ادامه گزارش مشاهده خواهید کرد.

۶,۱. ارائه دهنده زیتل

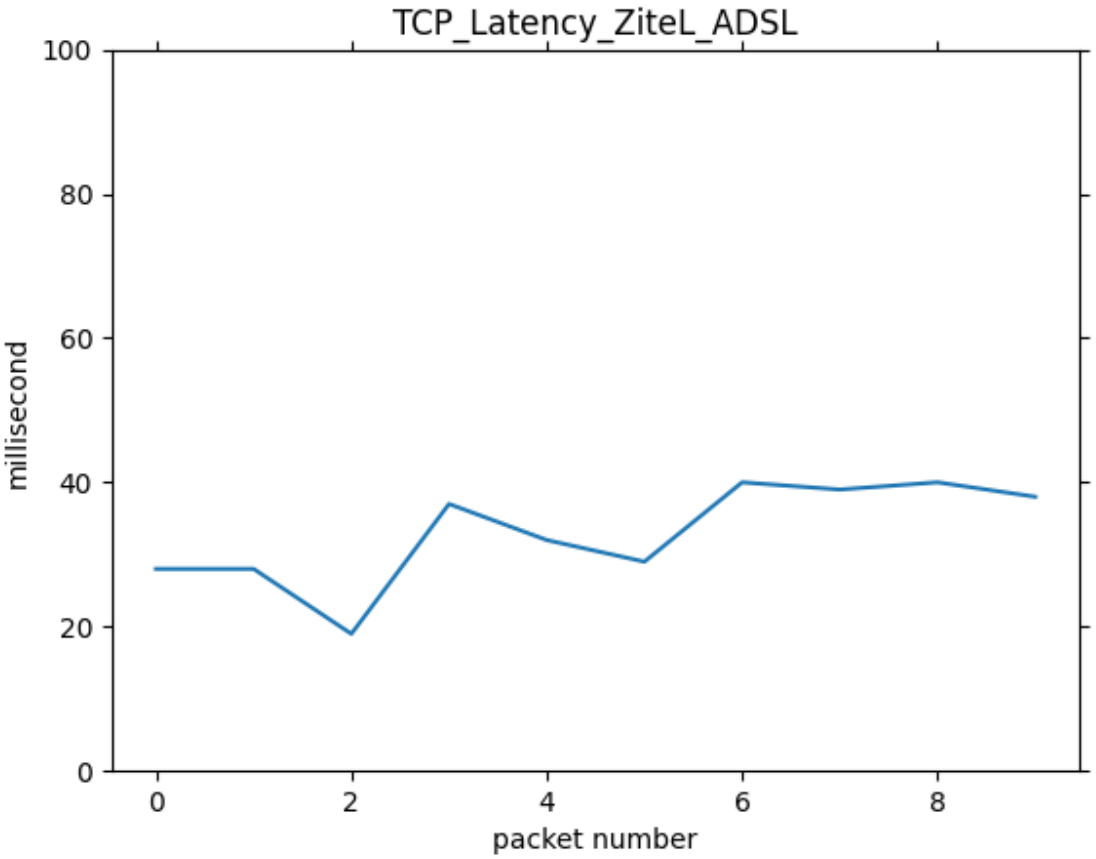
۶،۱،۱. نمودارهای هر آزمایش



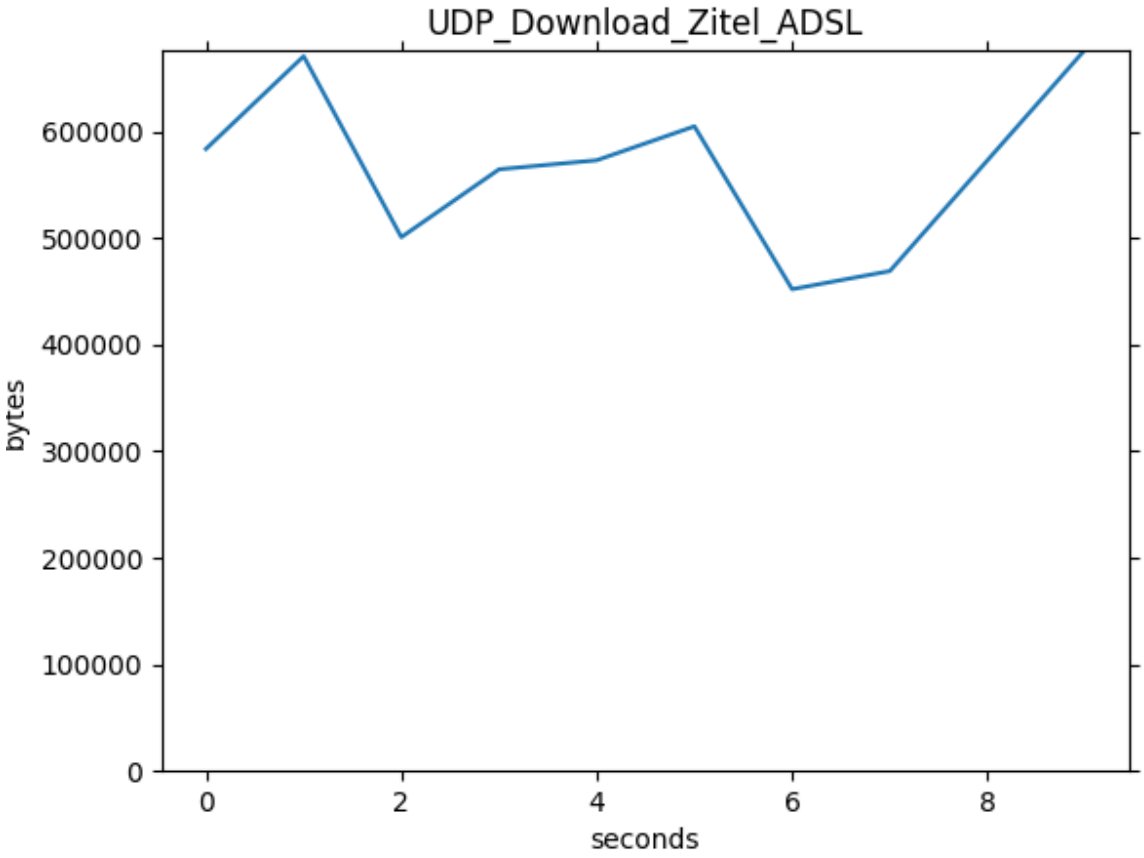
نمودار بالا، bandwidth را برای دانلود با استفاده از روش TCP برای این ارایه‌دهنده نمایش می‌دهد. میانگین سرعت دانلود حدود ۶۰ کیلوبایت بر ثانیه است.



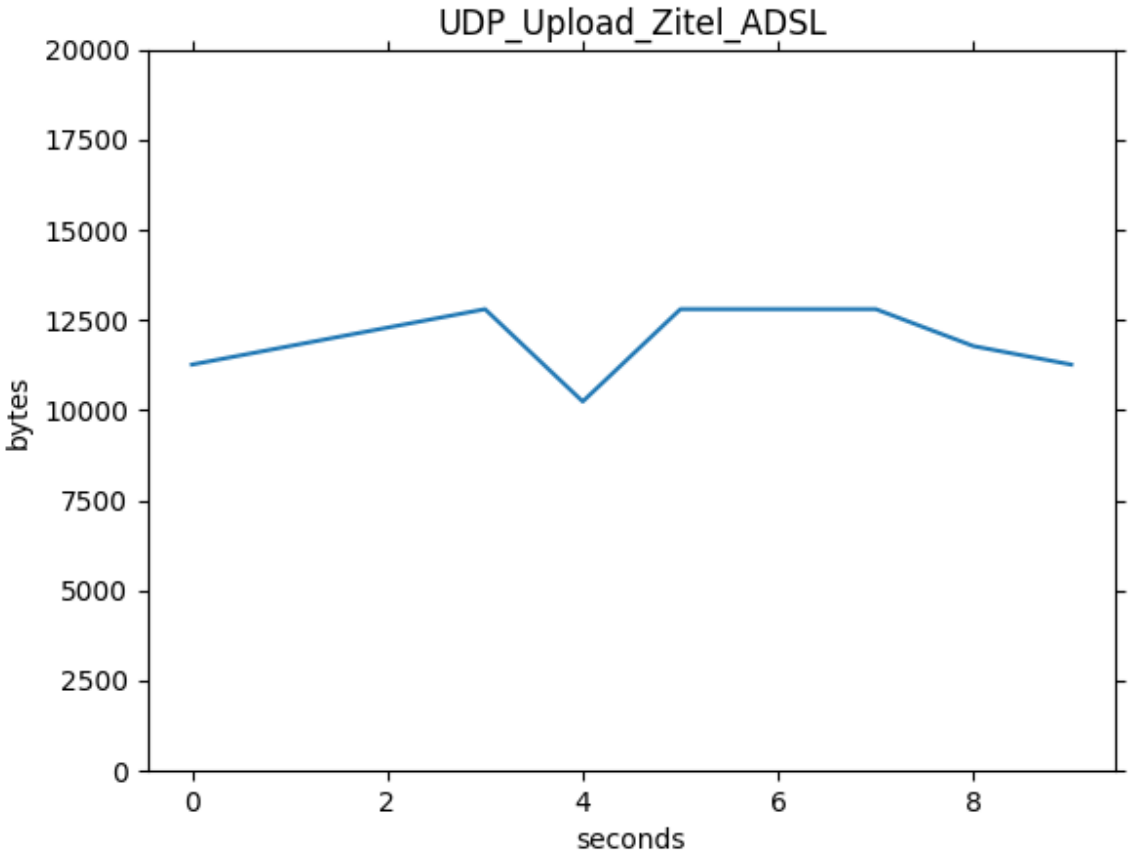
نمودار بالا سرعت آپلود را برای این ارایه‌دهنده نمایش می‌دهد. میانگین سرعت آپلود برای روش TCP، حدود ۱۵ کیلوبایت بر ثانیه است.



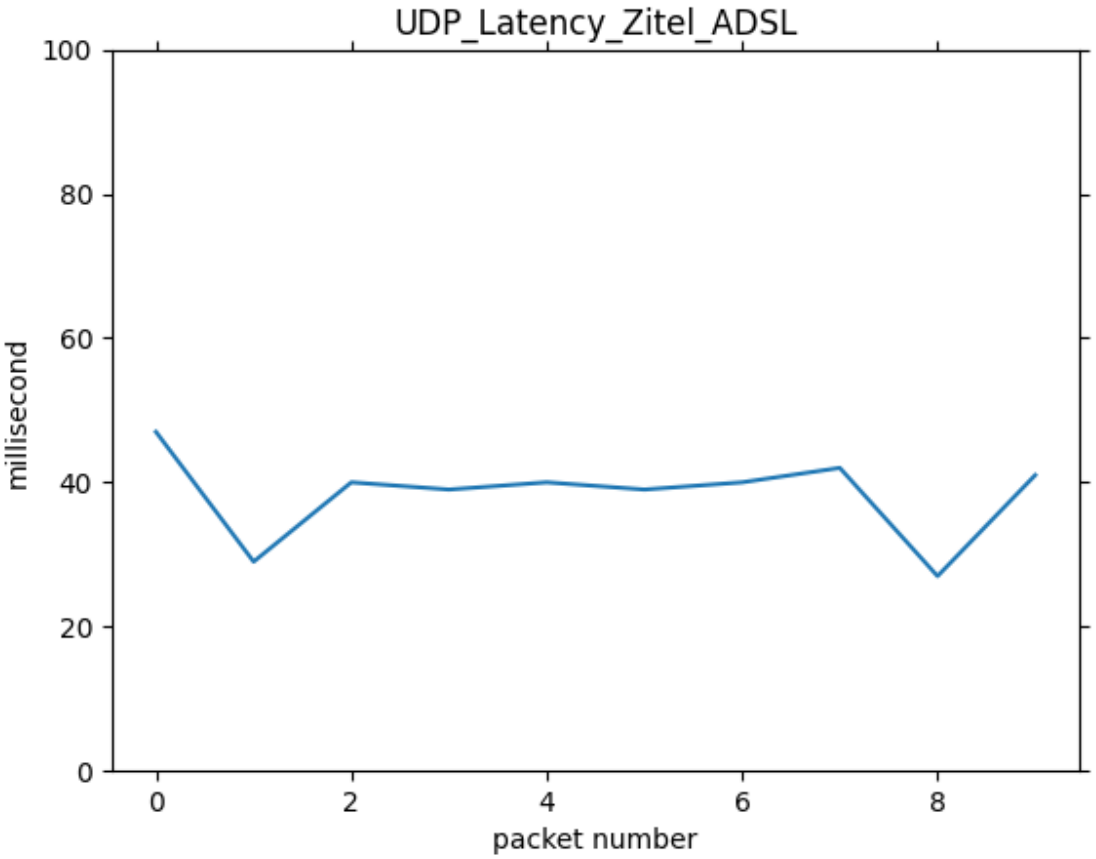
میزان تاخیر در حین دانلود و آپلود برای روش TCP در نمودار بالا نمایش داده شده است. میانگین تاخیر در حدود ۳۵ میلی‌ثانیه بوده است.



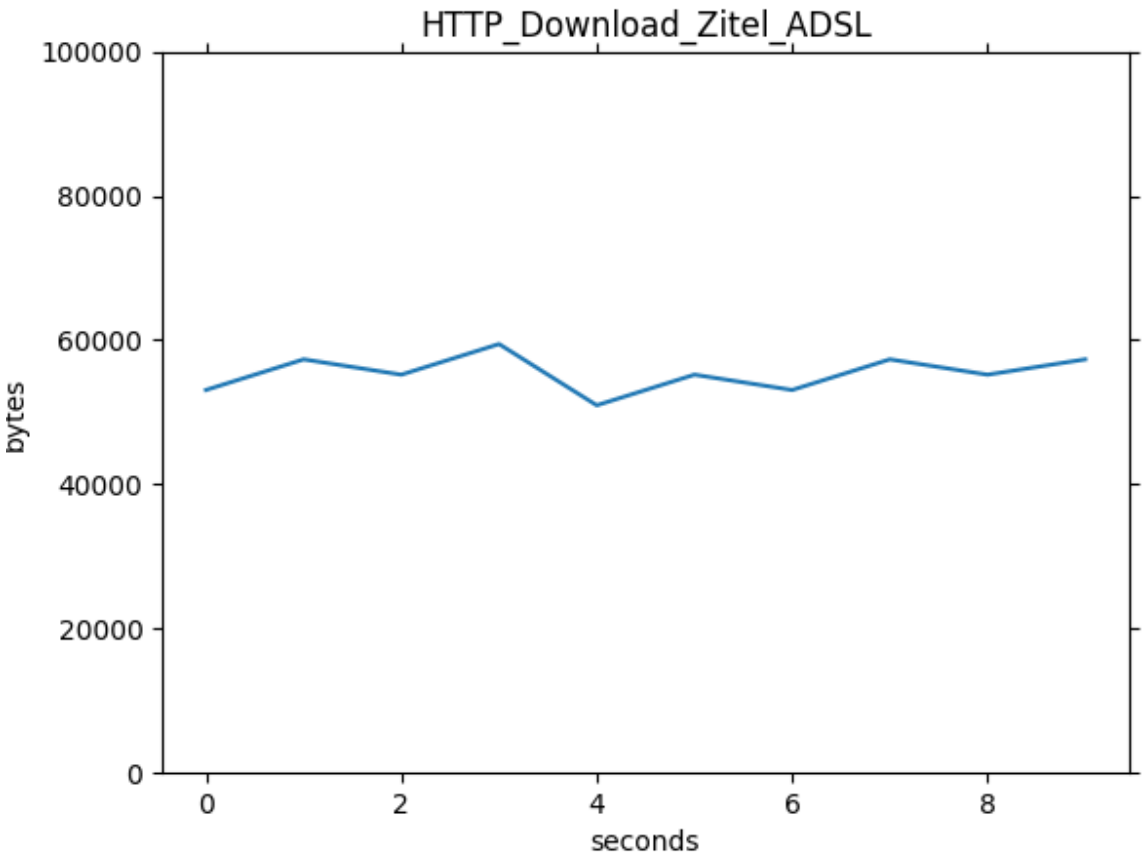
نمودار، نشان‌دهنده سرعت دانلود UDP برای این ارایه‌دهنده است. سرعت دانلود این ارایه‌دهنده برای پروتکل UDP حدود ۵۵۰ کیلوبایت است.



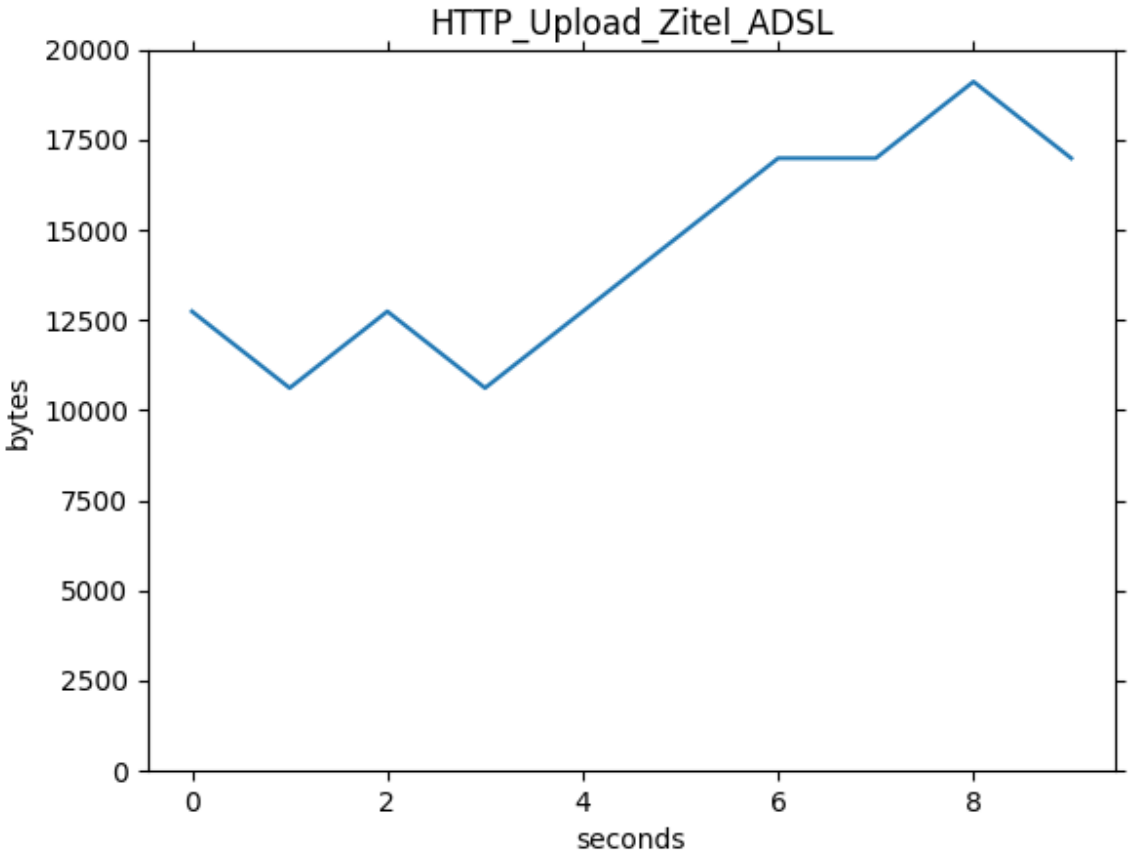
نمودار نشان‌دهنده سرعت آپلود با استفاده از روش UDP است. Bandwidth این شرایط میانگین ۱۱ کیلوبایت را داراست.



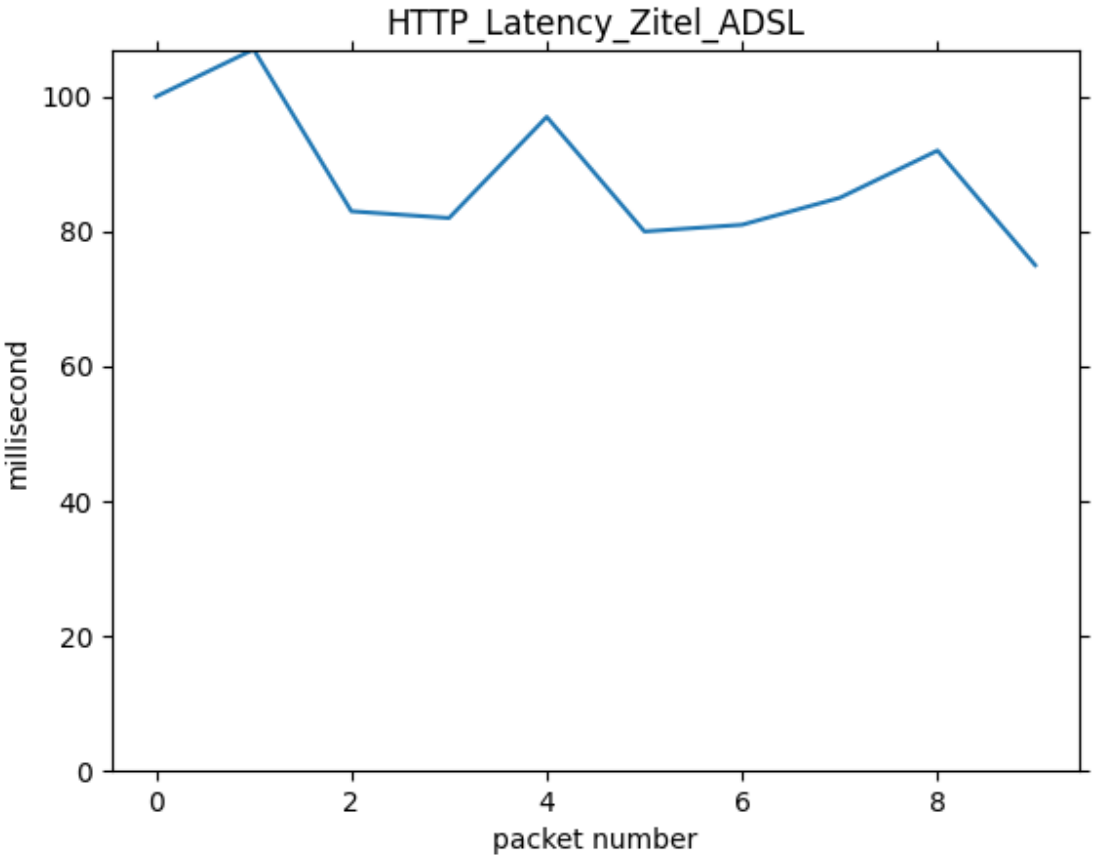
تاخیر روش UDP برای این ارایه‌دهنده، میانگین ۴۰ میلی‌ثانیه را داراست.



این نمودار، سرعت دانلود برای پروتکل HTTP را نشان می‌دهد. این پروتکل، بیشترین ثبات را داراست و میانگین ۵۸ کیلوبایت را دارد.

