DChannel: Accelerating Mobile Applications With Parallel High-bandwidth and Low-latency Channels

Presenter: Yunxiang Chi
10/04/2024

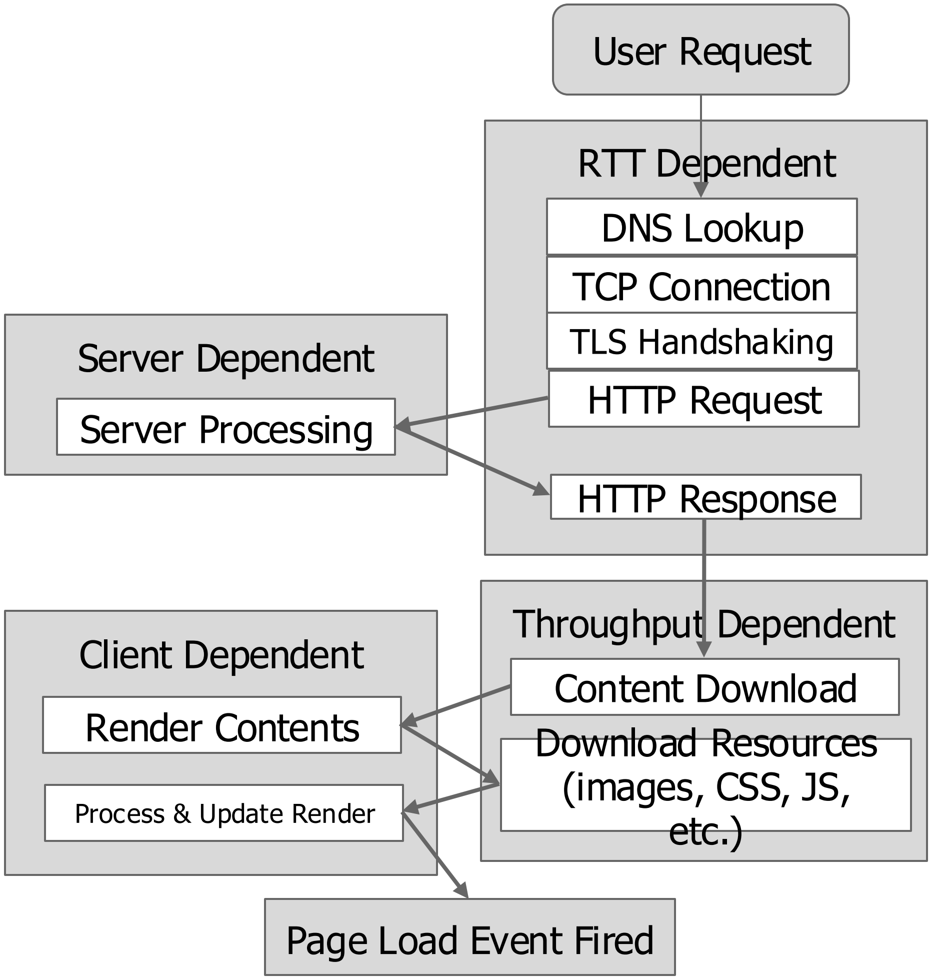
Background

- Real-time Interactive applications:
  - E.g. web browsing, virtual and augmented reality, and cloud gaming.
    - "An increase of 100 ms latency can result in as much as 1% revenue loss"[1]
    - "VR requires 20 ms or lower latency to avoid any simulator sickness"[1]
    - "Cloud gaming requires at most 96ms to ensure normal experience"[2]

Source: [1] nsdi23_slides_sentosa
[2] Zhuge: https://dl.acm.org/doi/10.1145/3544216.3544225

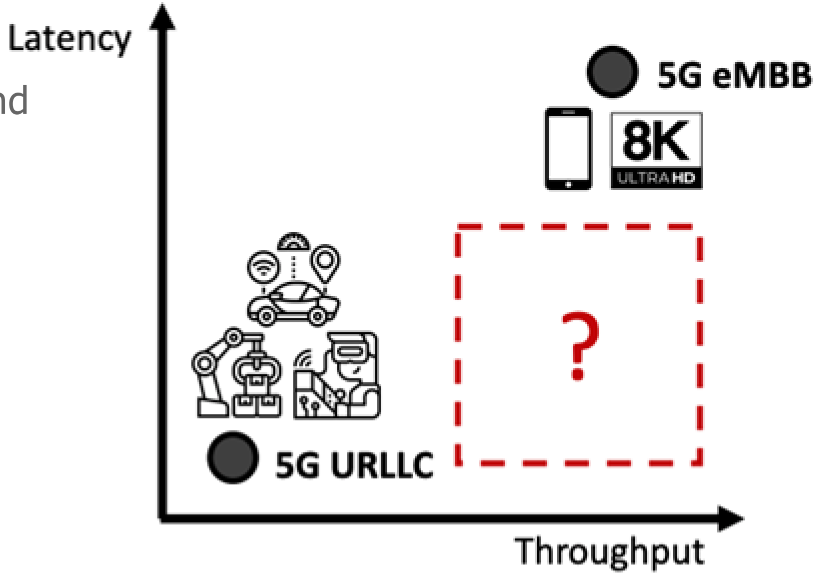
Background - PLT

- Web pages consist of many small objects from multiple servers.
- Web browsing generates short, bursty flows.
- Page Load Time (PLT) is heavily influenced by RTT, not just throughput.
- Increasing TCP throughput beyond ~16 Mbps has little impact on PLT.



Background - continued

- Enhanced Mobile Broadband (eMBB)
  - Operate at sub6G(<6GHz) and mmWave(around 28 GHz/39 GHz) range
  - High throughput(~2Gbps) and high and inconsistent latency
- Ultra-reliable low-latency communication (URLLC)
  - Operate at sub6G(<6GHz) range
  - Highly reliable, low latency, low throughput
    - 0.4-16Mbps
    - 2-10ms E2E latency



Source: [1] nsdi23_slides_sentosa

Can we break the latency throughput tradeoff?

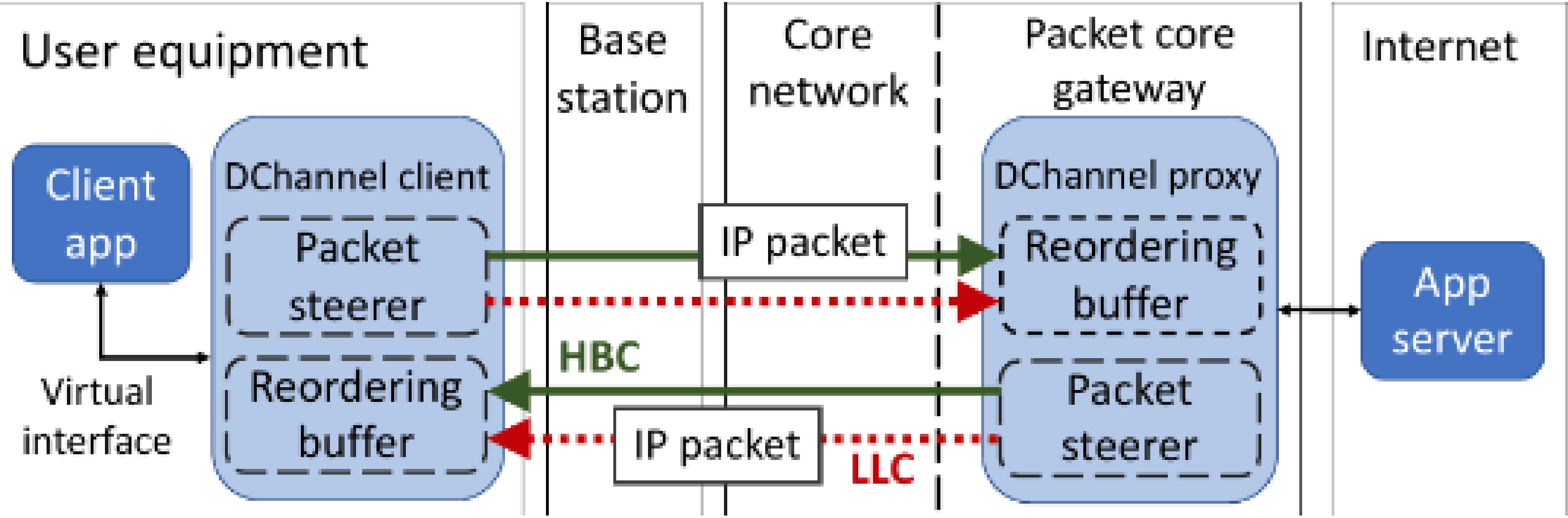
Insights

- Using high bandwidth channel(HBC) and low latency channels(LLC) in parallel on mobile devices