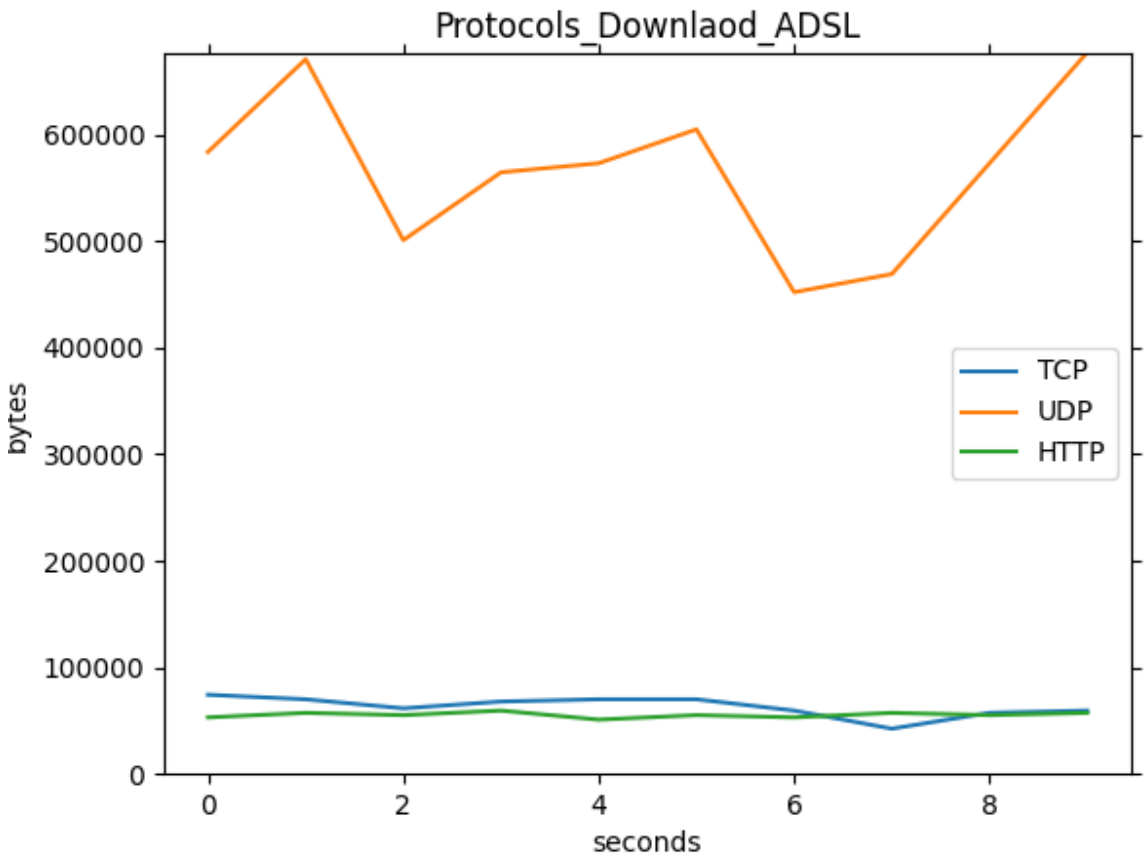


این نمودار، سرعت آپلود را برای پروتکل HTTP نشان می‌دهد. میانگین این نمودار، ۱۵ کیلوبایت است.

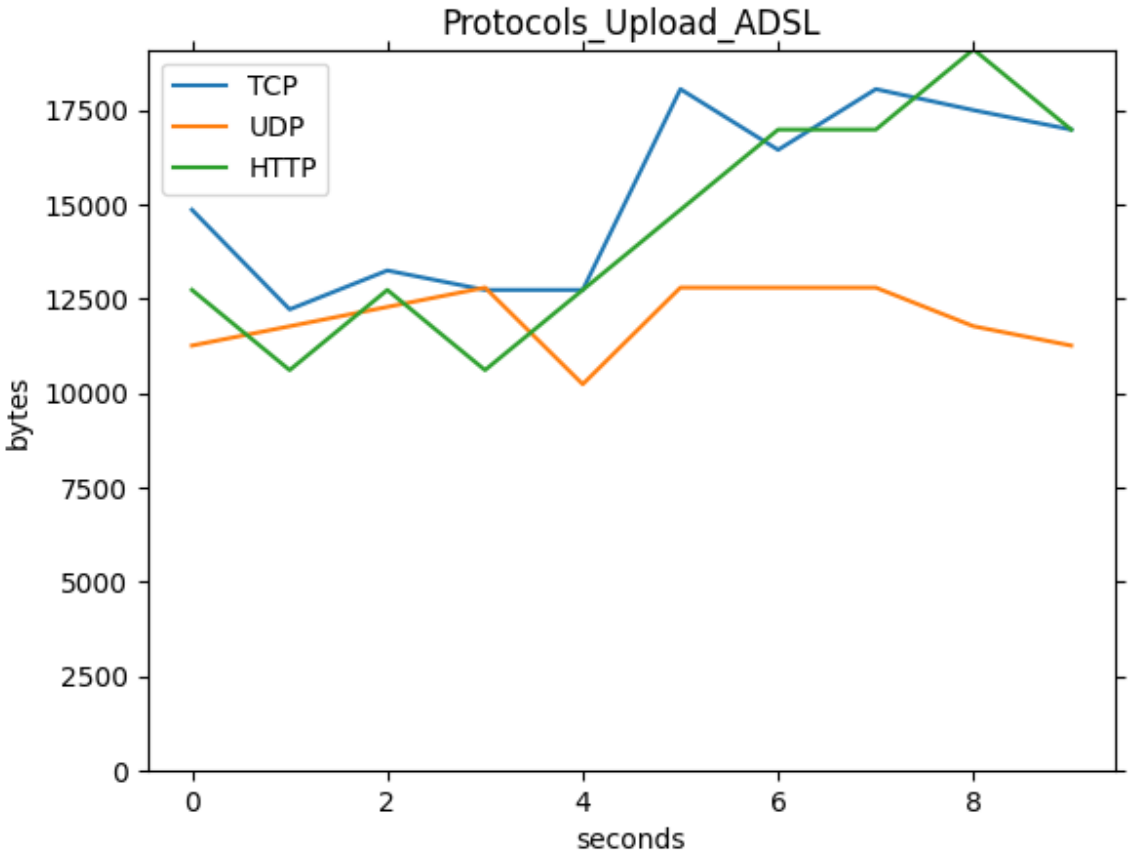


این نمودار، تاخیر روش HTTP را برای این ارایه‌دهنده نمایش می‌دهد. تاخیر این روش، از UDP و TCP بیشتر است و میانگینی حدود ۹۰ میلی‌ثانیه دارد.

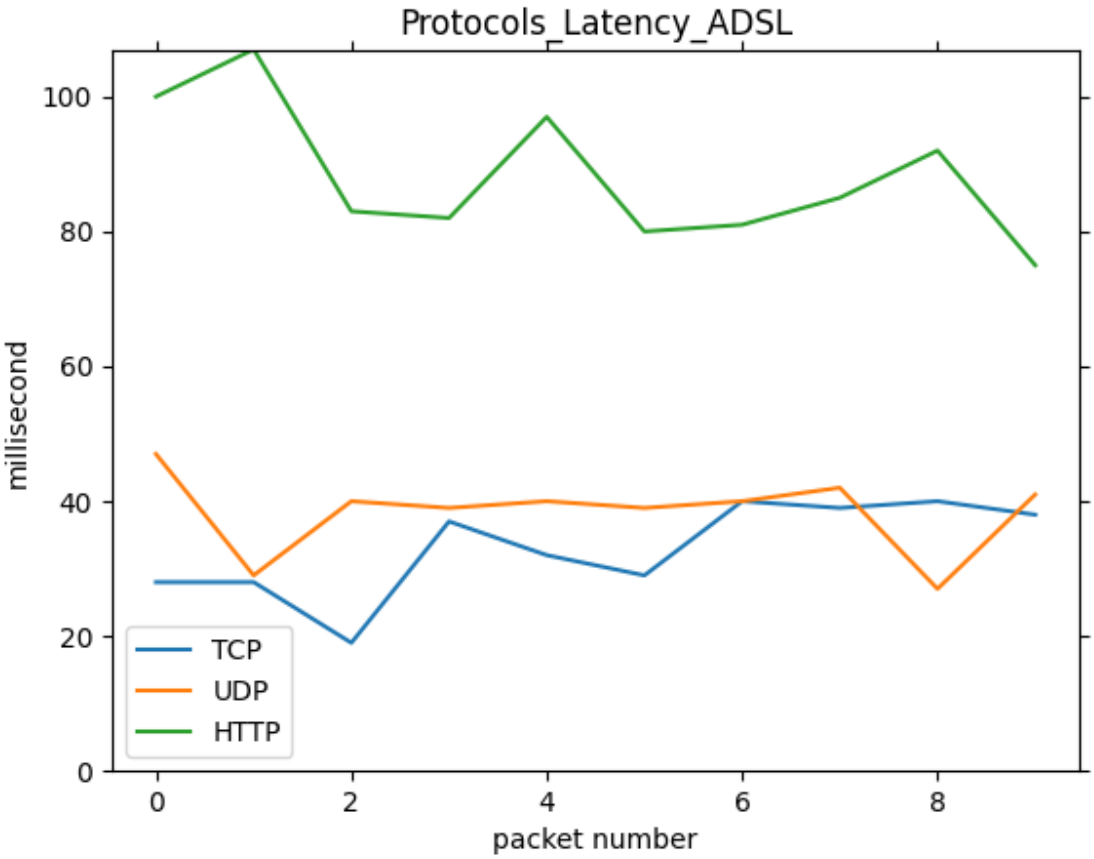
۶،۱،۲. مقایسه پروتکل‌های مختلف



سرعت دانلود روش‌های مختلف در این نمودار با هم مقایسه شده‌اند. همانطور که مشاهده می‌شود، سرعت دانلود HTTP و TCP بسیار مشابه هم است که با توجه به اینکه روش HTTP بر پایه TCP عمل می‌کند این ارزیابی قابل پیش‌بینی است. سرعت دانلود UDP از دو روش دیگر بسیار بیشتر است که به همین دلیل است که در استریم ویدیو از این روش استفاده می‌شود.



سرعت آپلود این سه پروتکل در نمودار بالا با هم مقایسه شده‌است. مشابه نمودار قبل، نمودار TCP و HTTP مشابه یکدیگر است. میانگین آپلود UDP از سه روش دیگر پایینتر است.

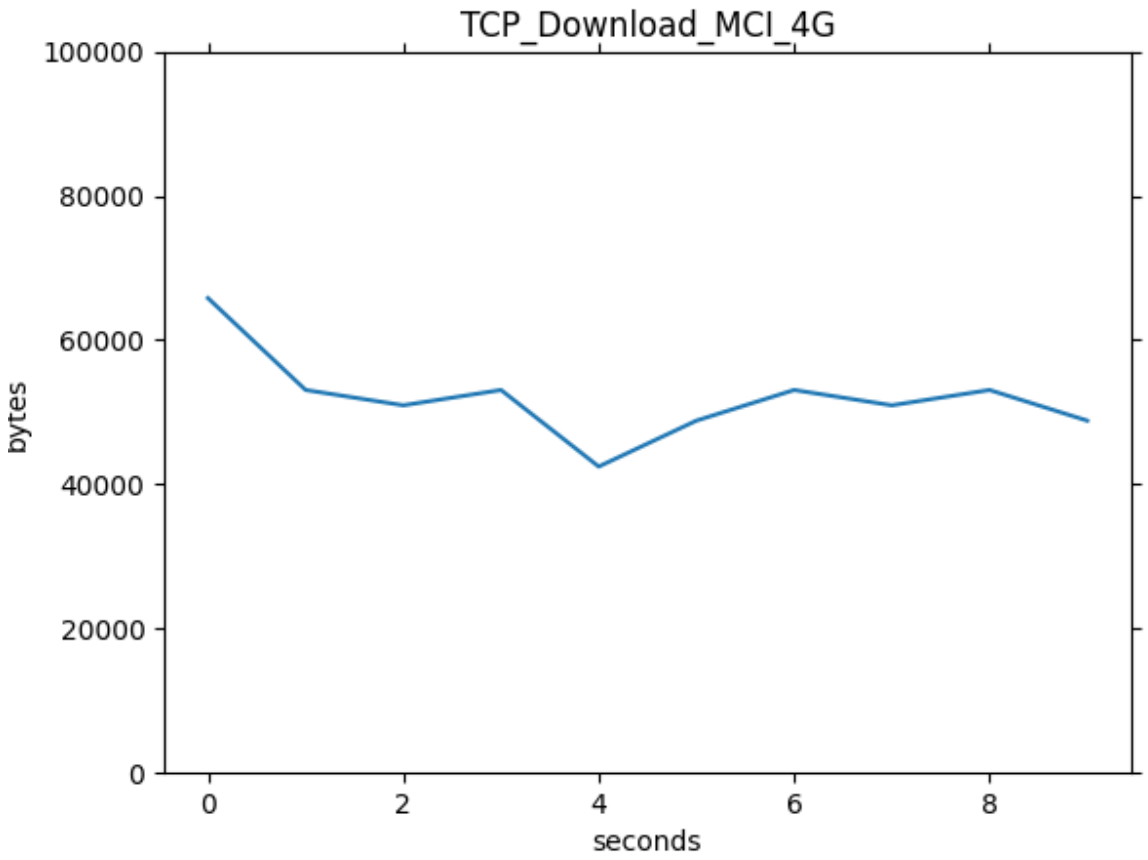


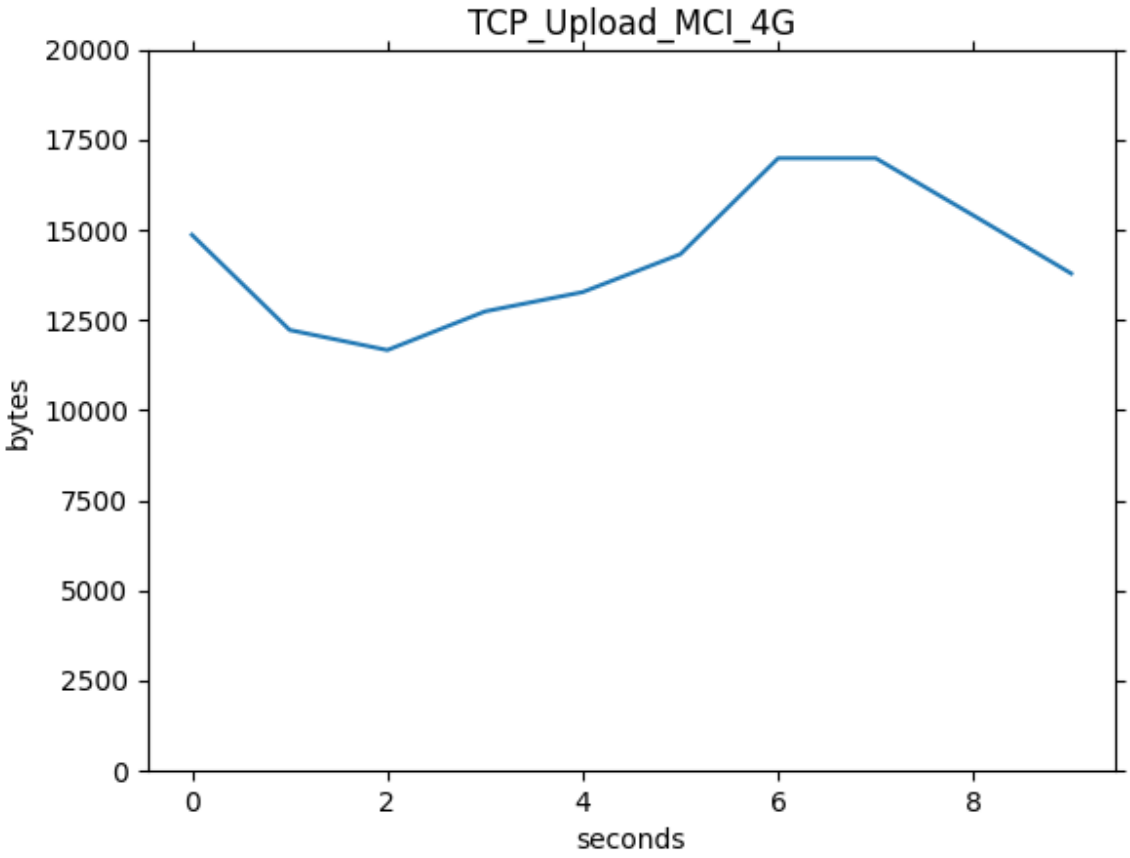
تاخیر سه روش در نمودار بالا مقایسه شده‌است. بیشترین تاخیر در پروتکل HTTP دیده می‌شود. تفاوت این پروتکل با TCP بدلیل overhead آن است.

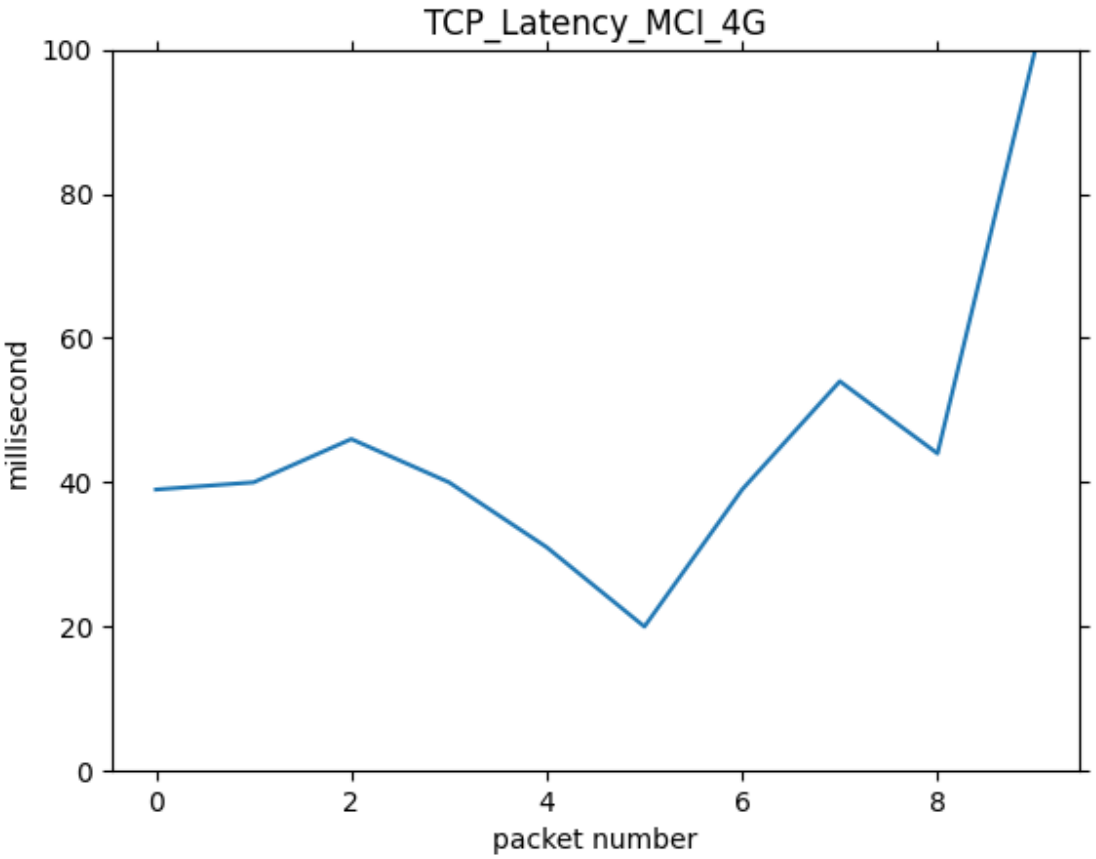
۶,۲. ارائه دهنده همراه اول

۶,۲,۱. اینترنت نسل ۴

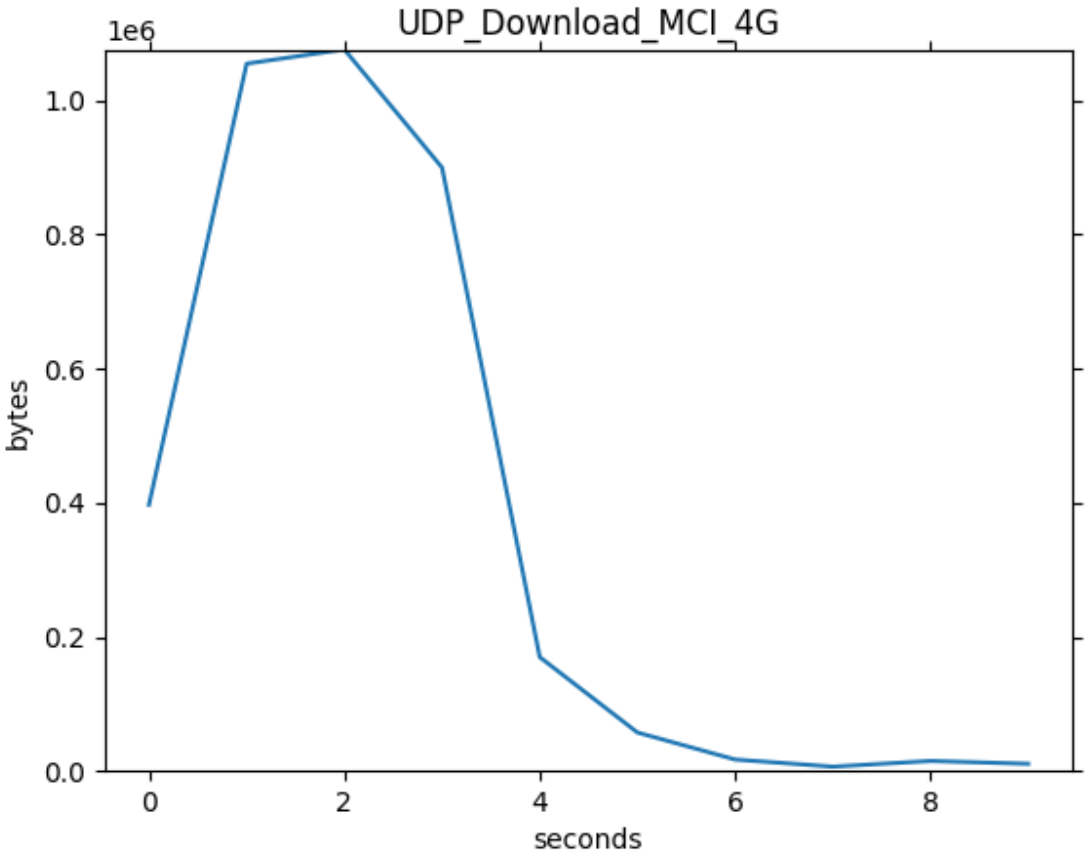
نمودارهای دانلود، آپلود و تاخیر برای پروتکل TCP را در زیر مشاهده می‌کنید. میانگین دانلود ۵۰ کیلوبایت، آپلود ۱۵ کیلوبایت و تاخیر ۶۰ میلی‌ثانیه است.

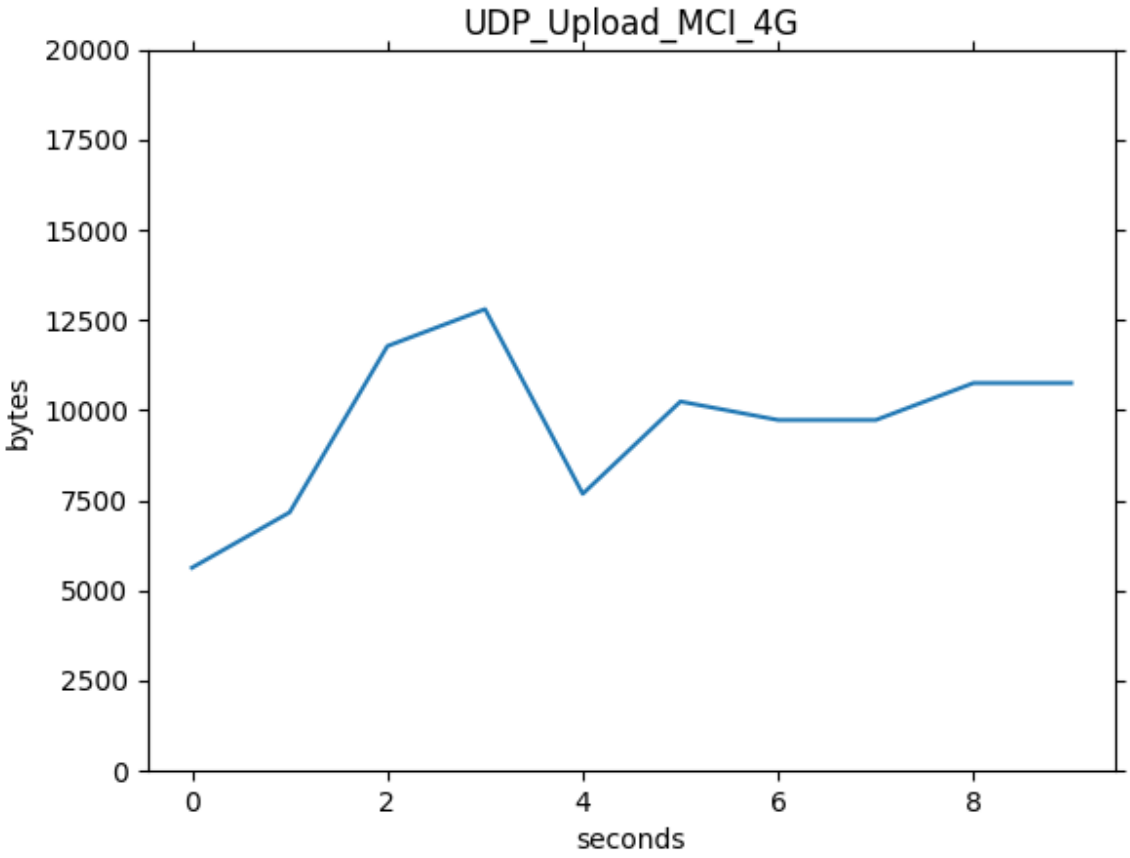

TCP_Download_MCI_4G
100000
80000
60000
40000
20000
0
bytes
0
2
4
6
8
seconds

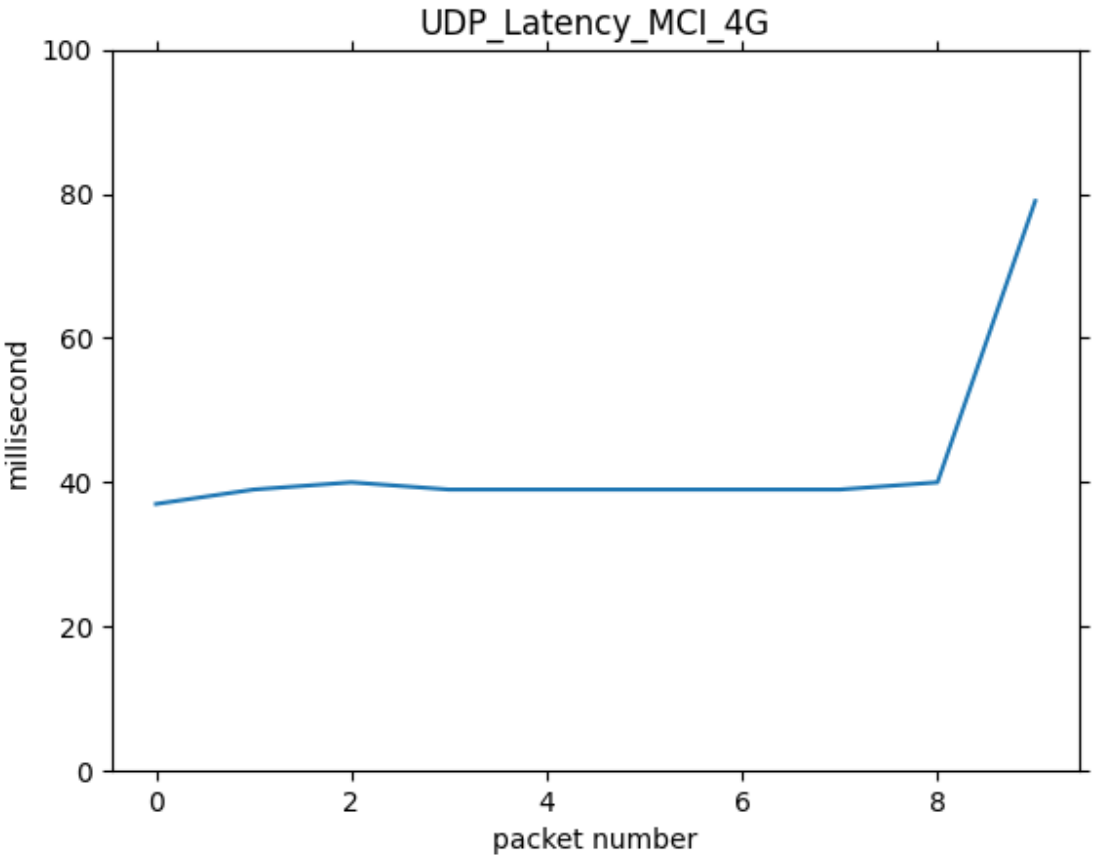

TCP_Upload_MCI_4G
20000
17500
15000
12500
10000
7500
5000
2500
0
bytes
0
2
4
6
8
seconds



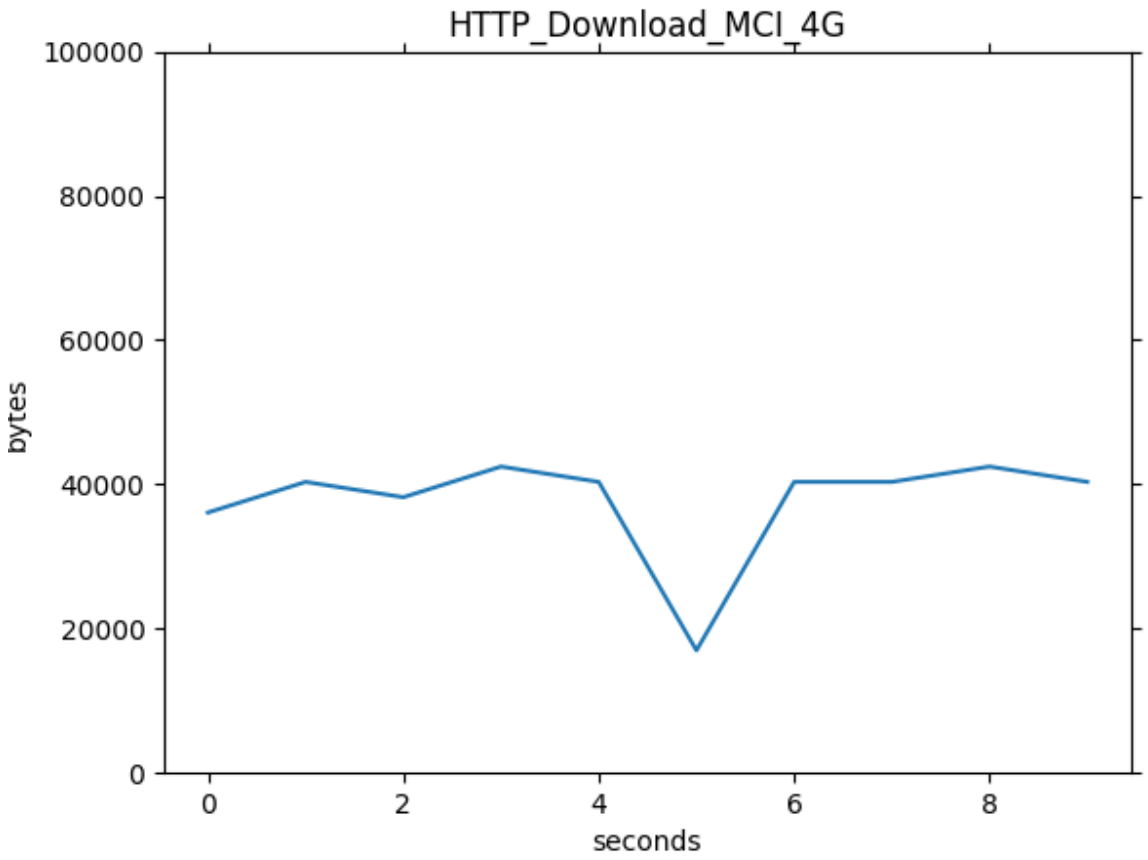
نمودارهای زیر پروتکل UDP را نشان می‌دهند. میانگین دانلود ۶۰۰ کیلوبایت و throughput آن حدود ۲/۱ مگابایت است. میانگین آپلود ۱۰ کیلوبایت و میانگین تاخیر ۴۳ میلی‌ثانیه است.

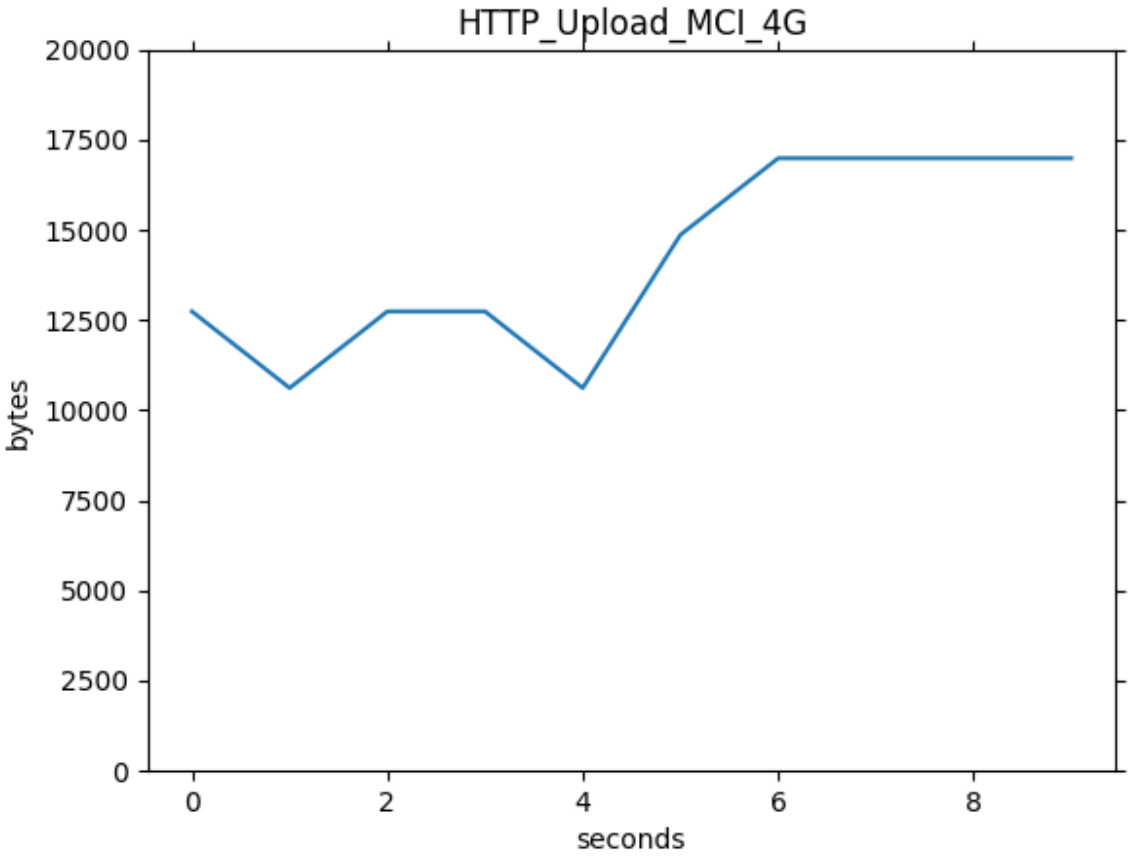

UDP_Download_MCI_4G
1e6
1.0
0.8
0.6
0.4
0.2
0.0
bytes
0
2
4
6
8
seconds

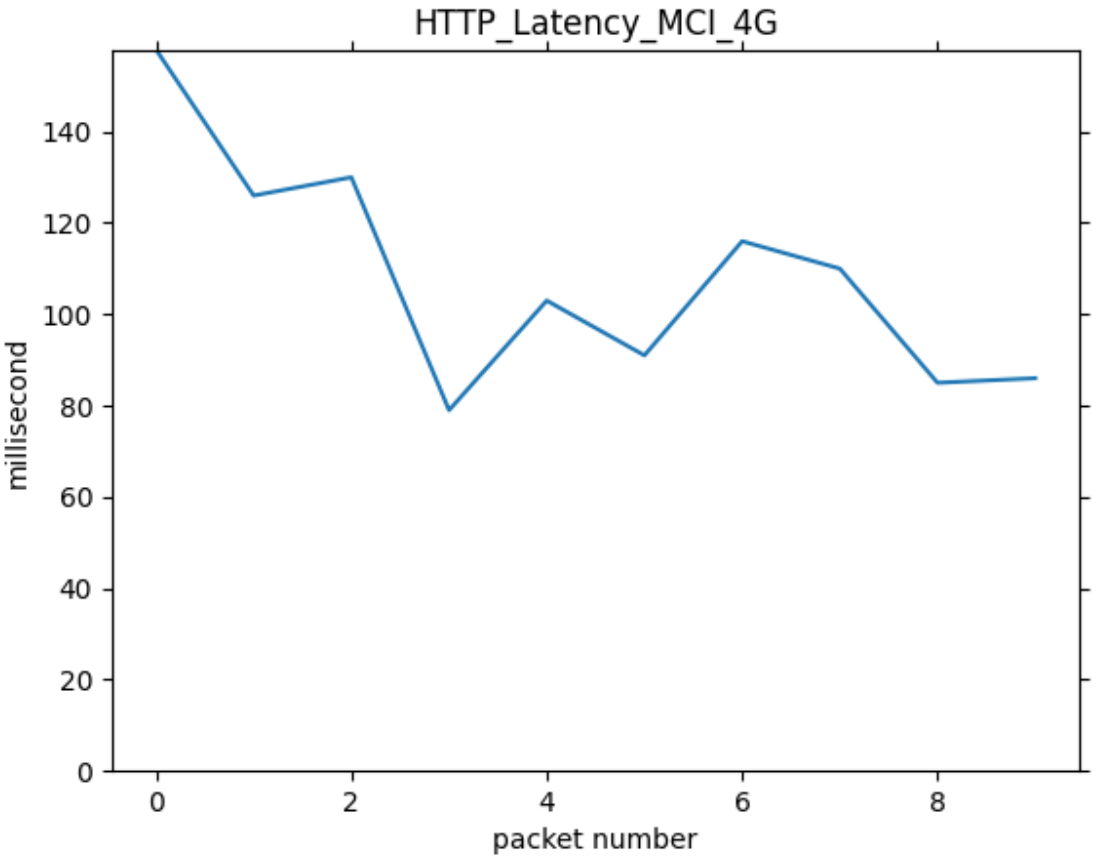

UDP_Upload_MCI_4G
20000
17500
15000
12500
10000
7500
5000
2500
0
bytes
0
2
4
6
8
seconds



سه نمودار آخر این بخش نیز متعلق به پروتکل HTTP است. میانگین دانلود ۳۸ کیلوبایت بر ثانیه، آپلود ۱۵ کیلوبایت و تاخیر حدود ۱۰۰ میلی‌ثانیه است.

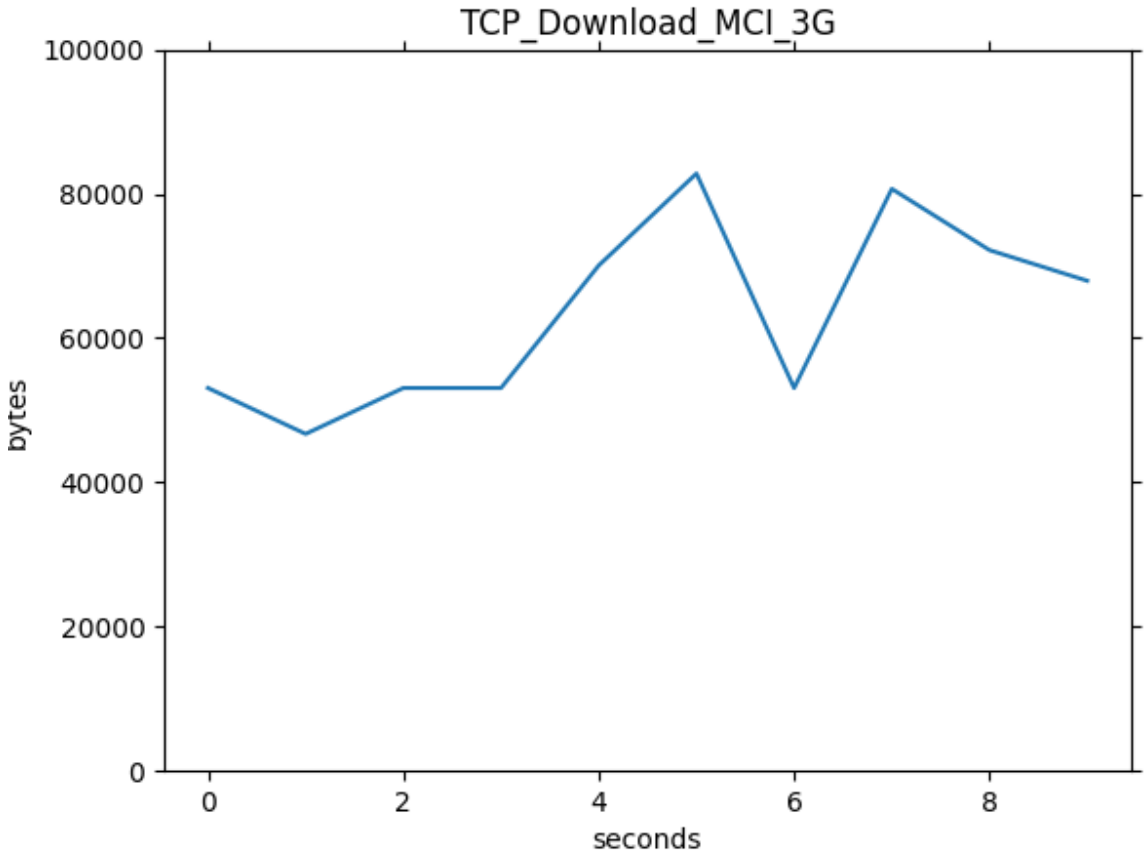

HTTP_Download_MCI_4G
100000
80000
60000
40000
20000
0
bytes
0
2
4
6
8
seconds

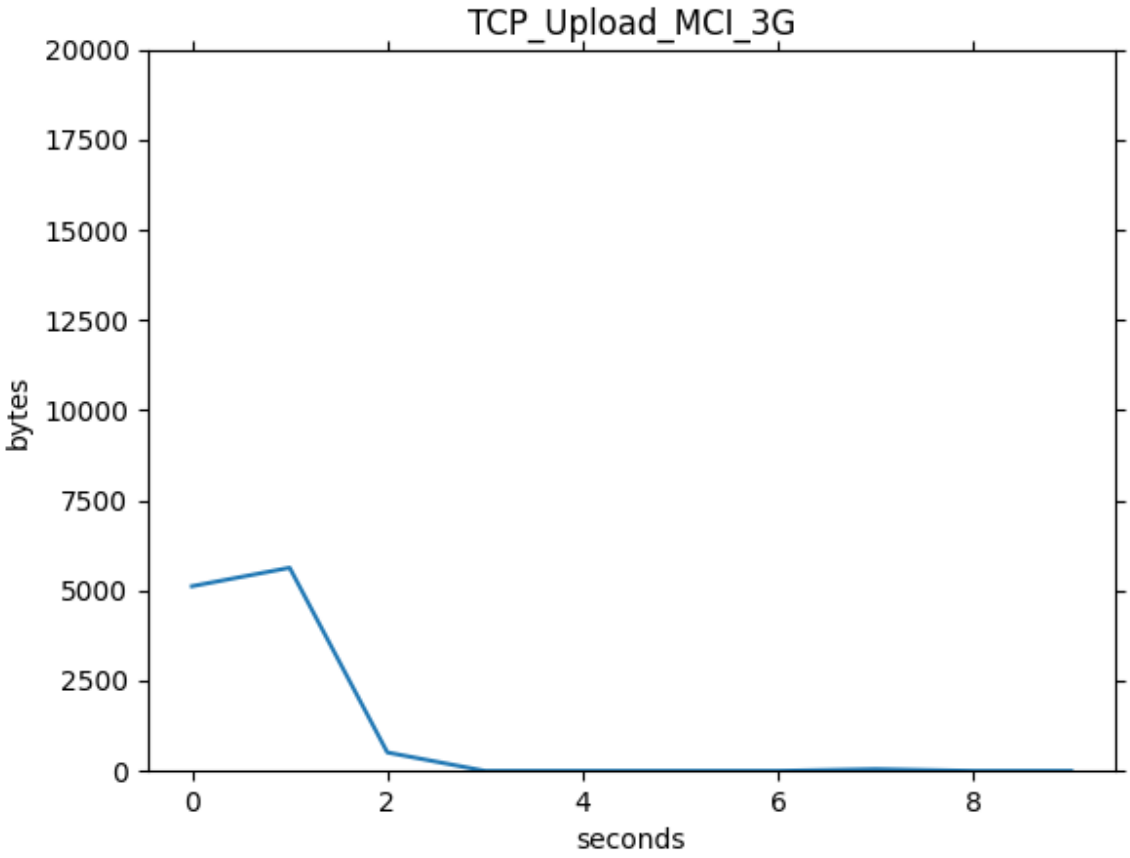

HTTP_Upload_MCI_4G
20000
17500
15000
12500
10000
7500
5000
2500
0
bytes
0
2
4
6
8
seconds

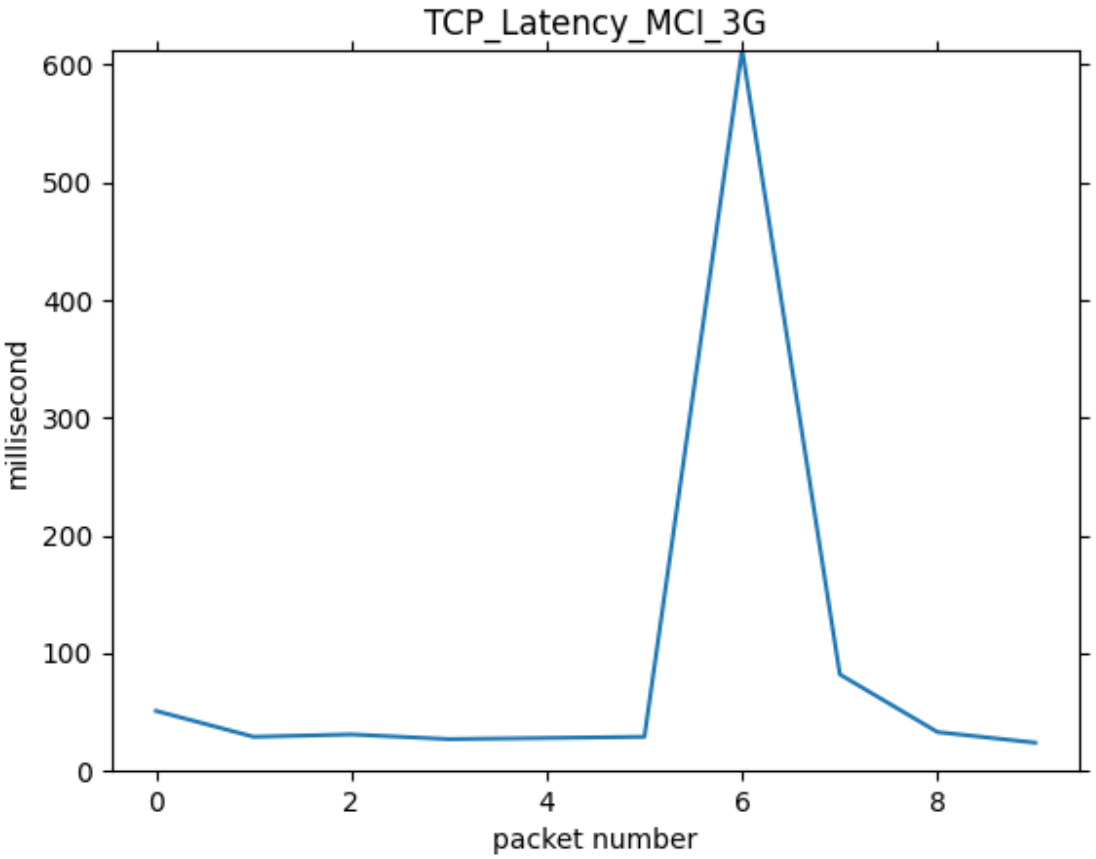


۶,۲,۲. اینترنت نسل ۳

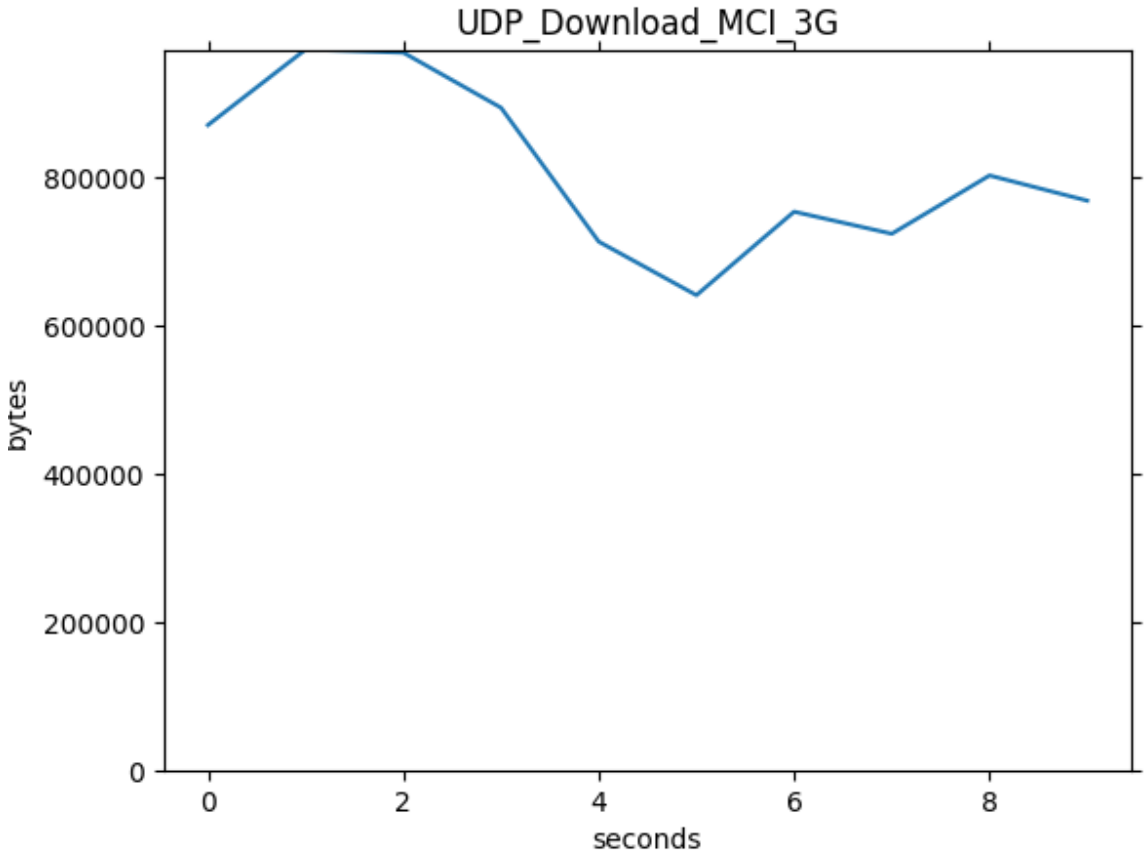
نمودارهای زیر مربوط به پروتکل TCP است. میانگین سرعت دانلود ۶۵ کیلوبایت، آپلود ۱ کیلوبایت و تاخیر،با صرف از نظر از افزایش ناگهانی در نقطه ۶ حدود ۴۰ میلی ثانیه است.