Challenges:

- Need to use LLC’s bandwidth very selectively
- MPTCP: transport-layer mechanism to combine multiple channels; assumes two interfaces that are each of significant bandwidth. But LLC’s bandwidth is not comparable to significant bandwidth.
- Socket Intents and TAPS: exploit multi-access connectivity through application-level input; difficult deployment and low scalability.

DChannel


User equipment
Base station
Core network
Packet core gateway
Internet
Client app
DChannel client
Packet steerer
Reordering buffer
Virtual interface
IP packet
HBC
IP packet
LLC
DChannel proxy
Reordering buffer
Packet steerer
App server

Packet Steering Intuition

- Accelerate the "message"
  - A sequence of one or more pkts i.e. the receiving endpoint can take some useful action
    - E.g. SYN, ACK, HTTP request
- Rewards and Cost
  - Rewards: $R(P_n) = C_{n,LLC} - C_{n,HBC}$ $\quad C_{n,link} = max(C_{n-1}, (t_n + D_{link}(P_n)))$

    $D_{link}(P_n) = Dprop_{link} + (size(P_n) + Q_{link}(t_n))/B_{link}$
  - Cost: $F(P_n) = (size(P_n) + Q_{llc}(t_n))/B_{llc}$

    Pn: the useful msg to accelerate; Cn,link: pkt completion time on that link
    Dlink: delivery time for a pkt; Dprop_link: channel/link propagation delay
    Blink: channel/link bandwidth; Qlink: queue size; tn: timestamp
  - Comparing: $R(P_n) > \alpha F(P_n)$, where ɑ is a parameter that tries to capture the tradeoff that: the benefit is immediate but the cost affects pkts afterwards.

Estimate latency

Perform periodic handshakes (e.g., in every 500 ms)with UDP packets:
(1) client agent sends a “D-SYN” pkt to the proxy agent via HBC and LLC.
(2) proxy agent responds with “D-SYN/ACK” packets via HBC and LLC.
(3) client agent updates the base RTT value for both channels based on the difference between D-SYN/ACK receive time and D-SYN release time, and replies with “D-ACK” via both channels.
(4) Proxy agent receives the D-ACK and updates the base RTT value for both.

Reordering Buffers

- Harmful: could be identified as a signal of congestion -> retransmission/sending rates drop
- At the receiving site of each agent, only buffer pkts from LLC
- Unbounded buffering delay:
  - If a packet expected to arrive via HBC is lost or severely delayed, packets in the ROB would keep waiting indefinitely.
  - Solution: The ROB releases packets after a set conservative timeout period of 100ms, even if the expected earlier packets haven't arrived.

Experiment Setup

Setup

- Test Environments:
  - Live-eMBB: Real 5G eMBB + Emulated URLLC (Ethernet)
  - Emulated-eMBB: Trace-driven emulation of both channels
- Collecting Network Traces:
  - Measured latency and throughput of eMBB channel over time
  - Latency: Periodic UDP probes (15ms intervals) from client to server
  - Bandwidth: Saturated uplink and downlink with MTU-sized UDP packets
  - Used separate devices for latency and bandwidth measurements to avoid interference
- Emulating Traces:
  - Used extended version of Mahimahi for emulation on a single machine
  - Latency: Modified delay shell to vary eMBB latency based on collected traces
  - Bandwidth: Extended link shell for time-varying bandwidth (1-second intervals)
  - URLLC: Emulated with 5ms propagation delay and 2Mbps bandwidth
  - Power states: Simulated UE sleep states and discontinuous reception
  - Queue: FIFO drop-tail queue with 800 MTU-sized packet buffer