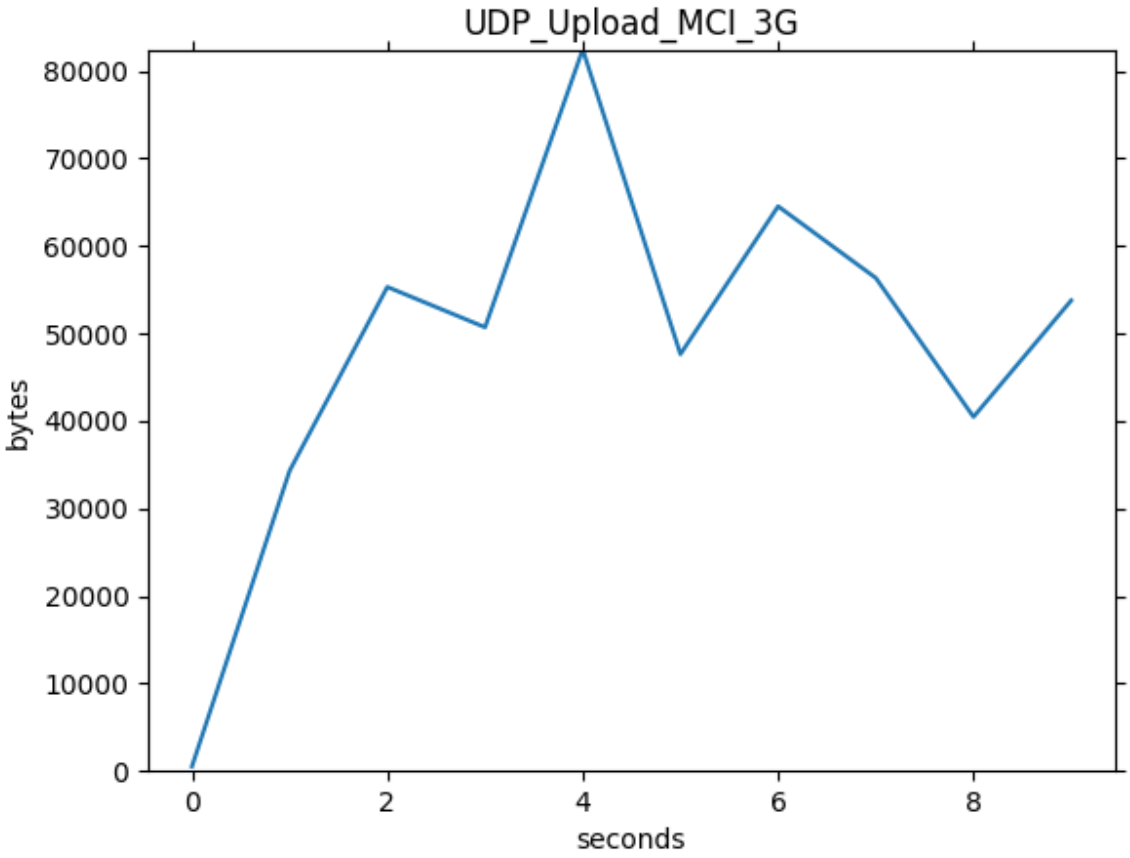

TCP_Download_MCI_3G
100000
80000
60000
40000
20000
0
bytes
0
2
4
6
8
seconds

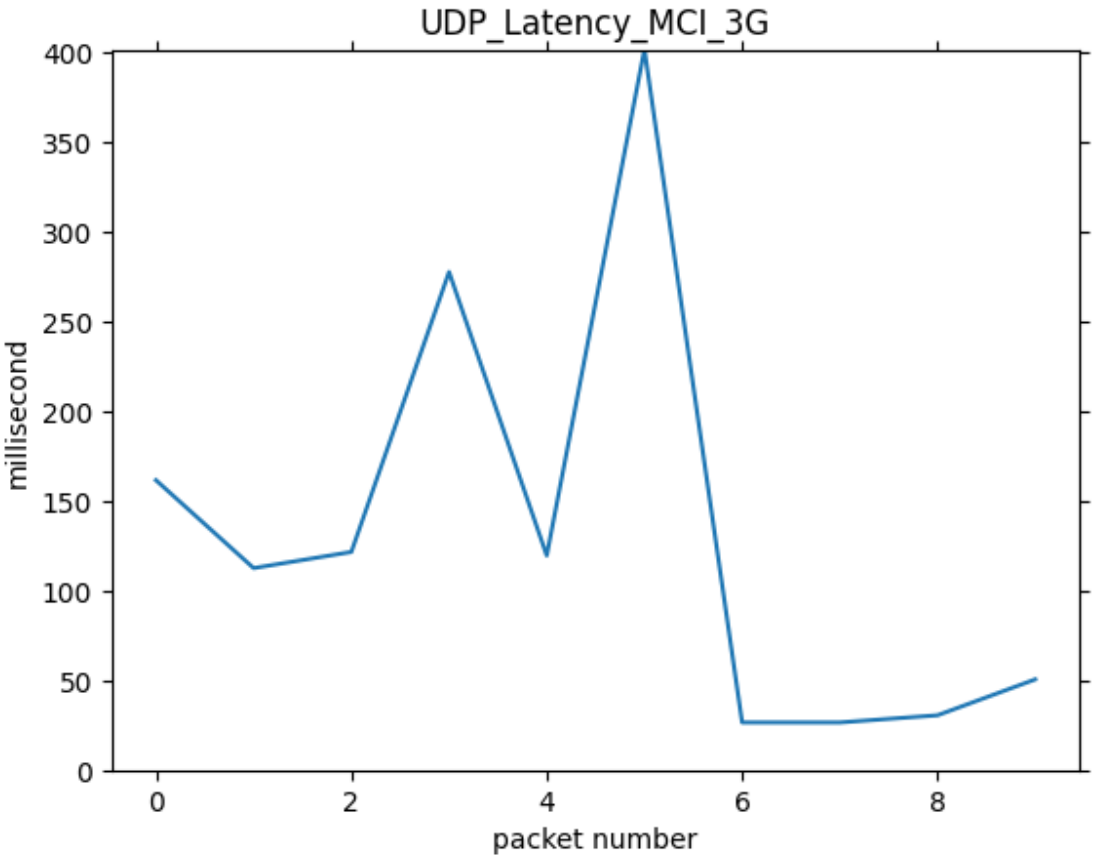

TCP_Upload_MCI_3G
20000
17500
15000
12500
10000
7500
5000
2500
0
bytes
0
2
4
6
8
seconds



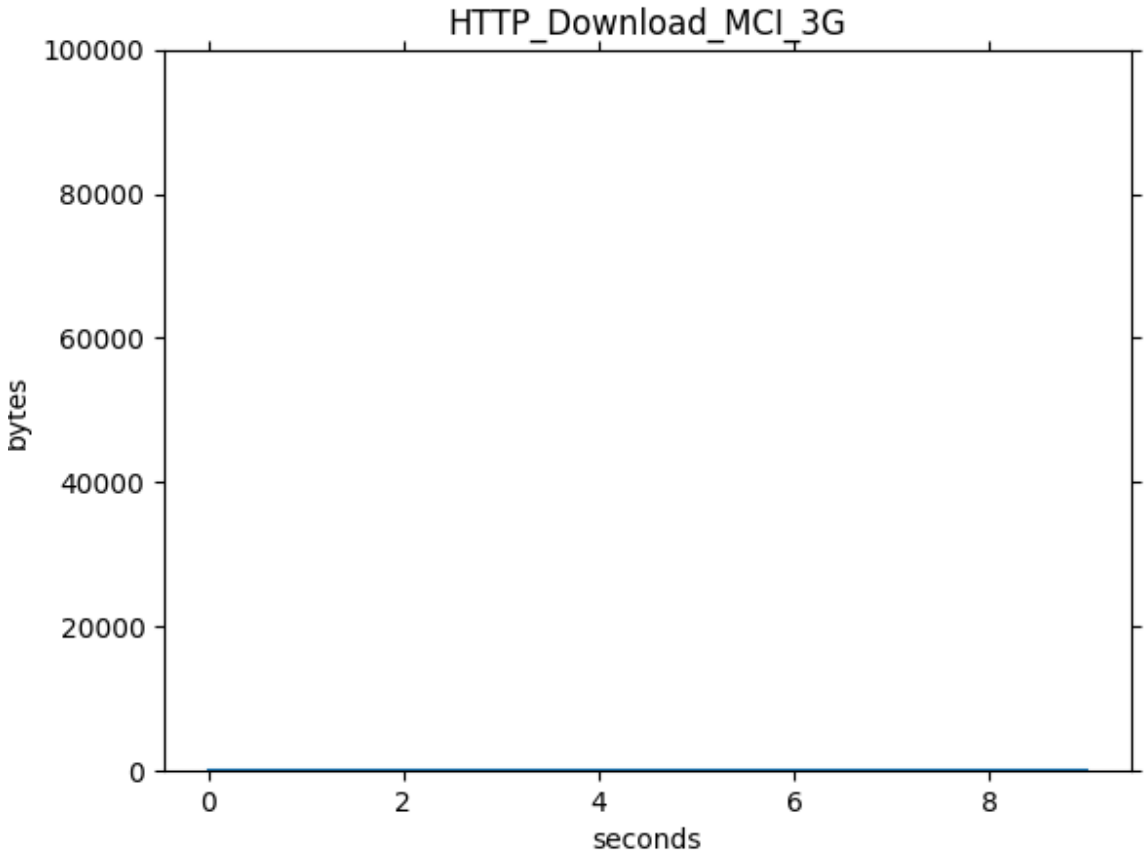
نمودارهای زیر مربوط به روش UDP هستند. میانگین دانلود ۸۰۰ کیلوبایت، آپلود ۴۰ کیلوبایت و تاخیر ۲۰۰ میلی ثانیه می‌باشد.

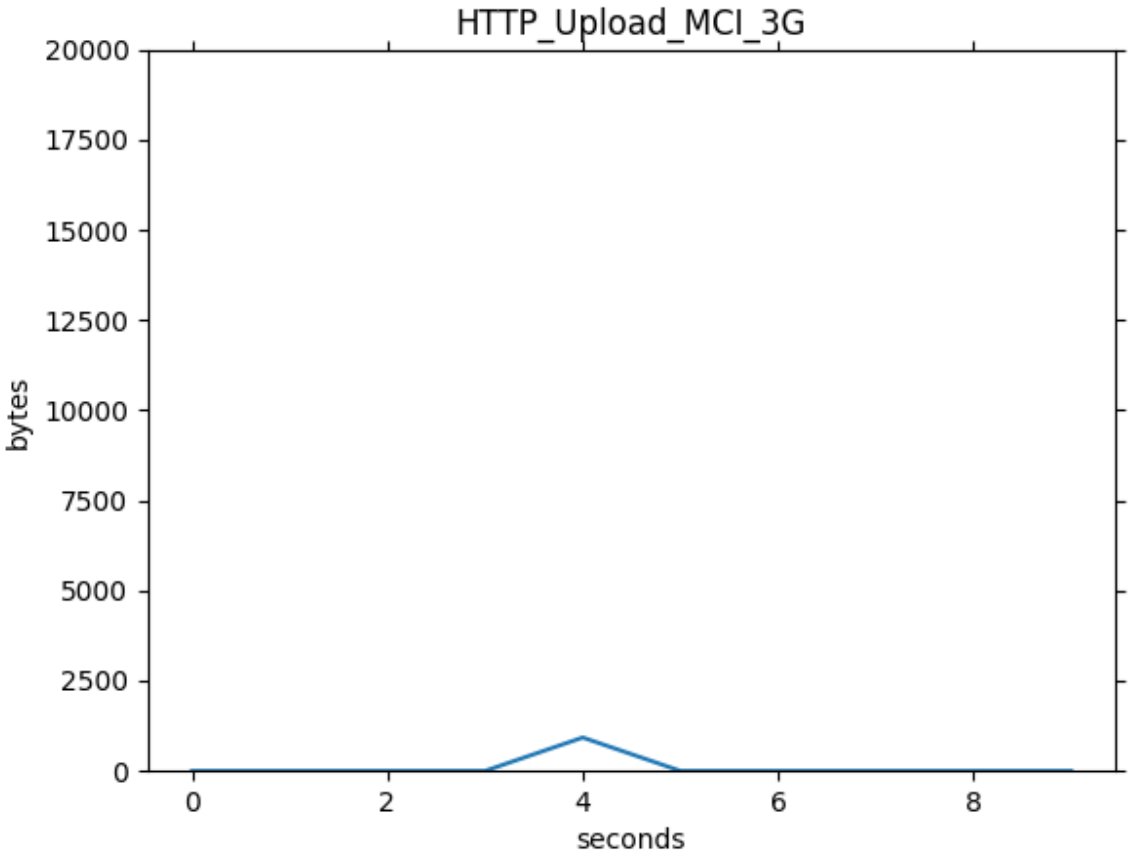

UDP_Download_MCI_3G
800000
600000
400000
200000
0
bytes
0
2
4
6
8
seconds

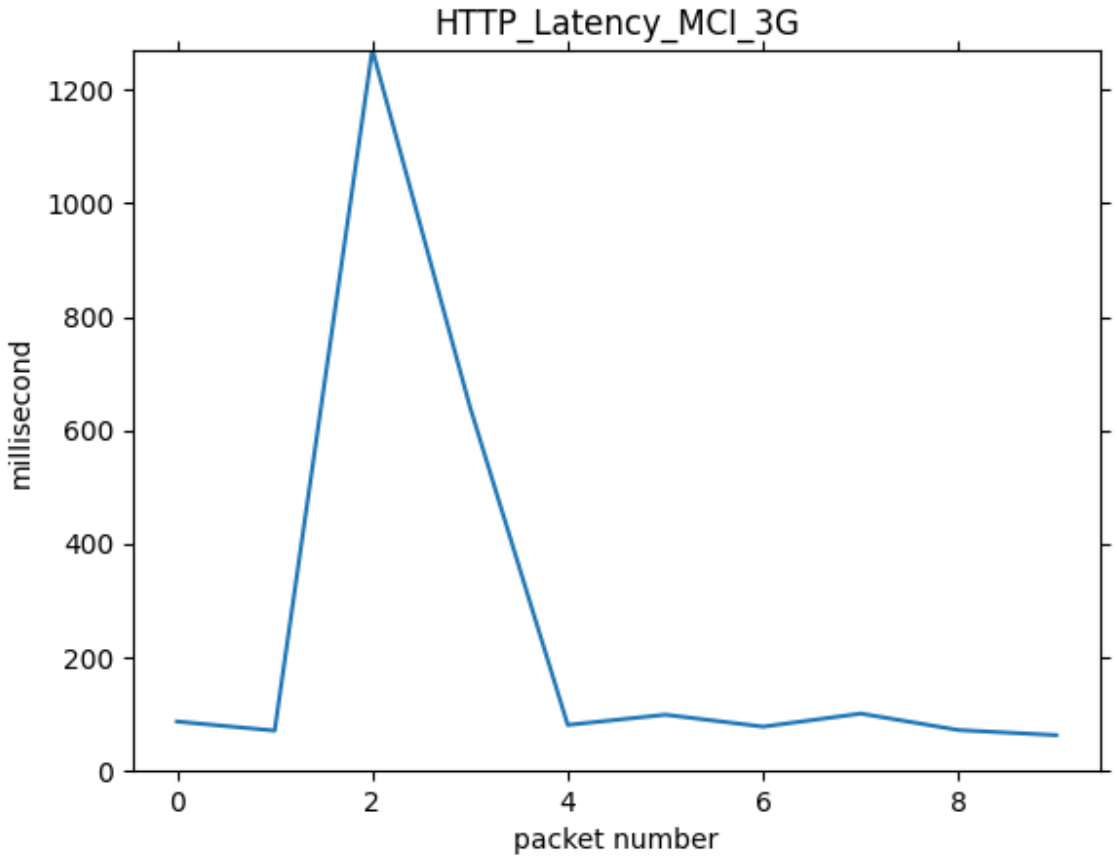

UDP_Upload_MCI_3G
bytes
seconds
0
10000
20000
30000
40000
50000
60000
70000
80000
0
2
4
6
8



نمودارهای زیر مربوط به پروتکل HTTP هستند. سرعت آپلود و دانلود نزدیک صفر است و بیشترین تاخیر تا به اینجا، یعنی ۱۲۰۰ میلی ثانیه را در نمودار این بخش شاهد هستیم.

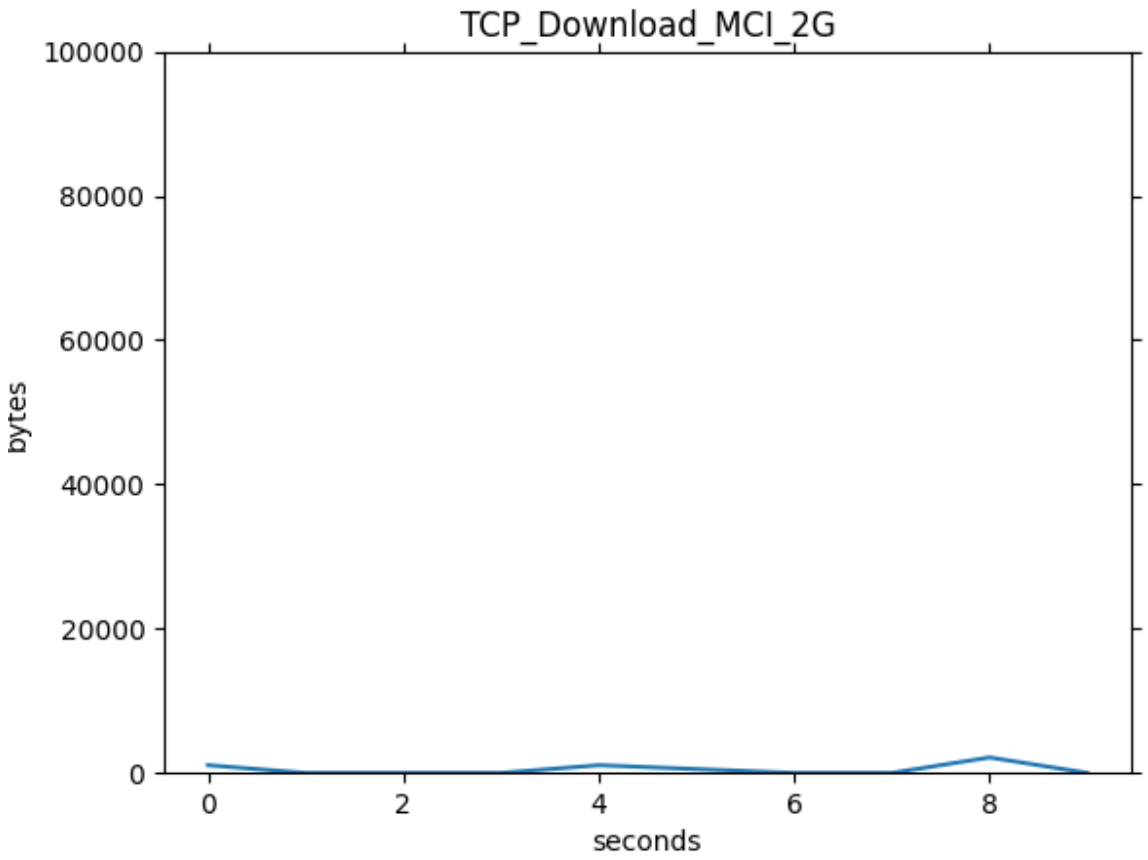

HTTP_Download_MCI_3G
100000
80000
60000
40000
20000
0
bytes
0
2
4
6
8
seconds

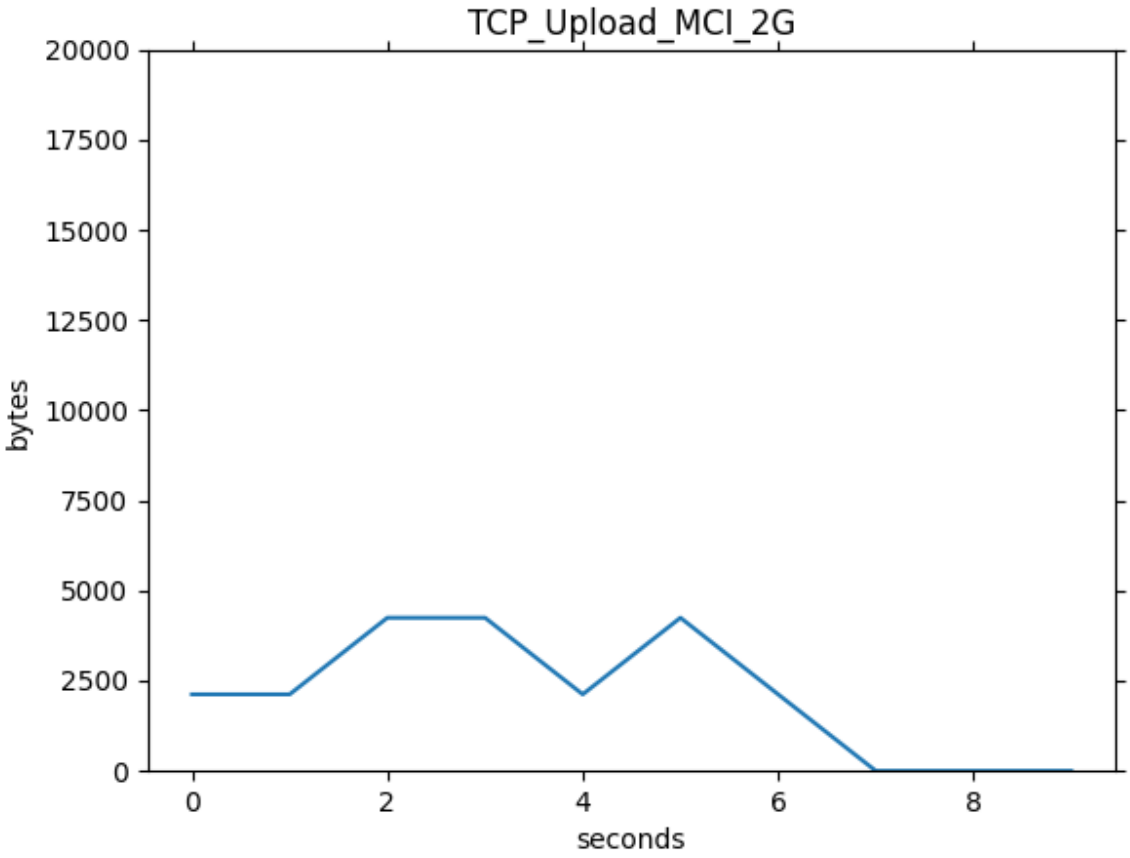

HTTP_Upload_MCI_3G
20000
17500
15000
12500
10000
7500
5000
2500
0
bytes
0
2
4
6
8
seconds