Setup - continued

- Testbed Configuration
  - Live-eMBB: Laptop tethered to Google Pixel 5 phone
  - Locations: UIUC campus (5G low-band) and Chicago downtown (5G mmWave)
  - URLLC emulation: Wired link with 5ms RTT, 2Mbps capacity
- Application Use Cases
  - a. Web Browsing
    - 200 web pages from Hispar corpus
    - Live and emulated eMBB settings
  - b. Mobile Applications
    - Three Android apps: Reddit, eBay, CNN
    - Emulated eMBB setting only
  - c. Bulk Download
    - Used curl to download a file
    - Repeated downloads to compare performance
- Performance Metrics
  - Page Load Time (PLT) for web browsing
  - Interaction Response Time (IRT) for mobile apps
  - Download time for bulk downloads

Setup -continued

- Methodology:
  - Multiple trials per test (5-10 repetitions)
  - Cleared caches between trials
  - Compared DChannel against baseline schemes
    - ALL-URLLC: steers all traffic over URLLC
    - Obj-steering: requests web objects on URLLC whenever its fetching time is smaller than eMBB
    - Best-pkt-size: steers pkts whose size is lower than the best predefined threshold
    - MPTCP
    - ASAP: identifies the different phases of a web transaction (e.g., TLS handshake and HTTP request) and accelerates packets of latency-sensitive phases. It accelerates, for instance, TLS/SSL handshake as well as HTTP request traffic, but leaves HTTP responses to eMBB

Evaluations

Comparing steering schemes

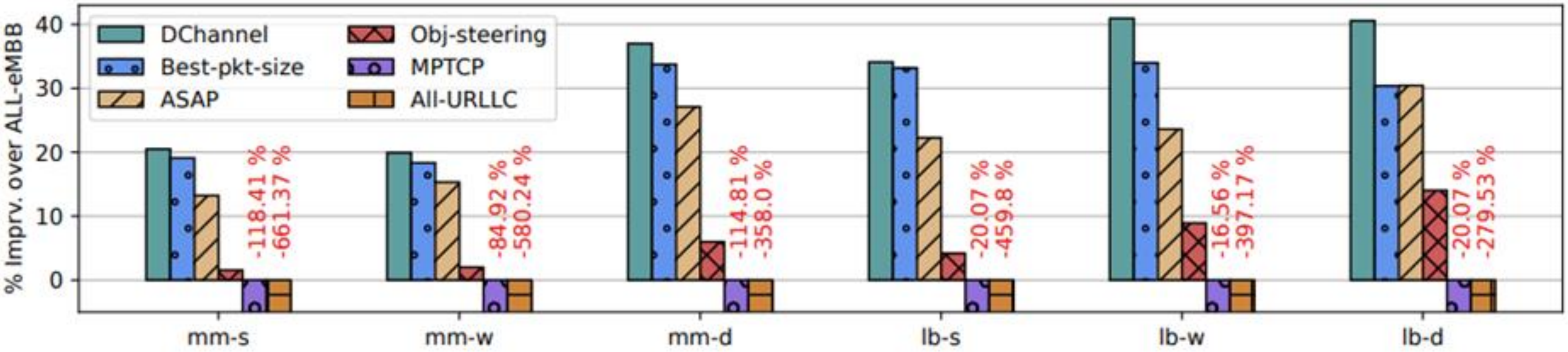
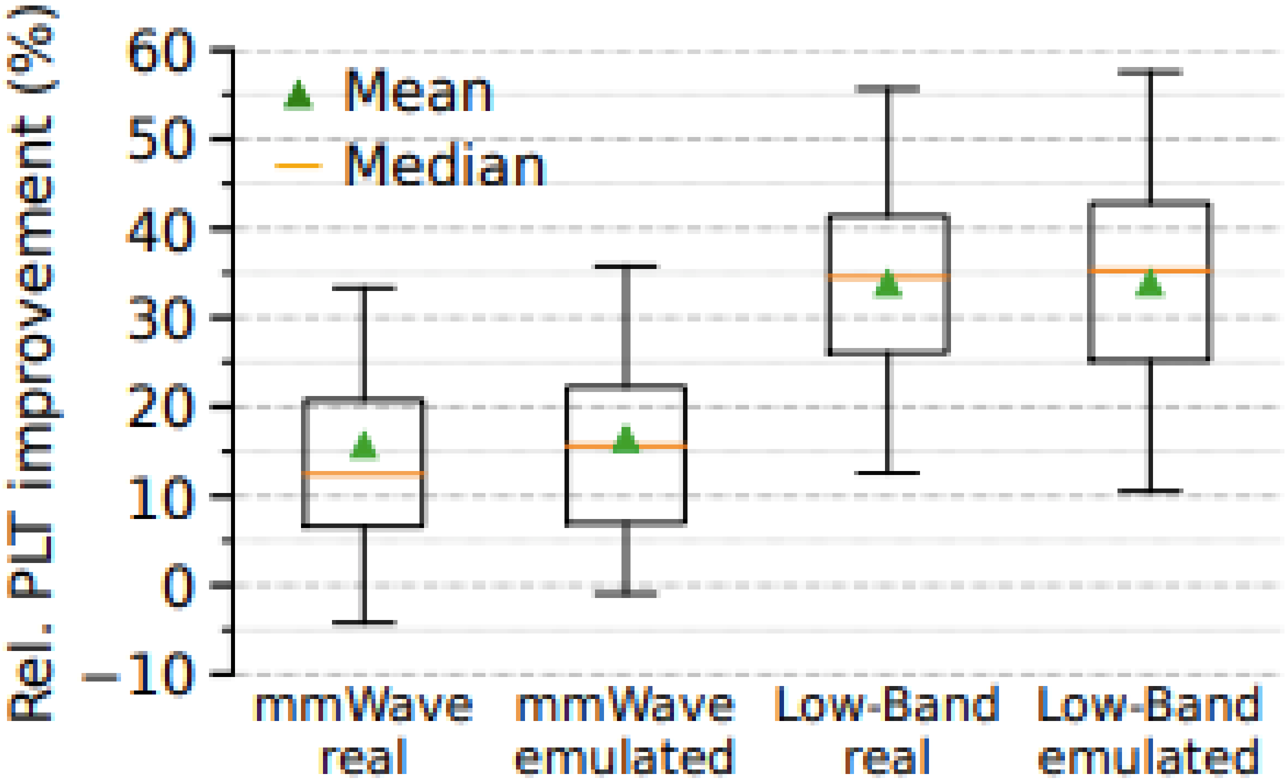


Figure 3: *DChannel offers at least 20% lower PLT compared to that of All-eMBB, and it performs better than all other schemes. MPTCP's PLTs are 17% to 118% worse than when using a single eMBB channel across all traces.*

Live 5G Experiments



Evaluate ROB

uses the default TCP CUBIC, which is sensitive to in-order packet delivery

A stochastic packet drop in the uplink and downlink channels with pkts being dropped in both eMBB and URLLC

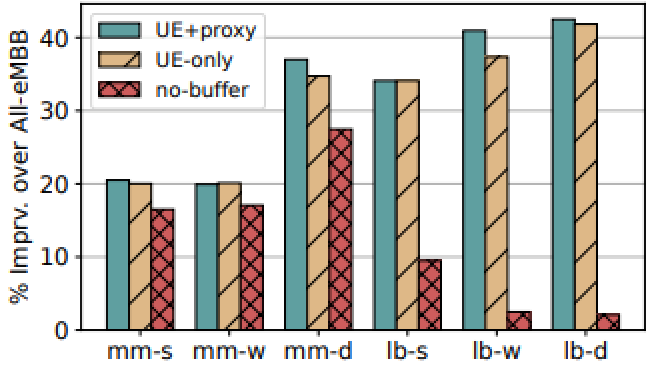


Table 3: *The p50 and (p95) of the avg. and max. buffer sizes (in bytes) when loading 200 web pages under MM-S and LB-D traces.*

	MM-S		*LB-D*	
	Proxy	*UE*	*Proxy*	*UE*
Avg buffer size (b)	2 (15,7)	12 (63.5)	14 (96)	130 (2638)
Max buffer size (b)	392 (1122)	944 (2597)	757 (2375)	2848 (15521)

Table 4: *PLT under different random packet drop rates.*

Loss	*All-eMBB (ms)*		DCHANNEL *(ms)*	
	MM-S	*MM-D*	*MM-S*	*MM-D*
0.0%	1108	1899	883 (20%)	1096 (42%)
0.1%	1203	1963	1011 (16%)	1311 (34%)
1.0%	2643	3421	2502 (5%)	3072 (10%)

Bulk download

Although the primary focus is latency-sensitive applications, how DChannel performed for a bandwidth-intensive use is also important–bulk HTTP transfer of a file.