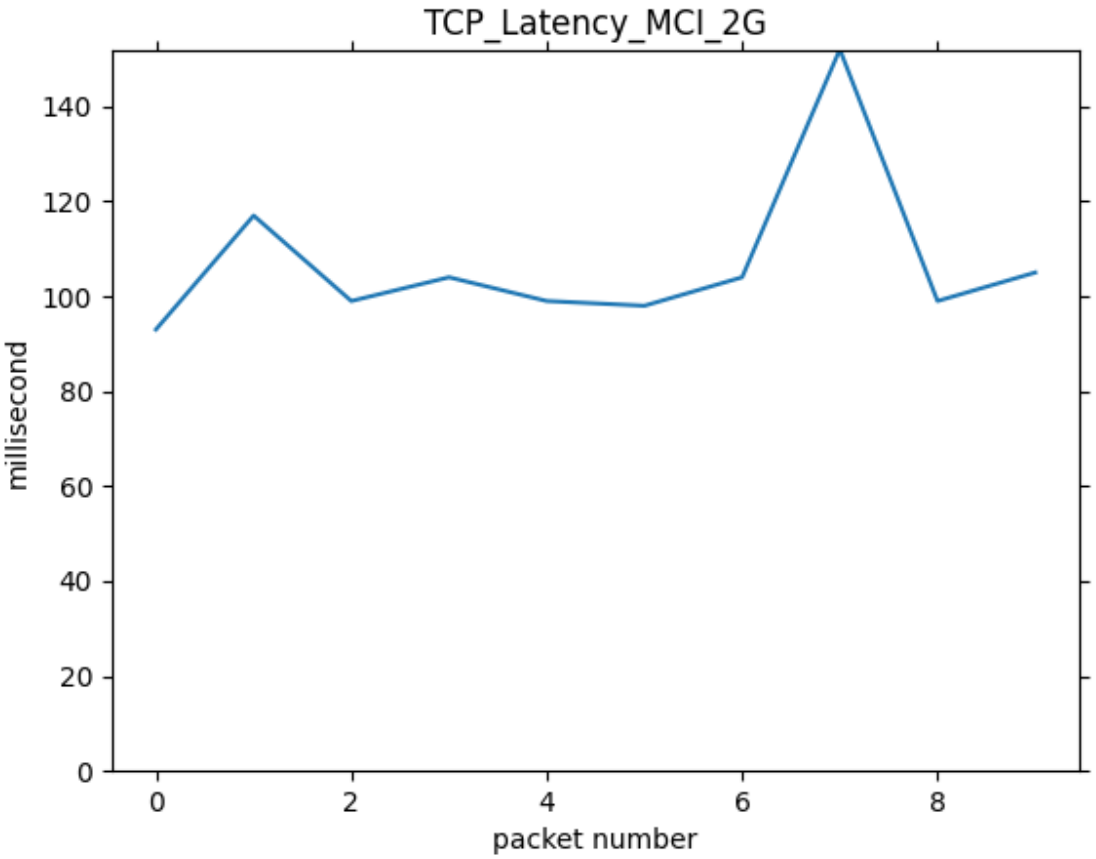


۶،۲،۳. اینترنت نسل ۲

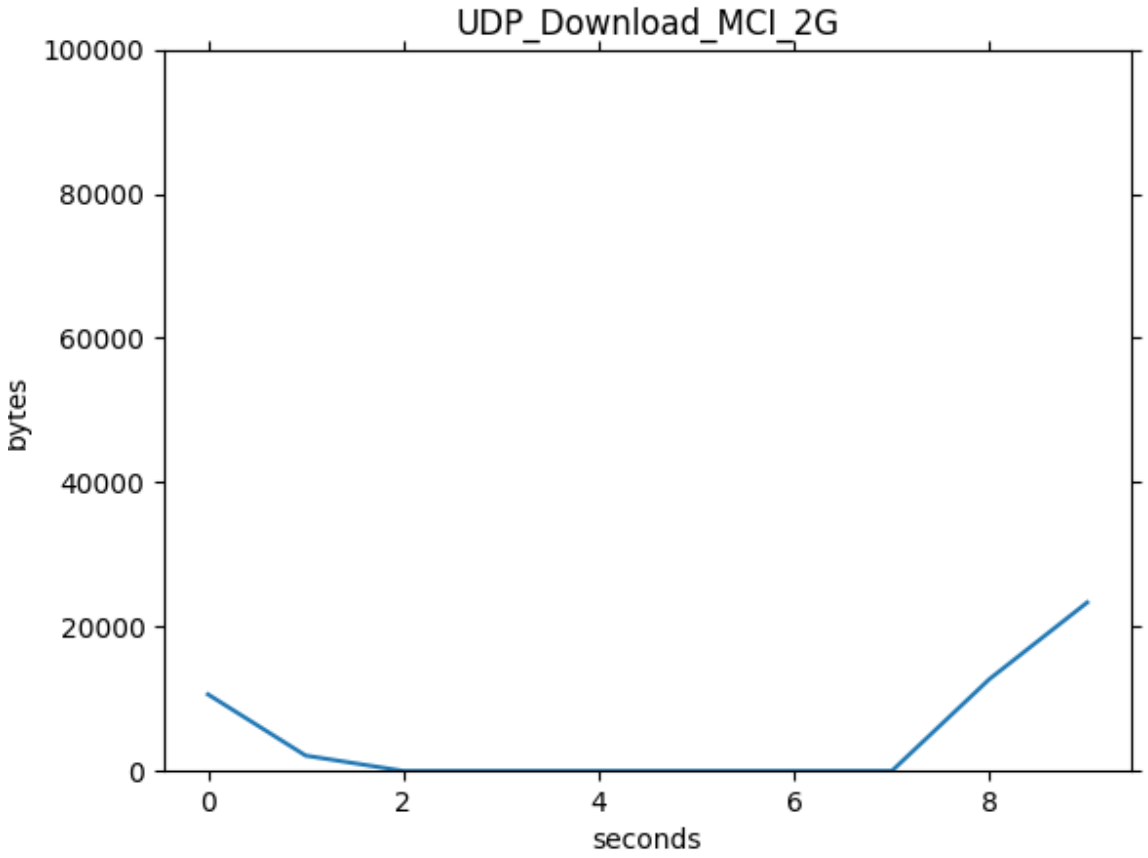
نمودارهای پروتکل TCP را در بخش زیر مشاهده می‌کنیم. میانگین سرعت آپلود و دانلود بسیار کم (حدود ۰ و ۲ کیلوبایت) و تاخیر نیز ۱۲۰ میلی‌ثانیه است.

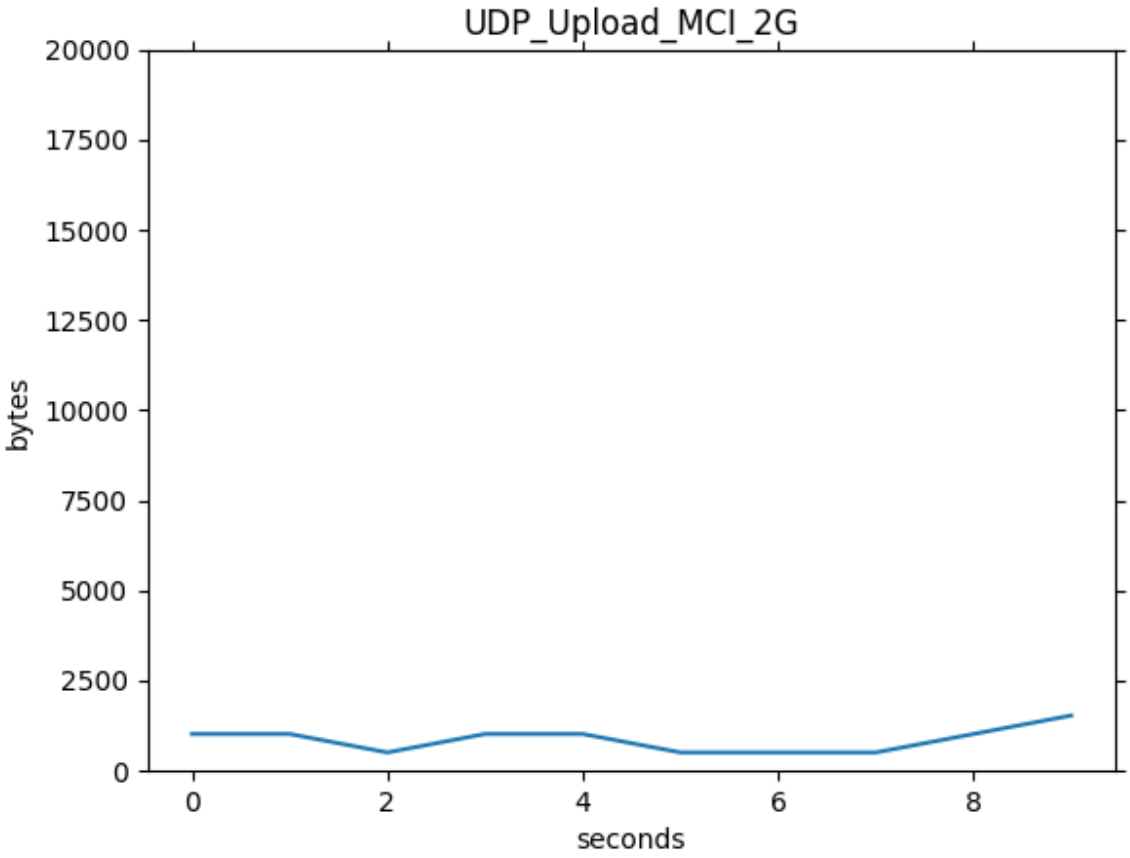

TCP_Download_MCI_2G
100000
80000
60000
40000
20000
0
bytes
0
2
4
6
8
seconds

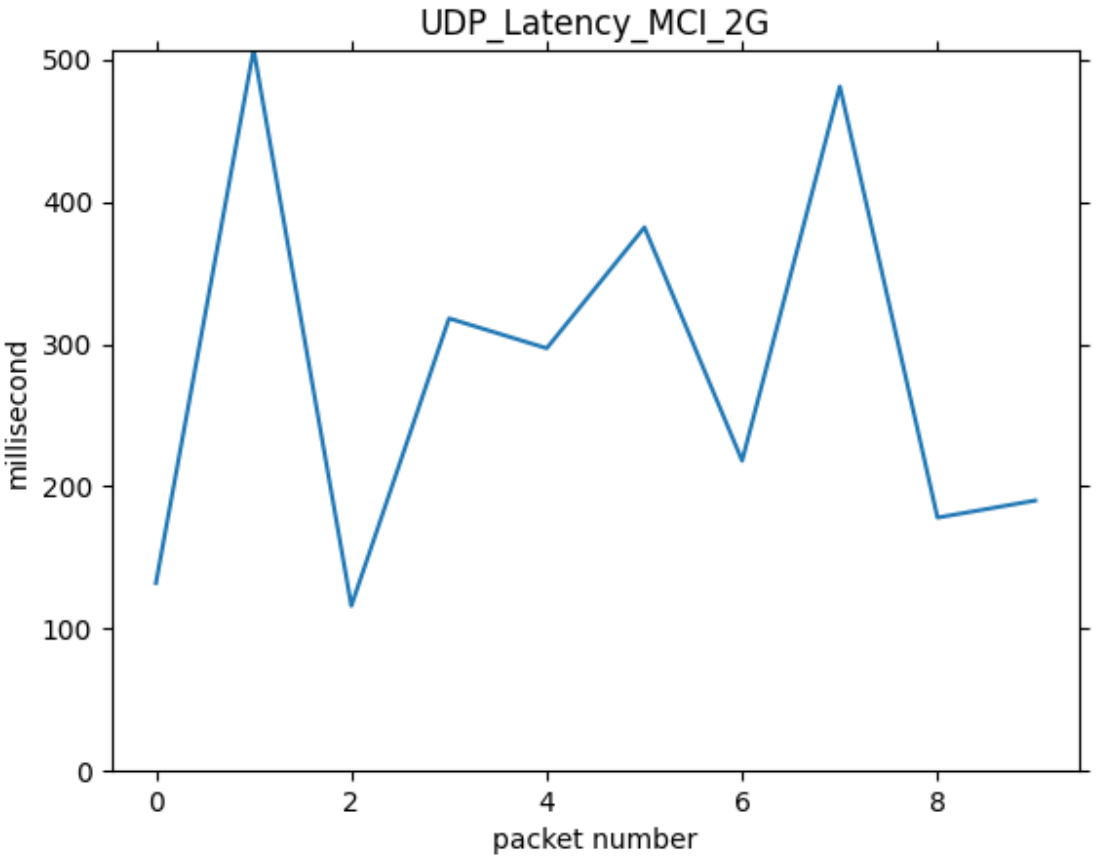

TCP_Upload_MCI_2G
20000
17500
15000
12500
10000
7500
5000
2500
0
bytes
0
2
4
6
8
seconds



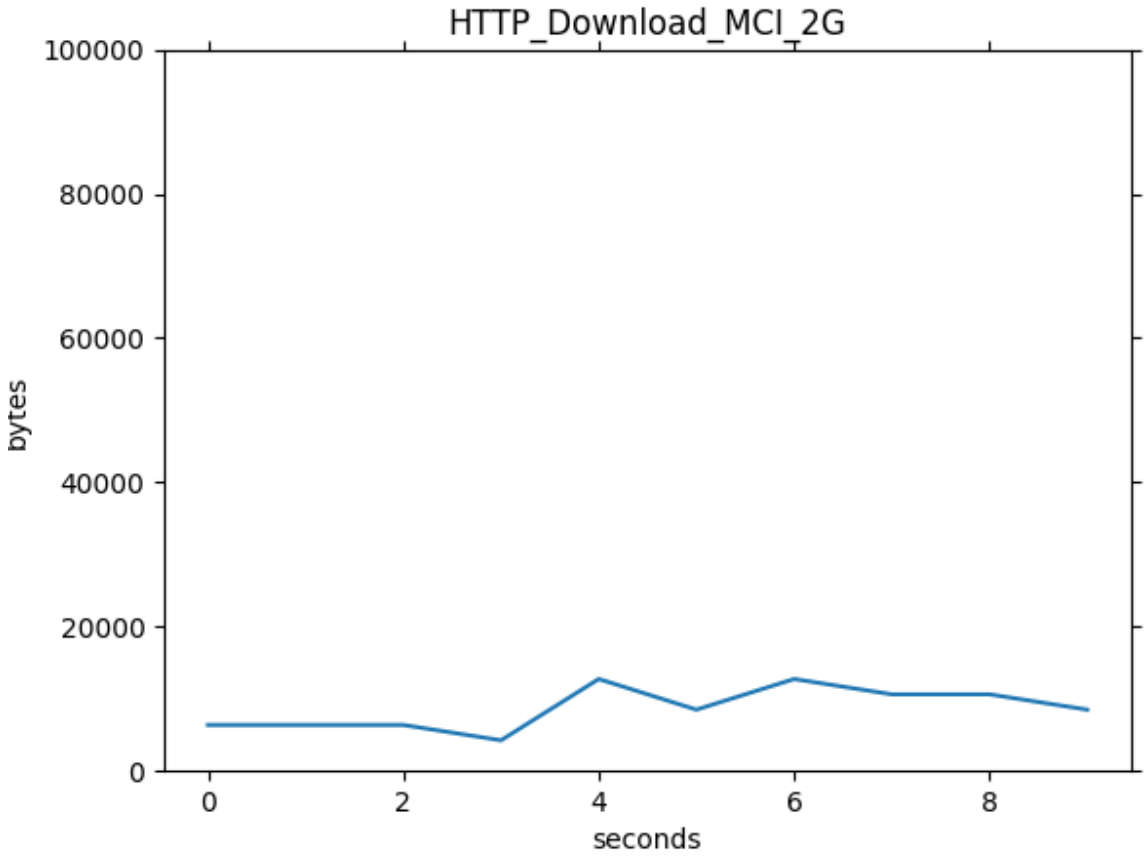
نمودارهای UDP این نسل را در زیر مشاهده می‌کنید. میانگین دانلود حدود ۱۰ کیلوبایت، آپلود ۱ کیلوبایت و تاخیر حدود ۳۵۰ میلی‌ثانیه است.

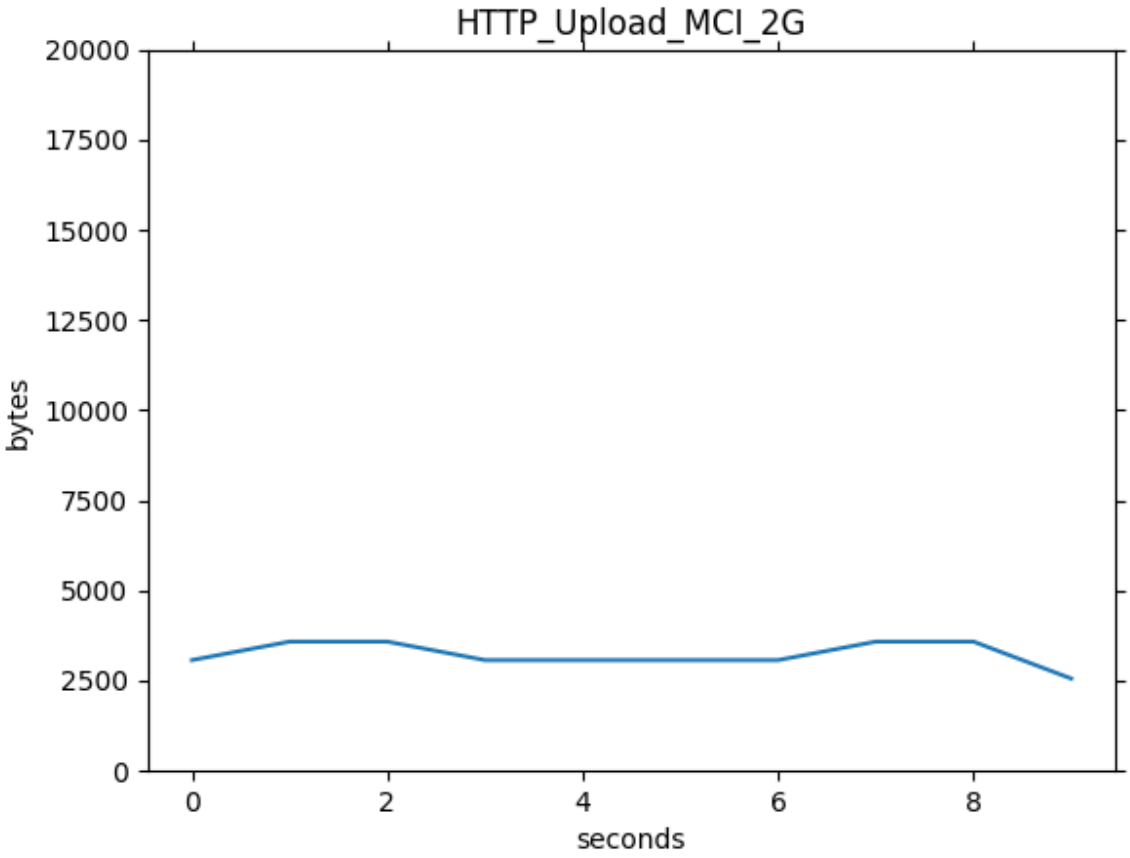

UDP_Download_MCI_2G
100000
80000
60000
40000
20000
0
bytes
0
2
4
6
8
seconds

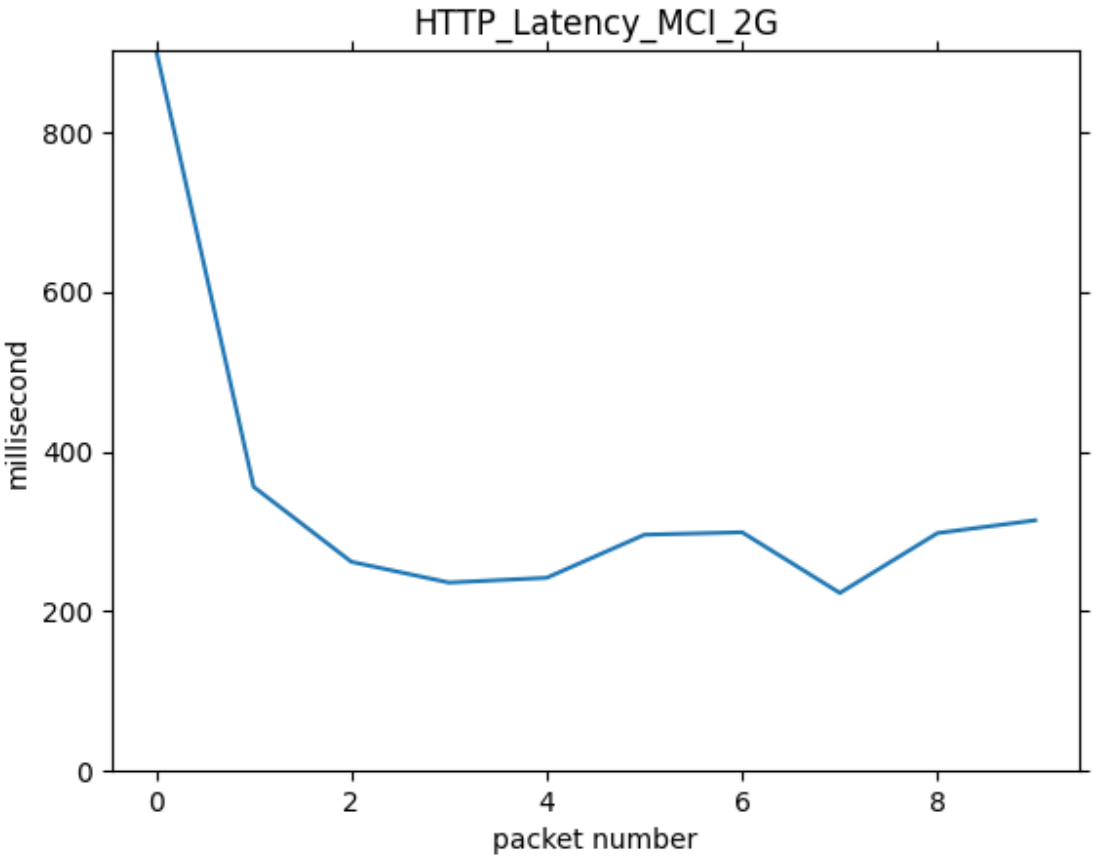

UDP_Upload_MCI_2G
20000
17500
15000
12500
10000
7500
5000
2500
0
bytes
0
2
4
6
8
seconds



نمودارهای HTTP این نسل را در زیر مشاهده می‌کنید. میانگین دانلود ۵ کیلوبایت، آپلود ۵/۲ کیلوبایت و تاخیر حدود ۵۰۰ میلی ثانیه است.