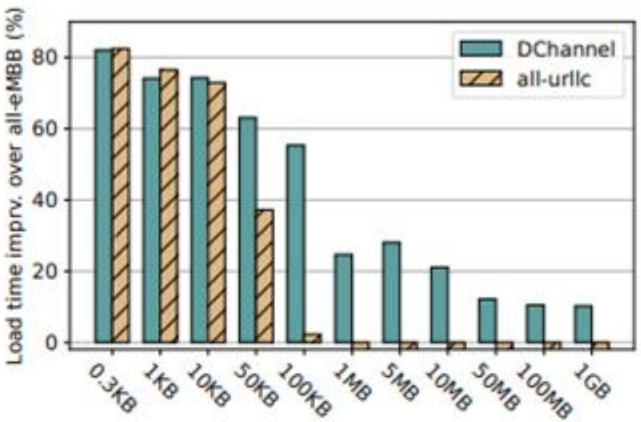


Figure 6: *Download time improvement of variable-sized data under HTTP. The experiment used the MM-D trace with the buffer set to 800 packets (≈ 2× trace BDP).*

Mobile application

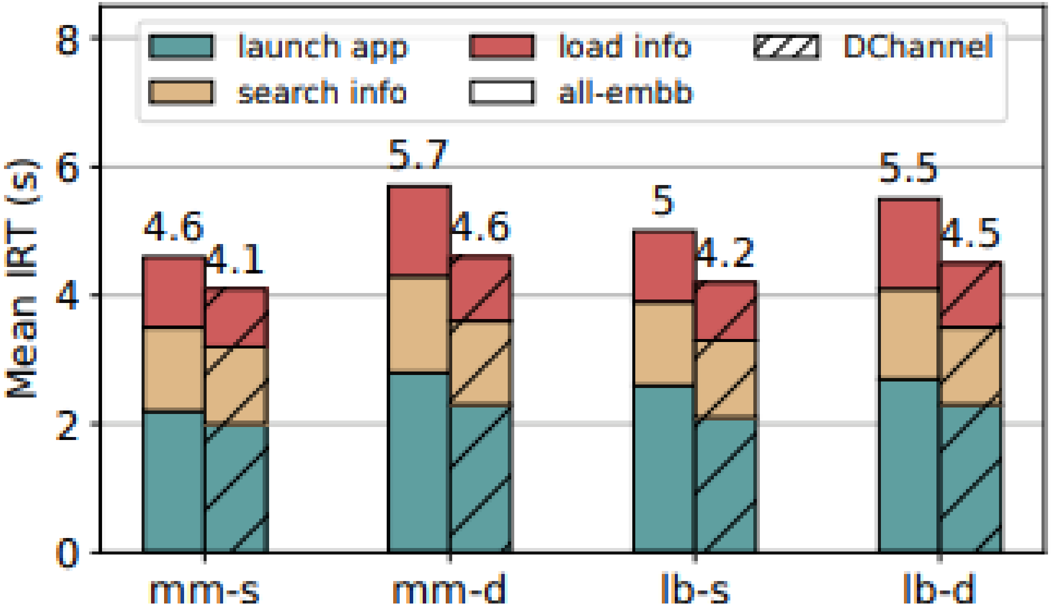