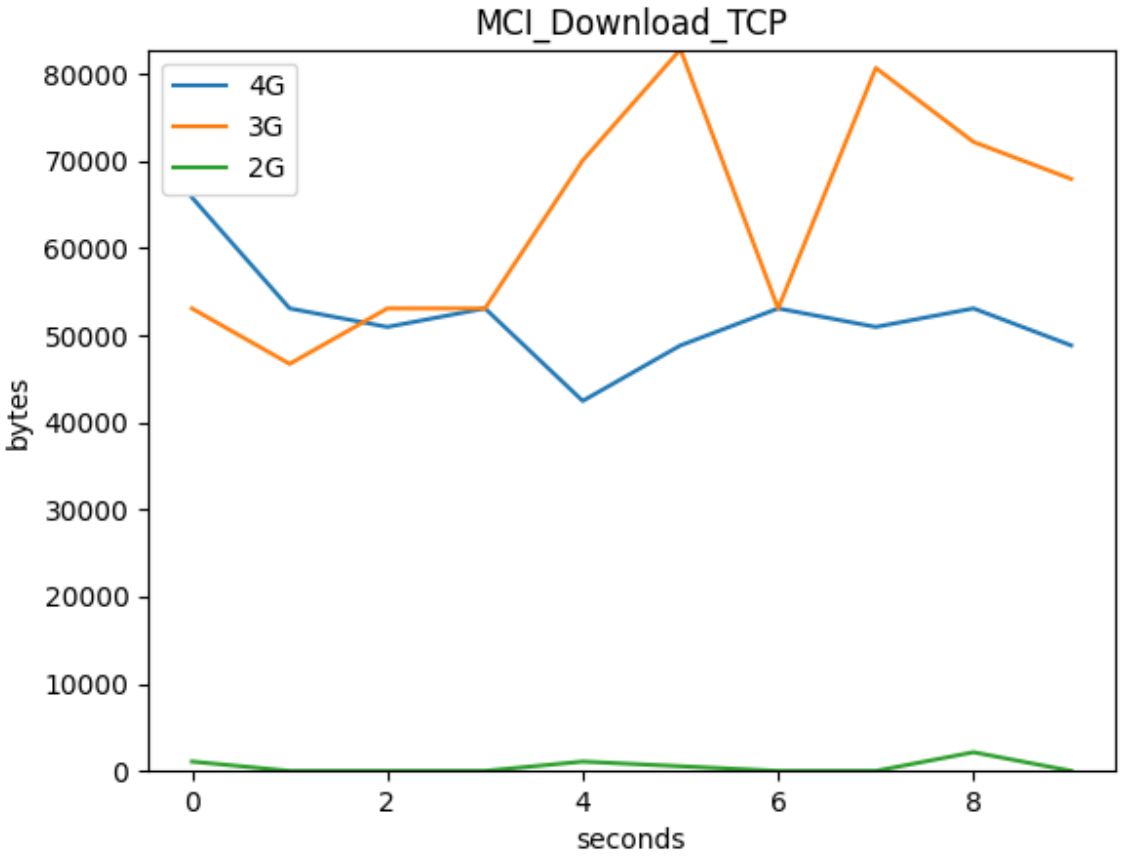

HTTP_Download_MCI_2G
100000
80000
60000
40000
20000
0
bytes
0
2
4
6
8
seconds


HTTP_Upload_MCI_2G
20000
17500
15000
12500
10000
7500
5000
2500
0
bytes
0
2
4
6
8
seconds

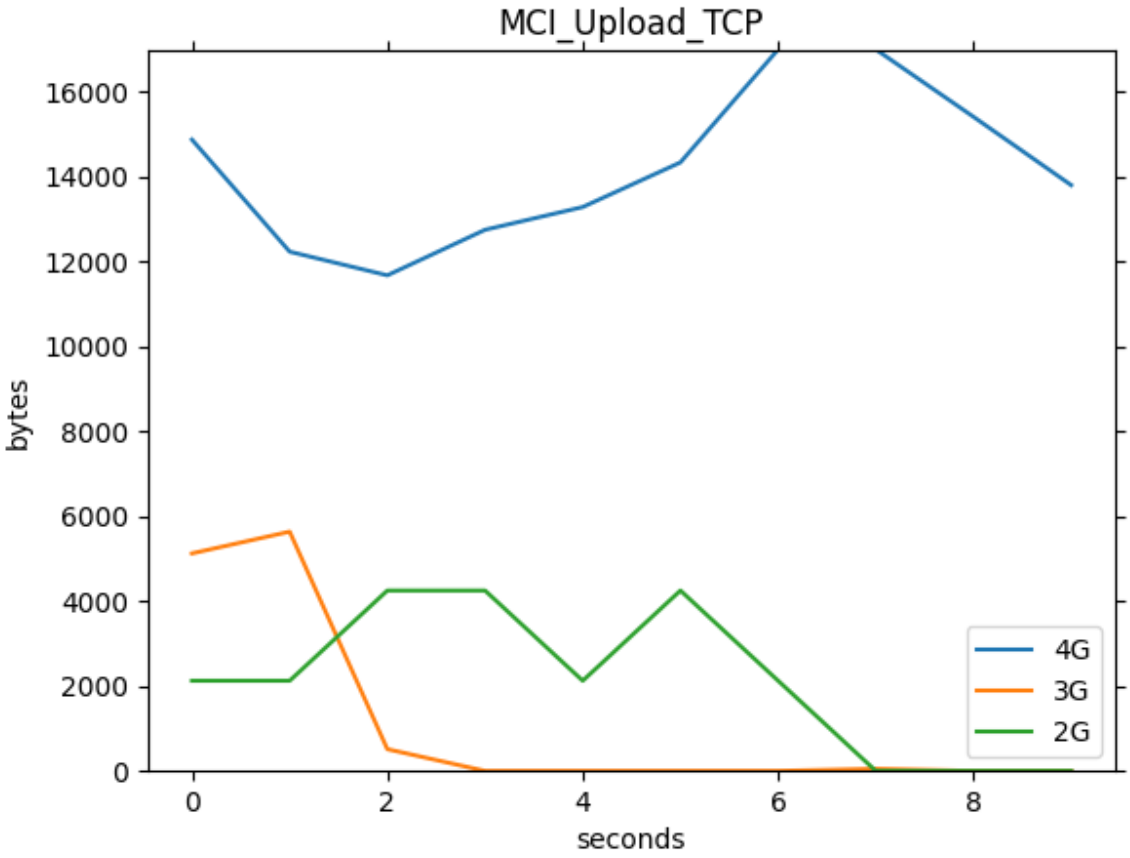


۶,۲,۴. مقایسه نسل‌های مختلف

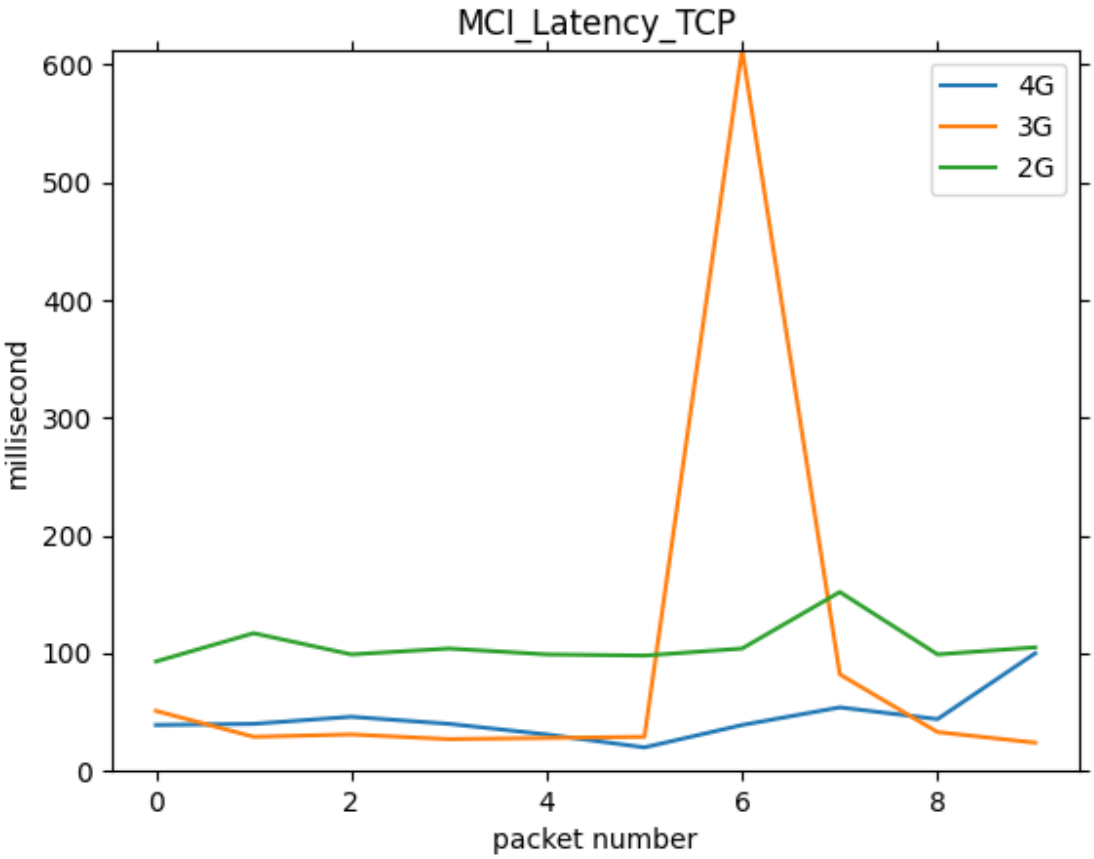
سرعت دانلود TCP را در سه نسل مختلف در نمودار زیر مشاهده می‌کنید.



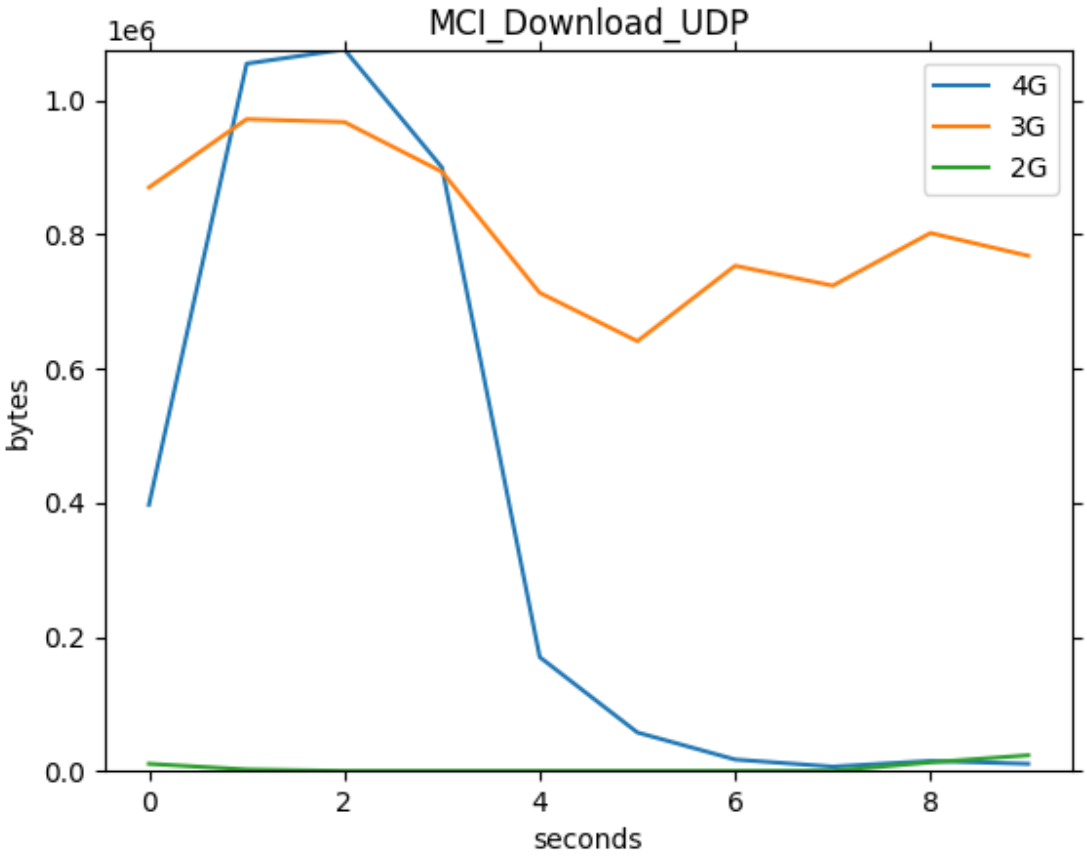
سرعت دانلود در نسل سوم و چهارم به شدت نسبت به نسل دوم افزایش پیدا کرده است.



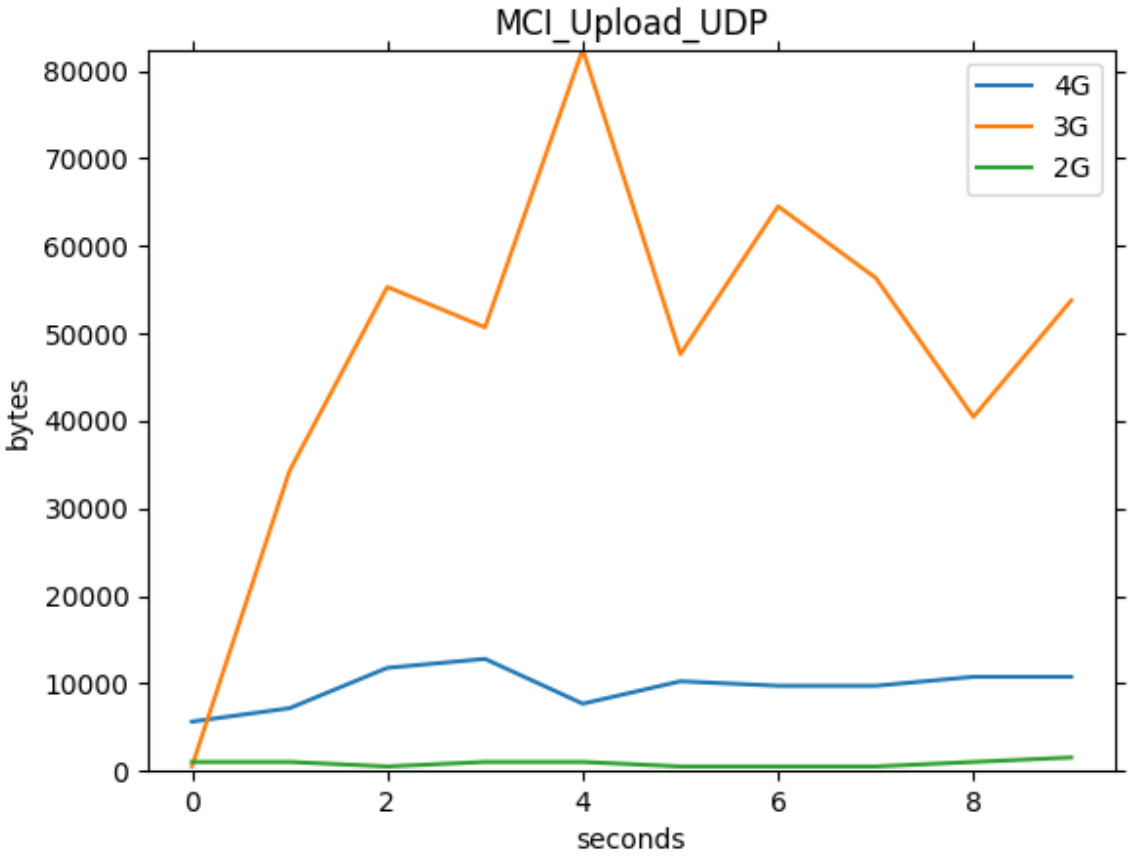
سرعت آپلود را با استفاده از روش TCP در نمودار بالا مشاهده می‌کنید. نسل چهارم، افزایش چشمگیری در سرعت آپلود داشته است اما نسل دوم و سوم در حدود هم هستند.



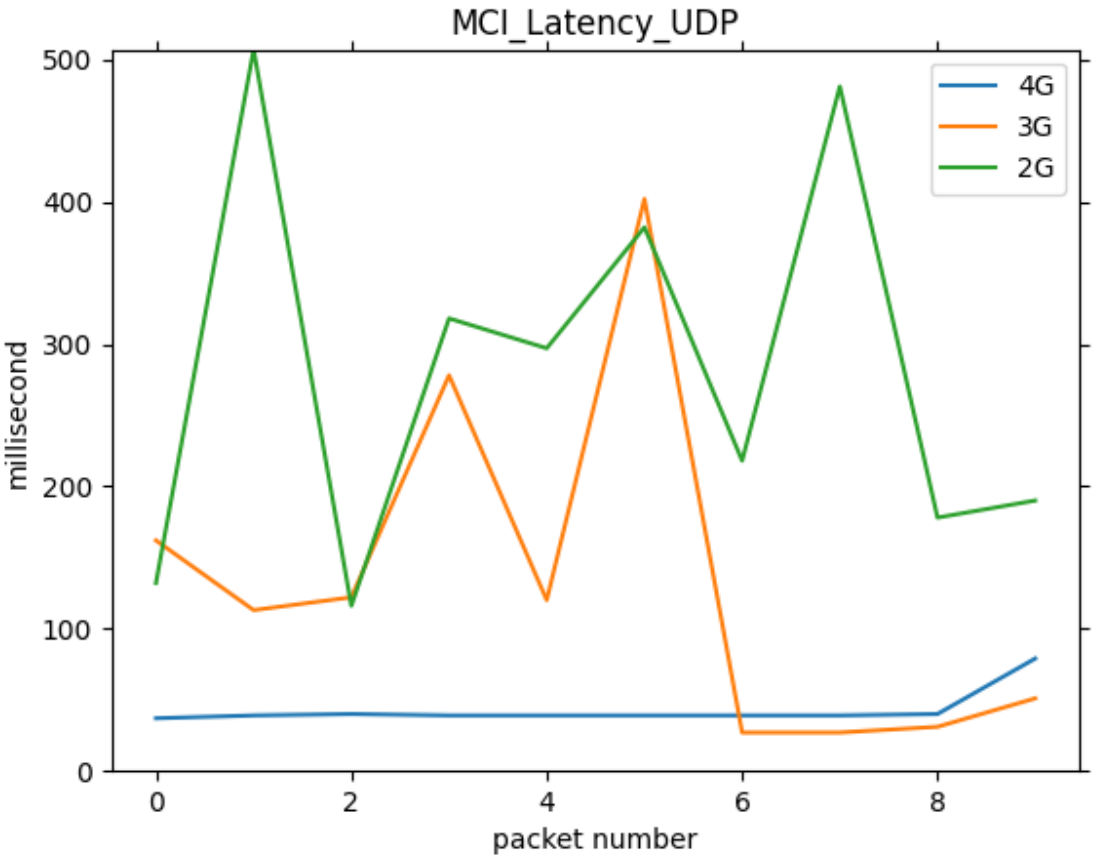
تاخیر سه نسل مختلف را با استفاده از روش TCP در نمودار بالا مشاهده می‌کنید. کمترین تاخیر مربوط به نسل چهارم است.



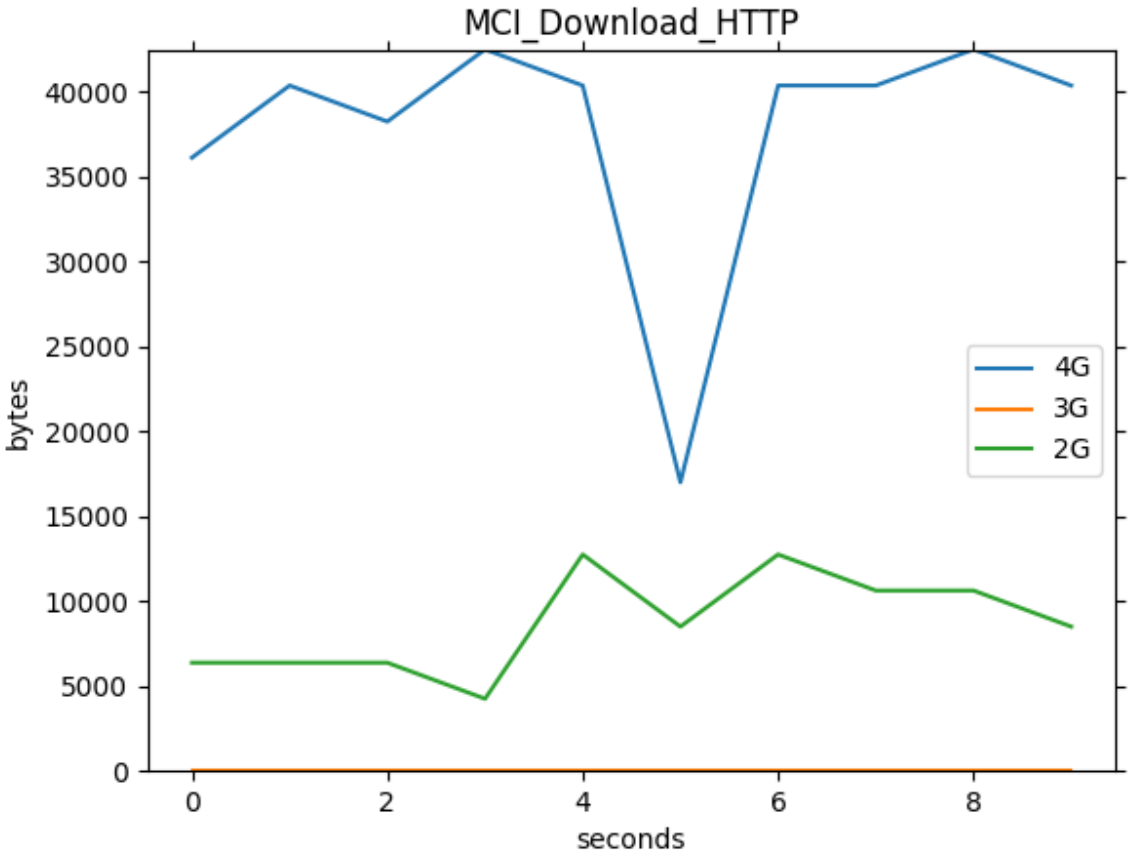
مقایسه سرعت دانلود در سه نسل مختلف با استفاده از پروتکل UDP در نمودار بالا انجام شده است. مشابه TCP، افزایش سرعت دانلود نسل سوم و چهارم نسبت به نسل دوم مشهود است.



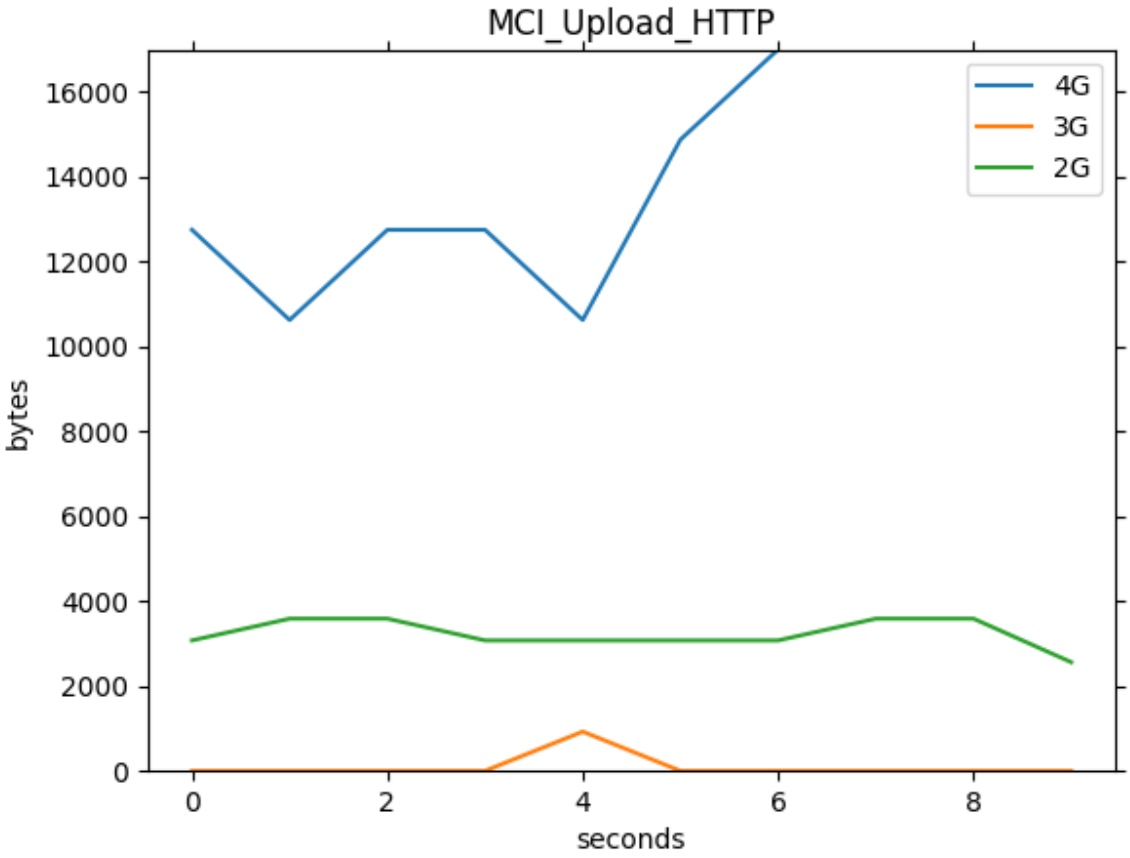
سرعت آپلود پروتکل UDP در نسل سوم در بیشترین مقدار خود است و در نسل چهارم کاهش یافته است.



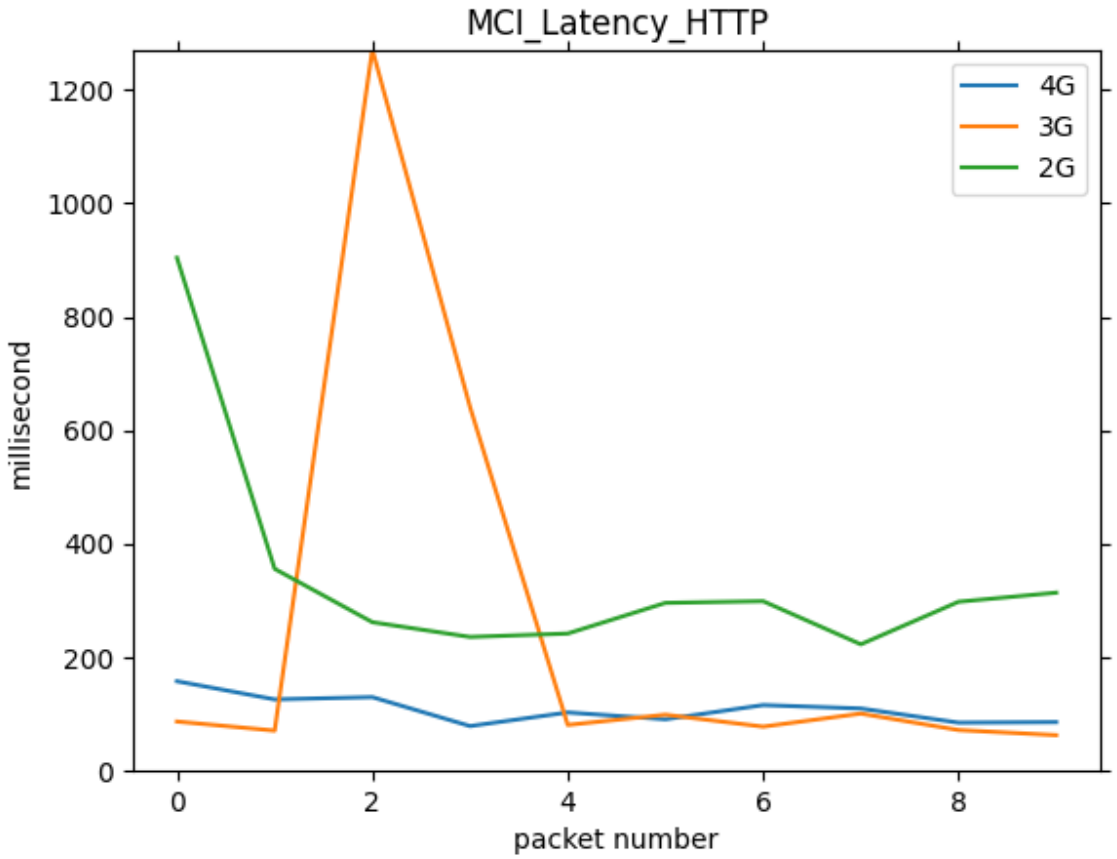
مقایسه تاخیر را در تصویر بالا برای پروتکل UDP مشاهده می‌کنید. مشابه پروتکل قبل، کمترین تاخیر مربوط به نسل چهارم است.



سرعت دانلود HTTP در نسل چهارم، با اختلاف از دو نسل قبلی بهتر است.



سرعت آپلود HTTP نیز در نسل چهارم پیشرفت زیادی داشته است.

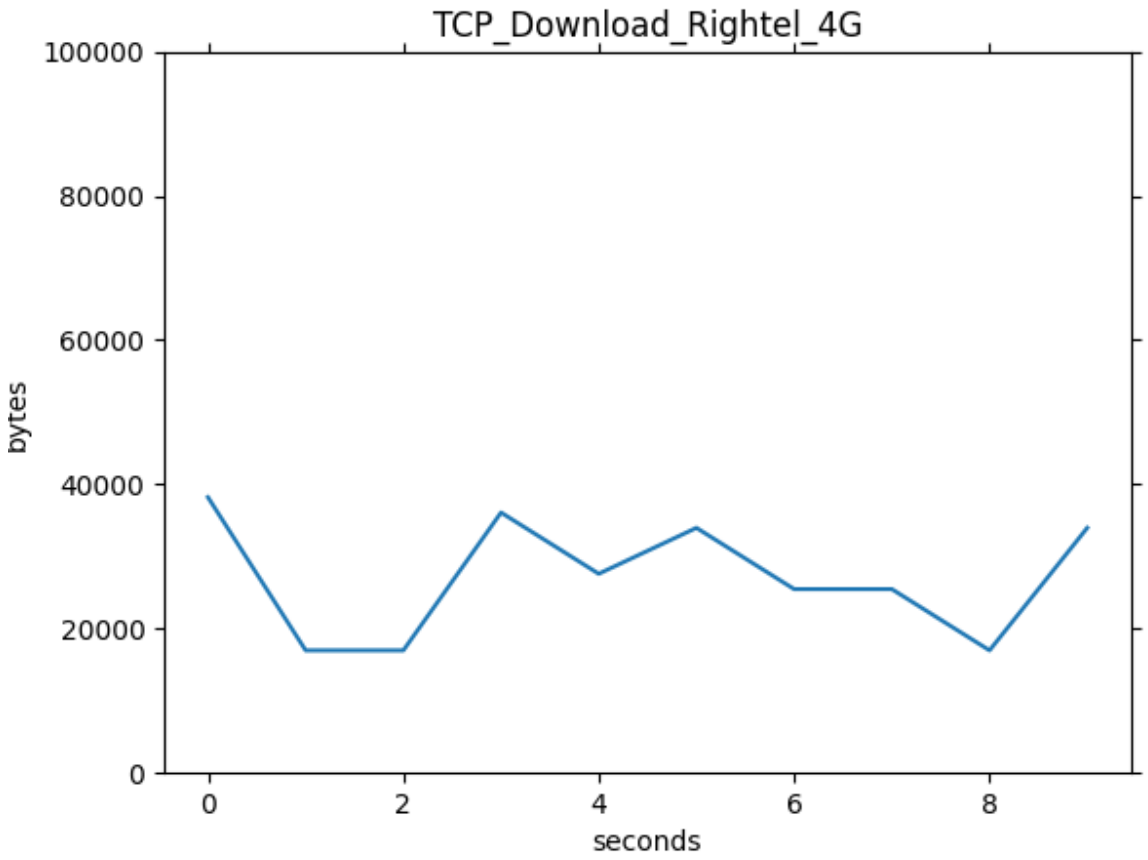


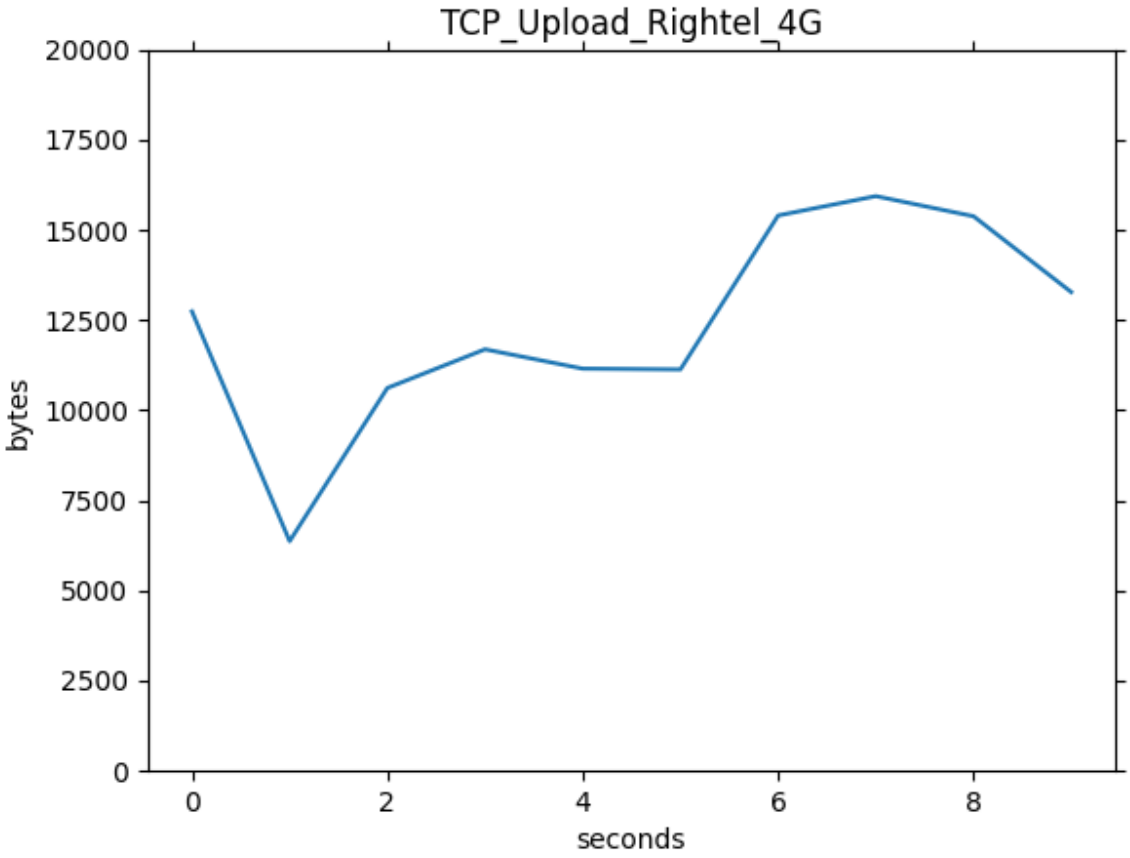
مشابه دو پروتکل قبل، برای HTTP نیز کمترین تاخیر در نسل چهارم ثبت شده است.

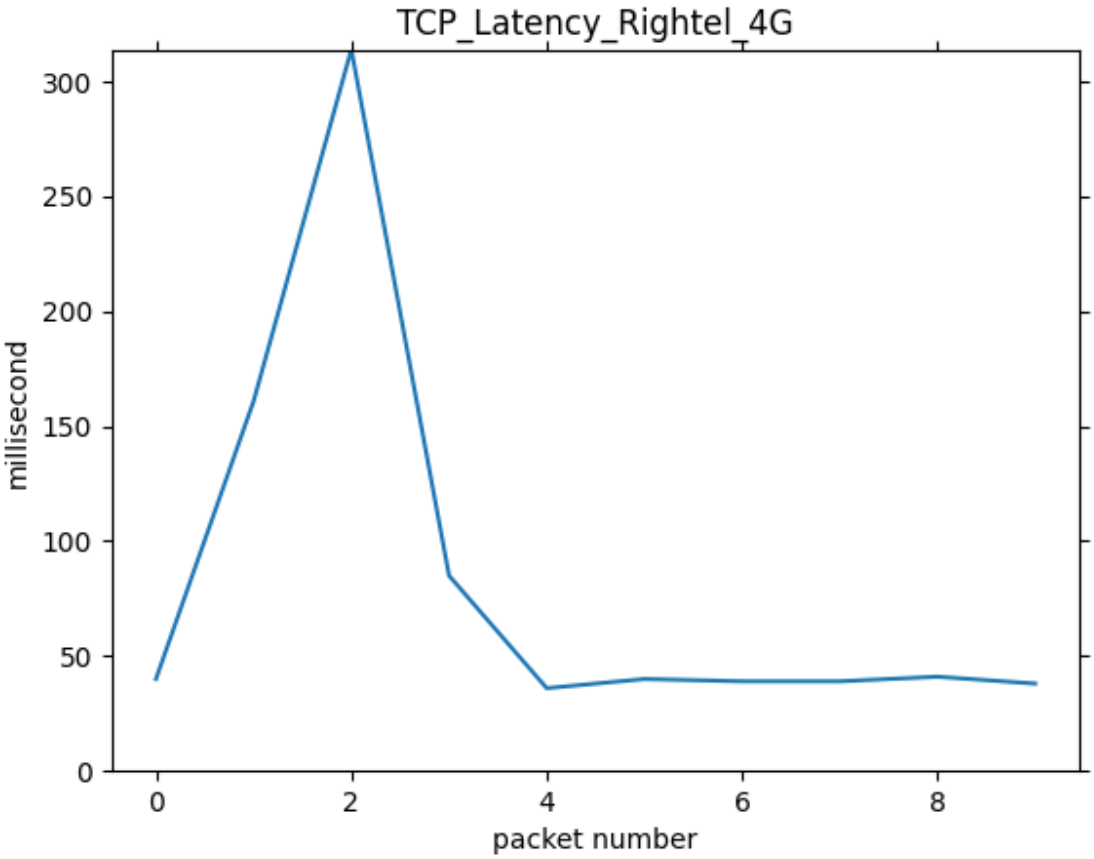
۶,۳. ارائه دهنده رایتل

۶,۳,۱. اینترنت نسل ۴

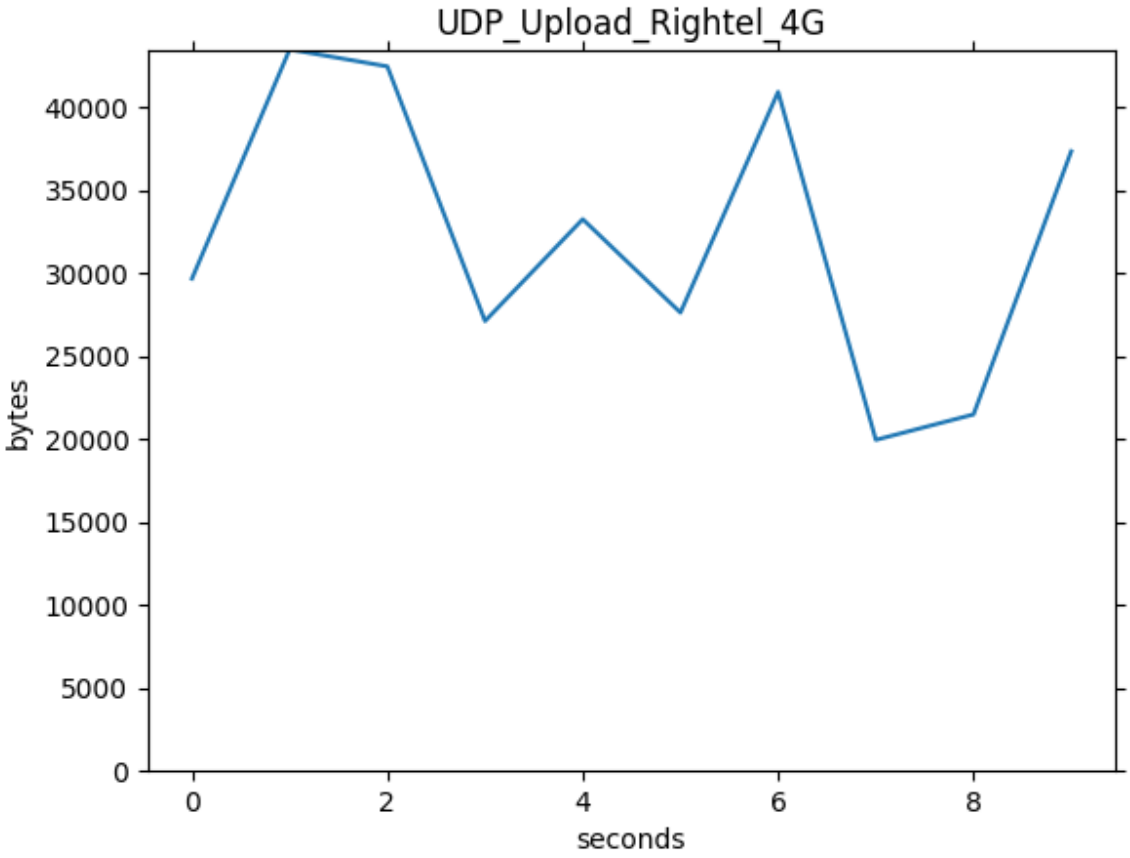
نمودارهای پروتکل TCP در زیر آمده است. میانگین دانلود ۳۰ کیلوبایت، آپلود ۱۲ کیلوبایت و تاخیر ۱۵۰ میلی‌ثانیه است.


TCP_Download_Rightel_4G
100000
80000
60000
40000
20000
0
bytes
0
2
4
6
8
seconds

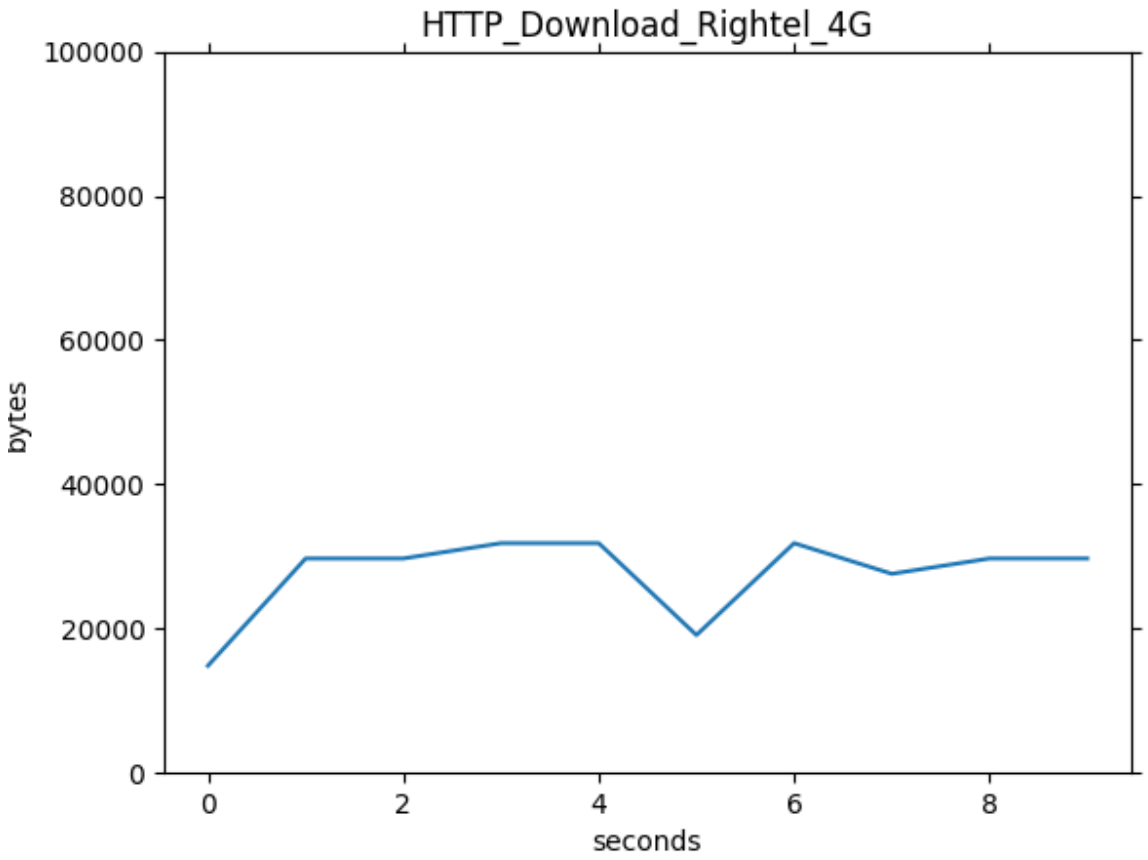

TCP_Upload_Rightel_4G
20000
17500
15000
12500
10000
7500
5000
2500
0
bytes
0
2
4
6
8
seconds

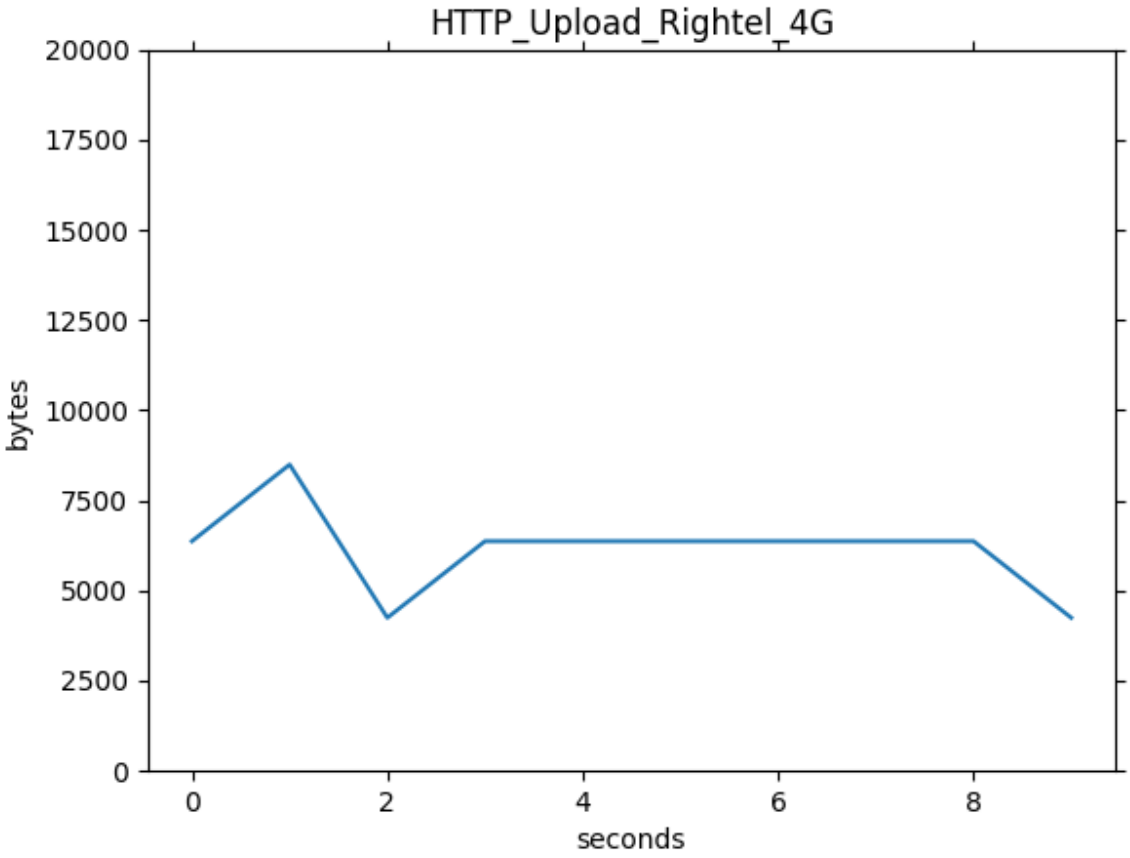


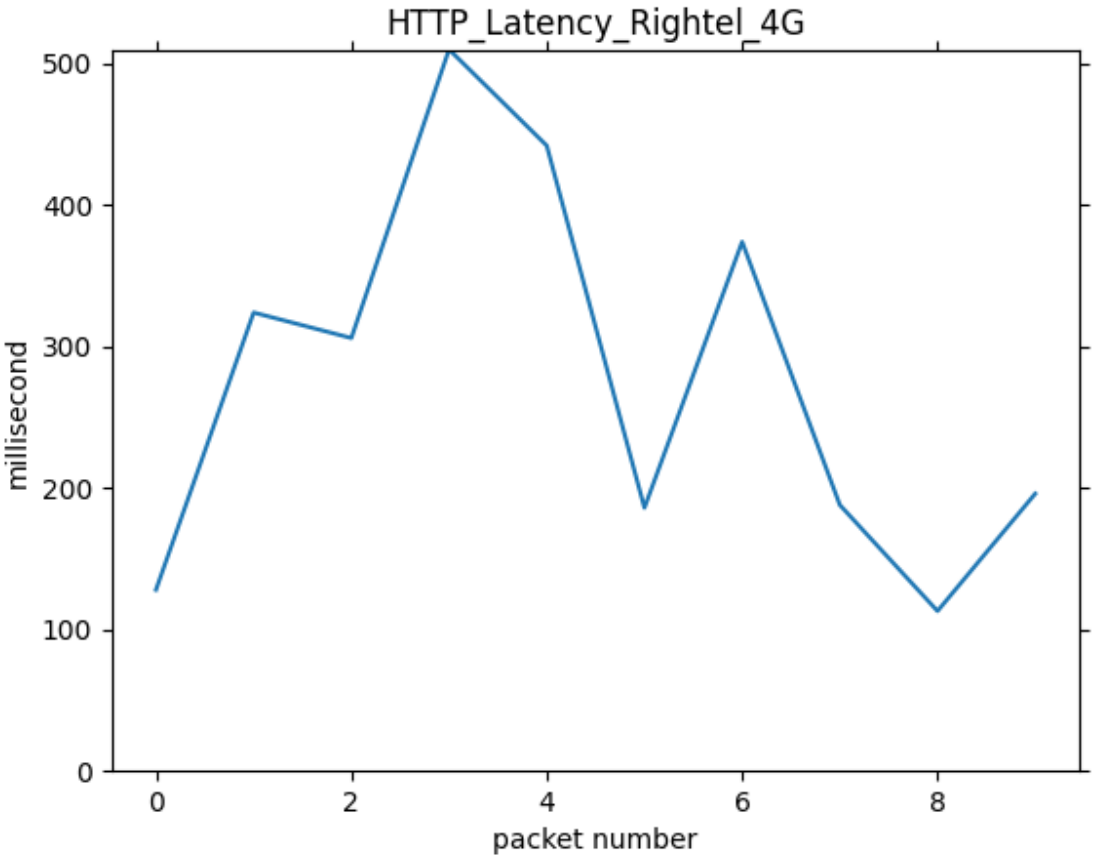
نمودارهای پروتکل UDP در زیر آمده است. این ارایه‌دهنده از دانلود با استفاده از پروتکل UDP پشتیبانی نمی‌کند بنابراین تنها اطلاعات آپلود در دسترس است. میانگین آپلود با استفاده از این پروتکل ۳۰ کیلوبایت است.