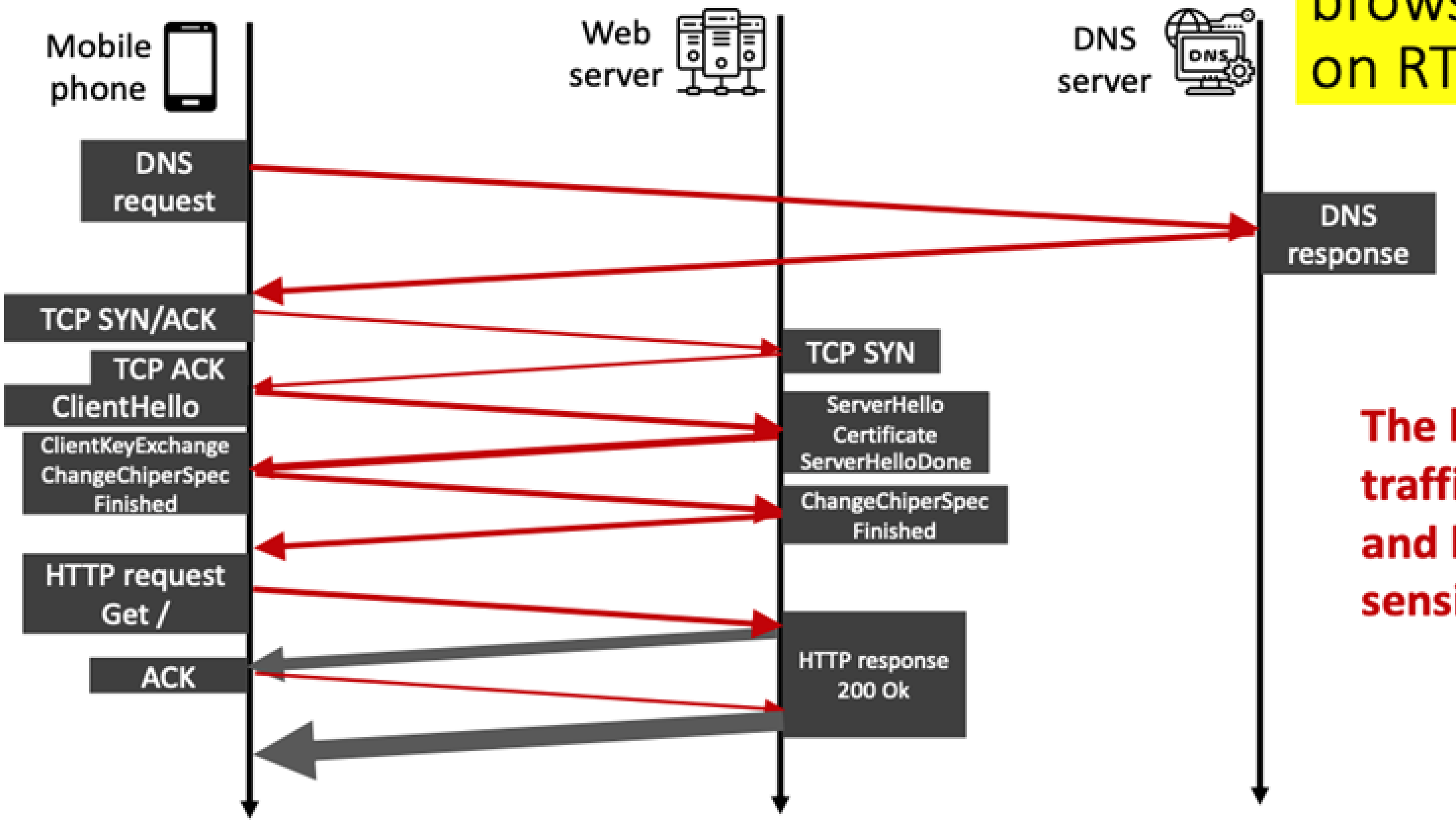


Figure 7: *Android mobile application interaction response time (IRT) of All-eMBB and DChannel when performing three different tasks. We averaged the result from three applications.*