نمودارهای زیر مربوط به روش HTTP هستند. میانگین دانلود ۲۵ کیلوبایت، آپلود ۶ کیلوبایت و تاخیر ۳۰۰ میلی ثانیه می‌باشد.

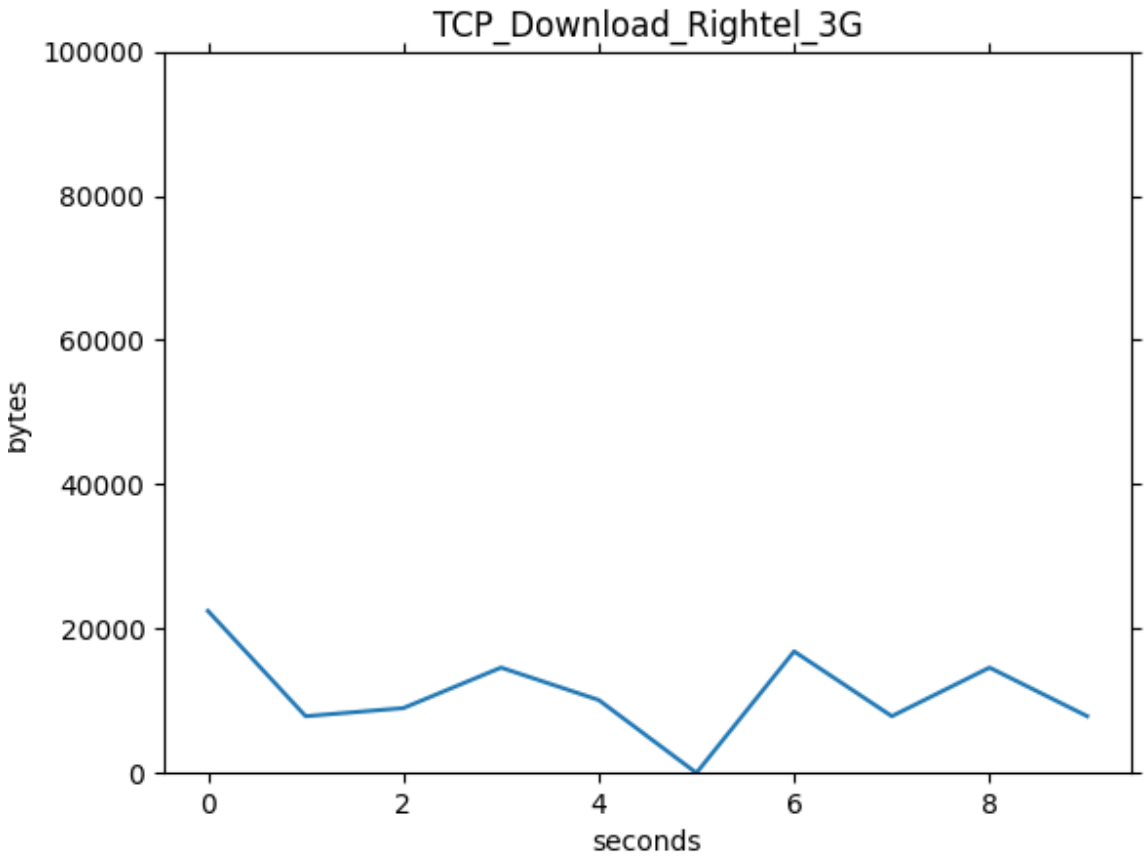

HTTP_Download_Rightel_4G
100000
80000
60000
40000
20000
0
bytes
0
2
4
6
8
seconds

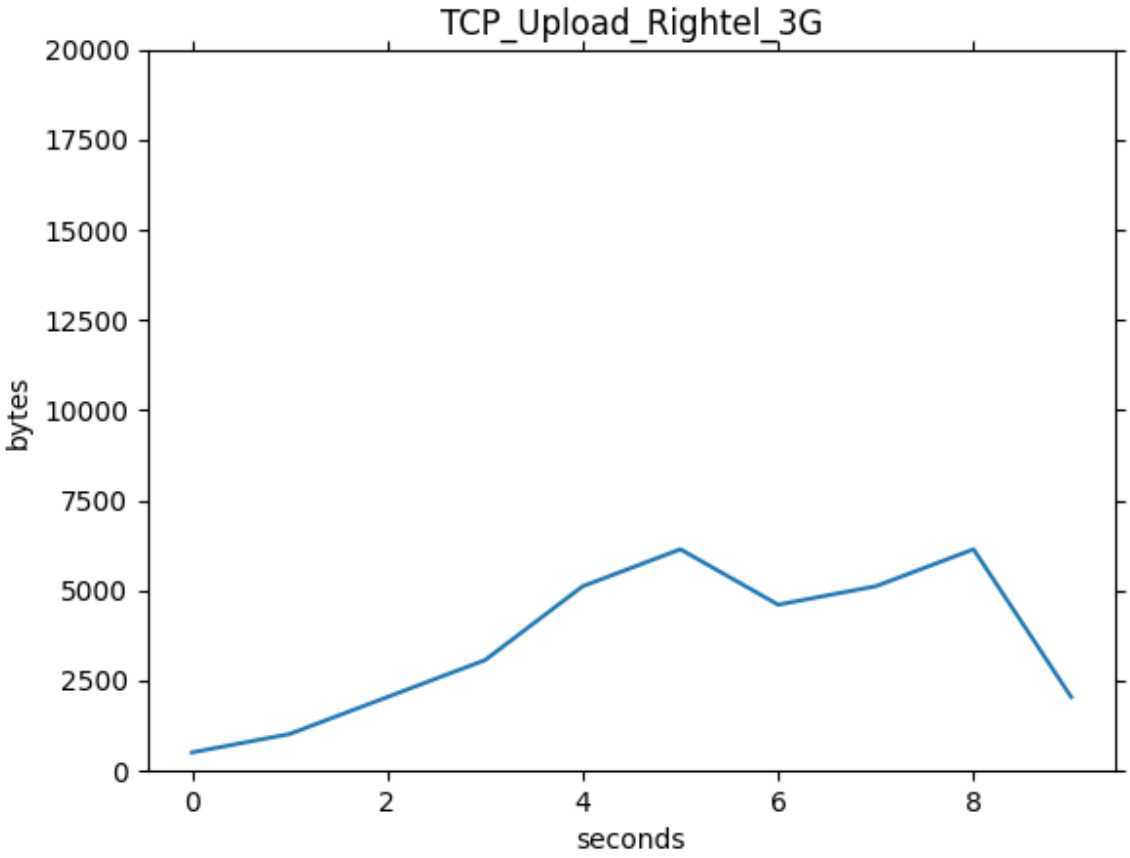

HTTP_Upload_Rightel_4G
20000
17500
15000
12500
10000
7500
5000
2500
0
bytes
0
2
4
6
8
seconds

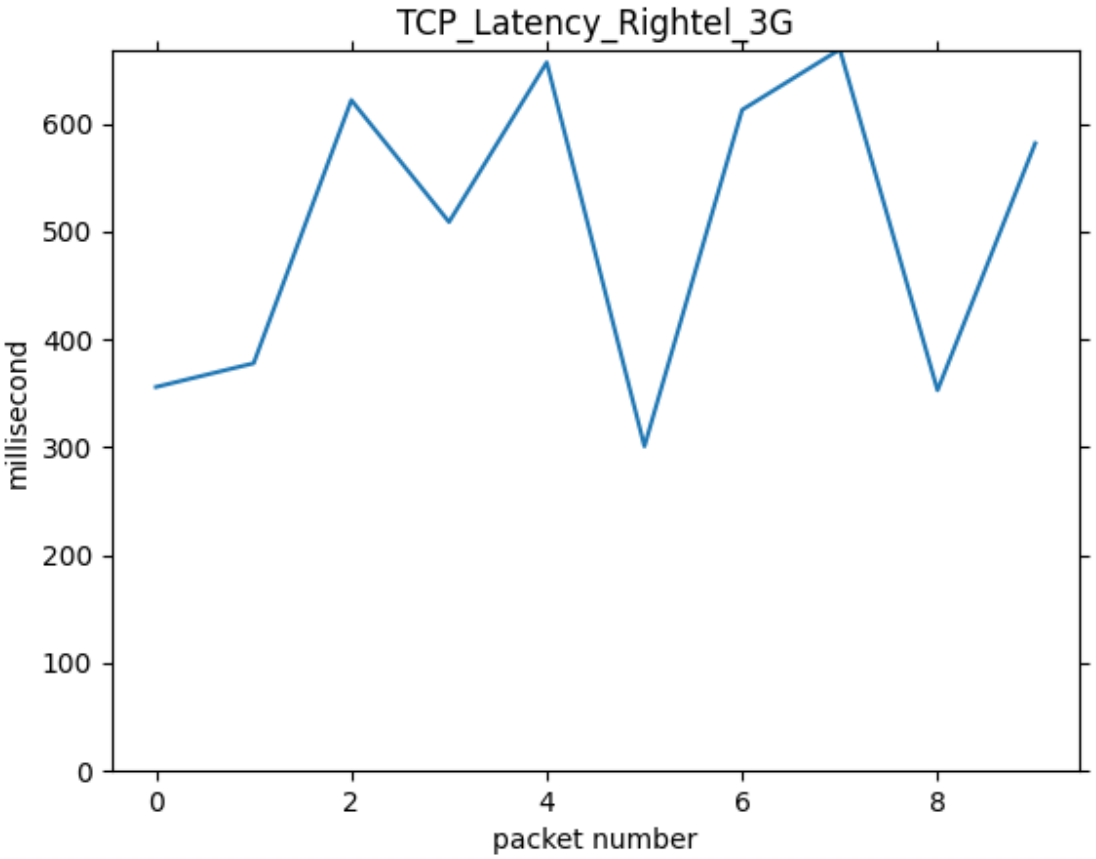


۶،۳،۲. اینترنت نسل ۳

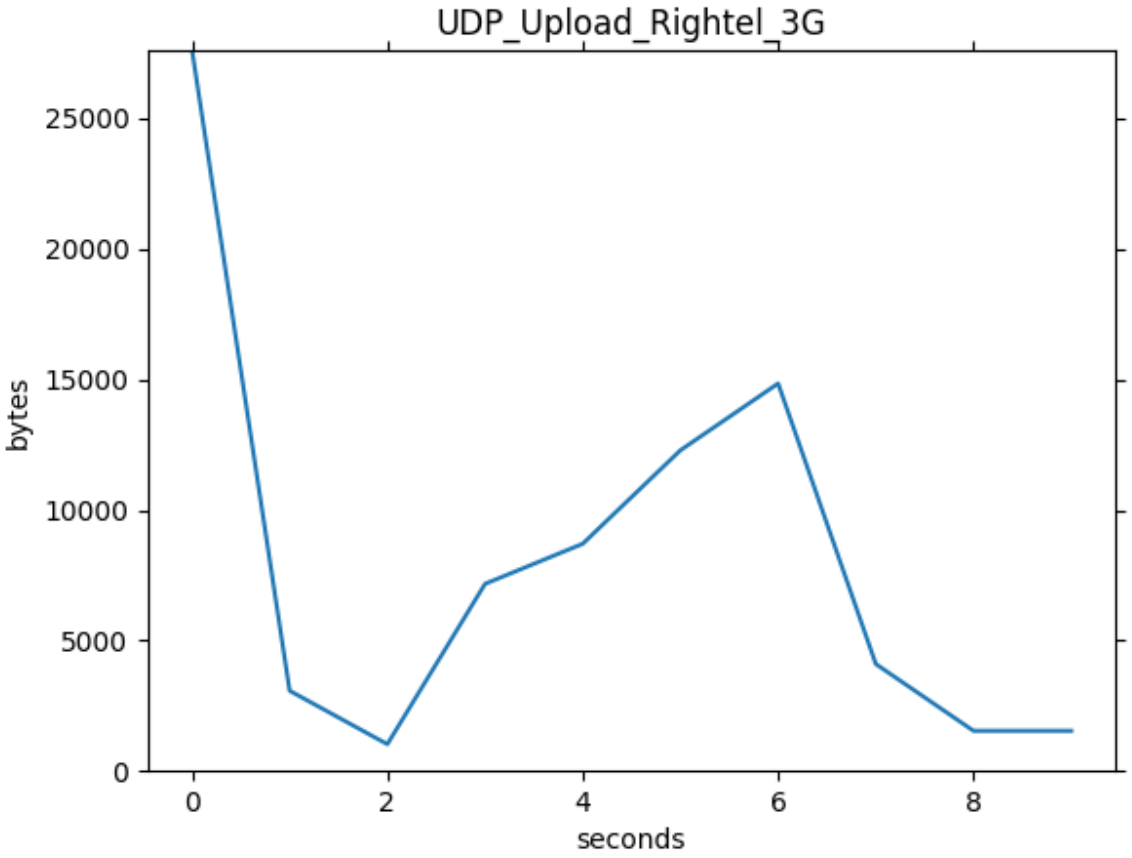
نمودارهای زیر برای پروتکل TCP هستند. میانگین دانلود ۱۰ کیلوبایت، آپلود ۴ کیلوبایت و تاخیر ۵۰۰ میلی ثانیه می‌باشد.


TCP_Download_Rightel_3G
100000
80000
60000
40000
20000
0
bytes
0
2
4
6
8
seconds

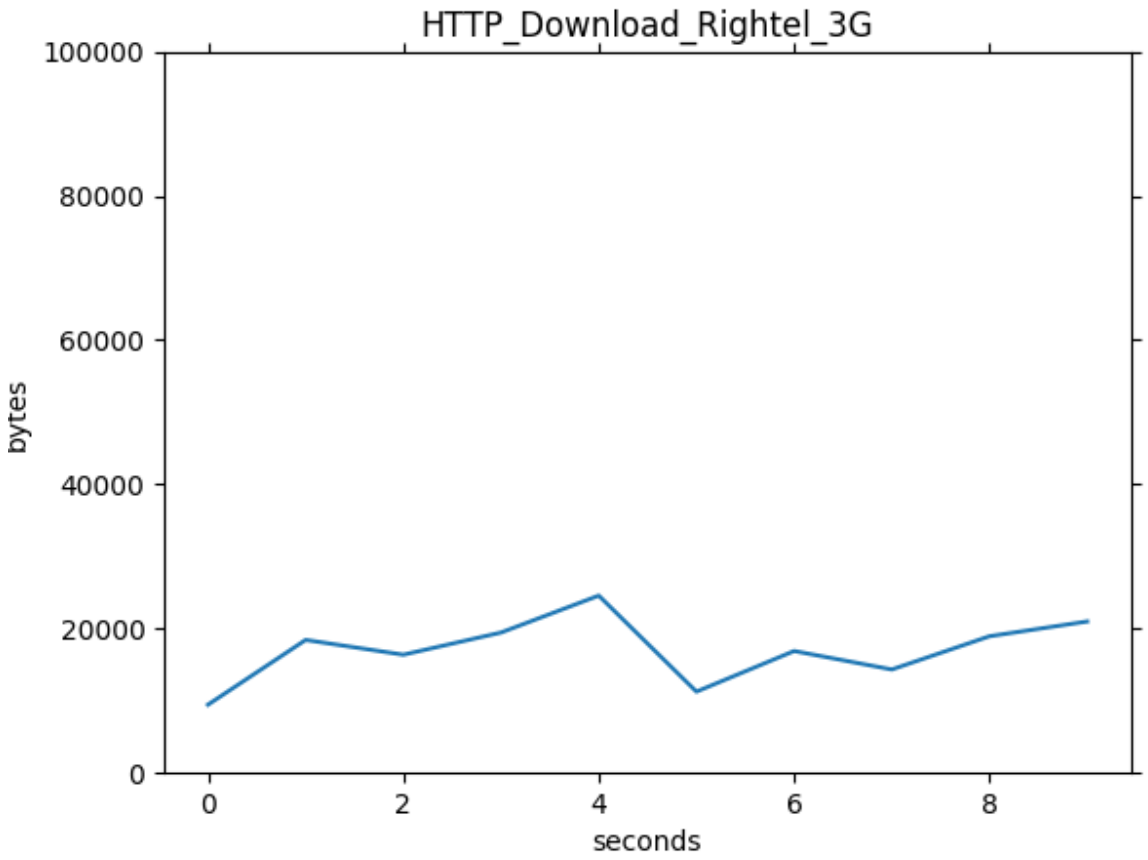

TCP_Upload_Rightel_3G
20000
17500
15000
12500
10000
7500
5000
2500
0
bytes
0
2
4
6
8
seconds

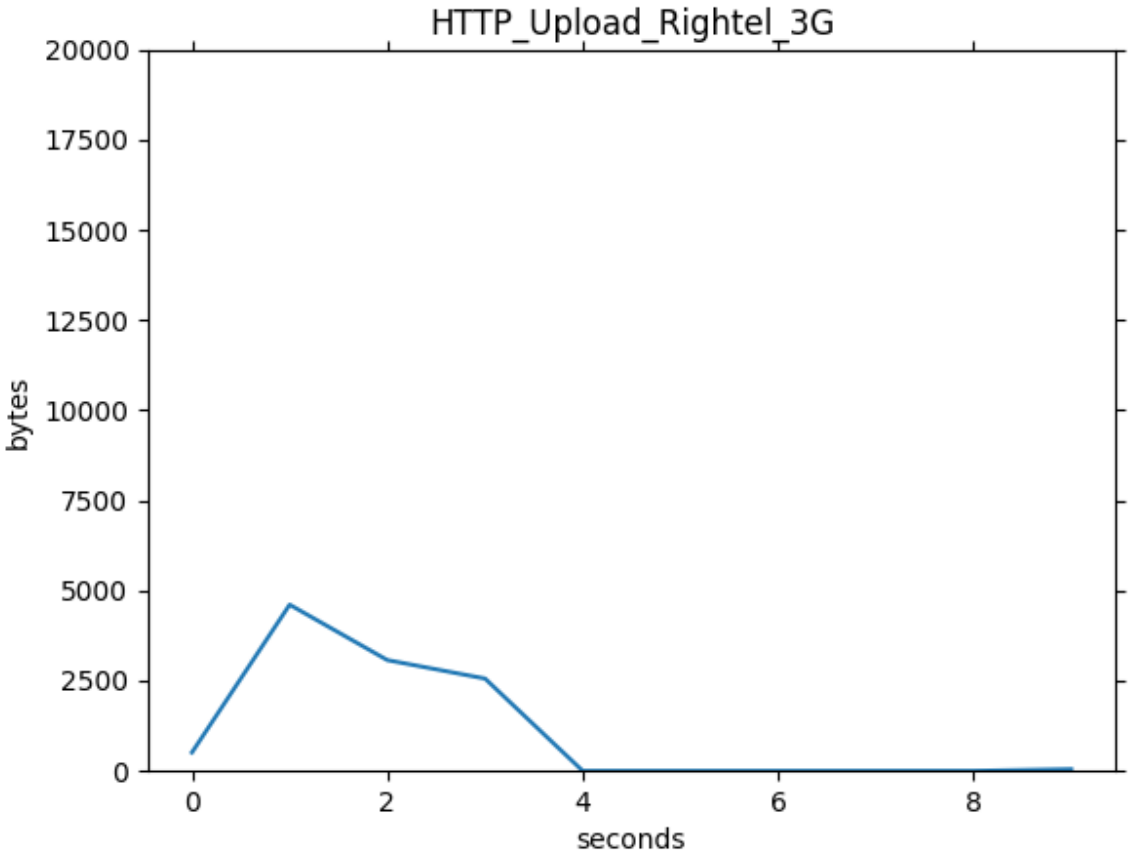


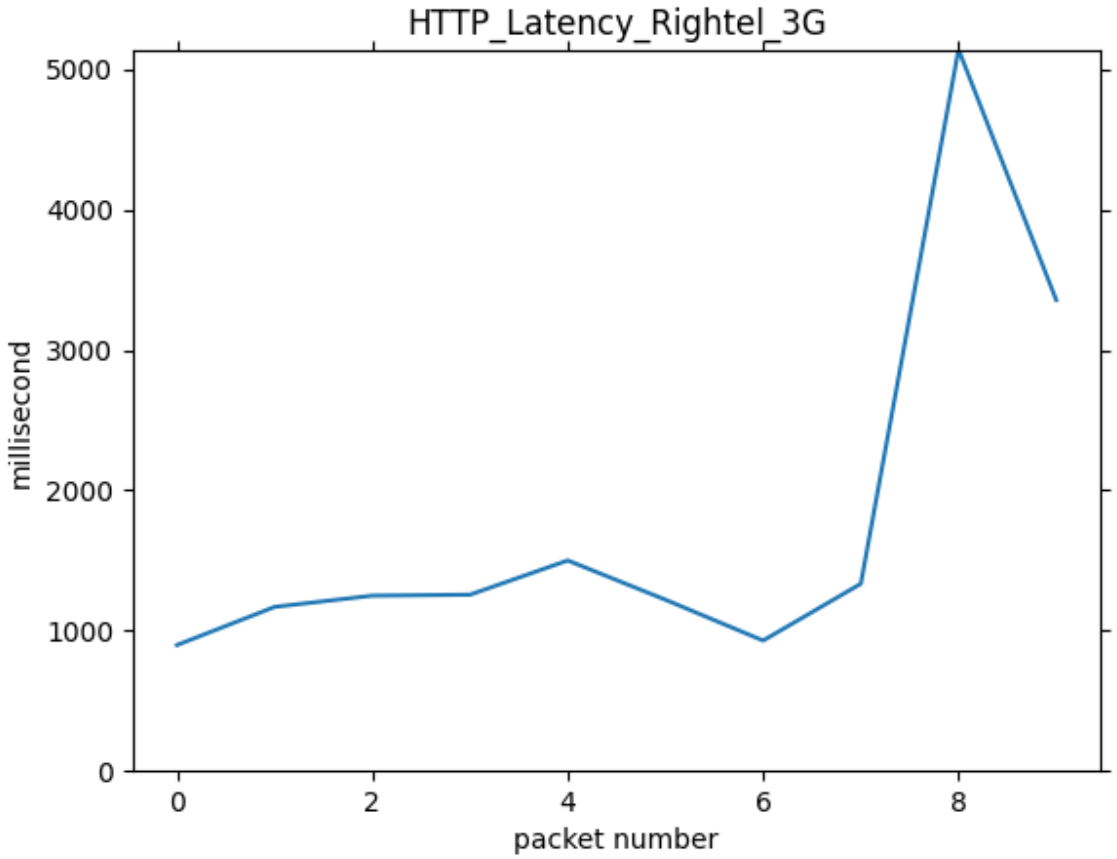
نمودار زیر، آپلود UDP است. میانگین سرعت آپلود با این روش ۷ کیلوبایت است.



نمودارهای زیر مربوط به HTTP است. میانگین دانلود ۱۵ کیلوبایت، آپلود ۲ کیلوبایت و تاخیر بیشتر از تمامی حالتهای دیگر ۲۰۰۰ میلی ثانیه است.


HTTP_Download_Rightel_3G
100000
80000
60000
40000
20000
0
bytes
0
2
4
6
8
seconds


HTTP_Upload_Rightel_3G
20000
17500
15000
12500
10000
7500
5000
2500
0
bytes
0
2
4
6
8
seconds


HTTP_Latency_Rightel_3G
millisecond
packet number
0
1000
2000
3000
4000
5000
0
2
4
6
8