Discussion

Back-up 1

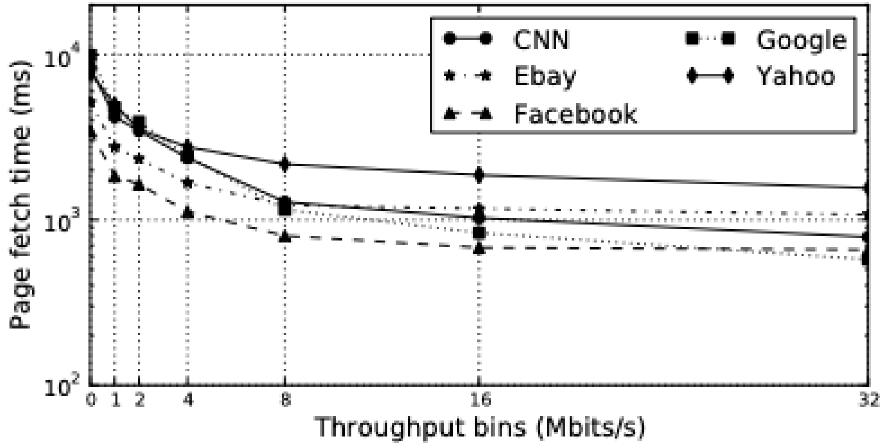


Intuition: web browsing depends on RTT

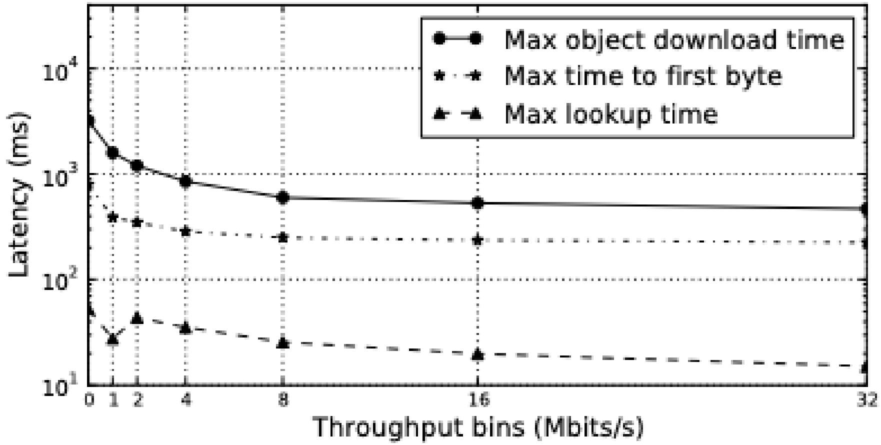
The highlighted traffic are tiny and latency sensitive

Source: [1] nsdi23_slides_sentosa

Back-up 2



(a) *Page load times*



(b) *Components of page load time*

Figure 5: *Page load times decrease with downstream throughput, but only up to 8–16 Mbits/s. X-axis labels denote the start of each throughput bin (e.g., "0" is the set of users with downstream throughput up to 1 Mbits/s.) (SamKnows)*

Source: [1] https://dl.acm.org/doi/pdf/10.1145/2504730.2504741

Back-up 3

performance is not very sensitive to the exact value of α. In particular, even with α = 0 – which corresponds to the greedy strategy, where each packet uses LLC whenever it expects a reward for itself – there is still a very good PLT improvement, within 5% or less of the best α

RTT	$\alpha = 0$		$\alpha = 0.25$		$\alpha = 0.5$		$\alpha = 0.75$		$\alpha = 1$		$\alpha = 2$		$\alpha = 3$	
(ms)	*%ps.*	*%sz.*	*%ps.*	*%sz.*	*%ps.*	*%sz.*	*%ps.*	*%sz.*	*%ps.*	*%sz.*	*%ps.*	*%sz.*	*%ps.*	*%sz.*
20	16.7	13.3	16.6	10.7	18.4	5.9	18.9	5.7	18.5	5.5	15.6	4.7	14.2	4.1
40	31.1	18.8	33.2	16.2	34.0	13.9	34.7	11.9	34.0	11.1	32.8	8.5	31.8	5.8
60	37.5	22.6	40.9	19.5	41.2	17.4	42.1	14.9	40.8	14.2	40.6	11.1	37.4	9.8
80	43.2	26.3	46.3	22.9	46.3	19.9	46.3	17.7	47.1	16.5	45.6	13.3	45.1	11.6
100	47.1	29.4	48.6	26.1	49.7	22.5	50.2	19.9	50.7	18.3	49.8	14.9	48.5	13.1

Back-up4

Table 2: *Comparing the performance of DChannel with All-eMBB and All-URLLC when fetching the (182 KB) landing page of* amazon.com.

Perf. metric	***All-eMBB***	***All-URLLC***	***DChannel***
DNS lookup (ms)	44	8	8
TCP connect (ms)	42	6	6
TLS connect (ms)	53	30	30
Object transfer (ms)	209	809	144
Total load time (ms)	349	853	189