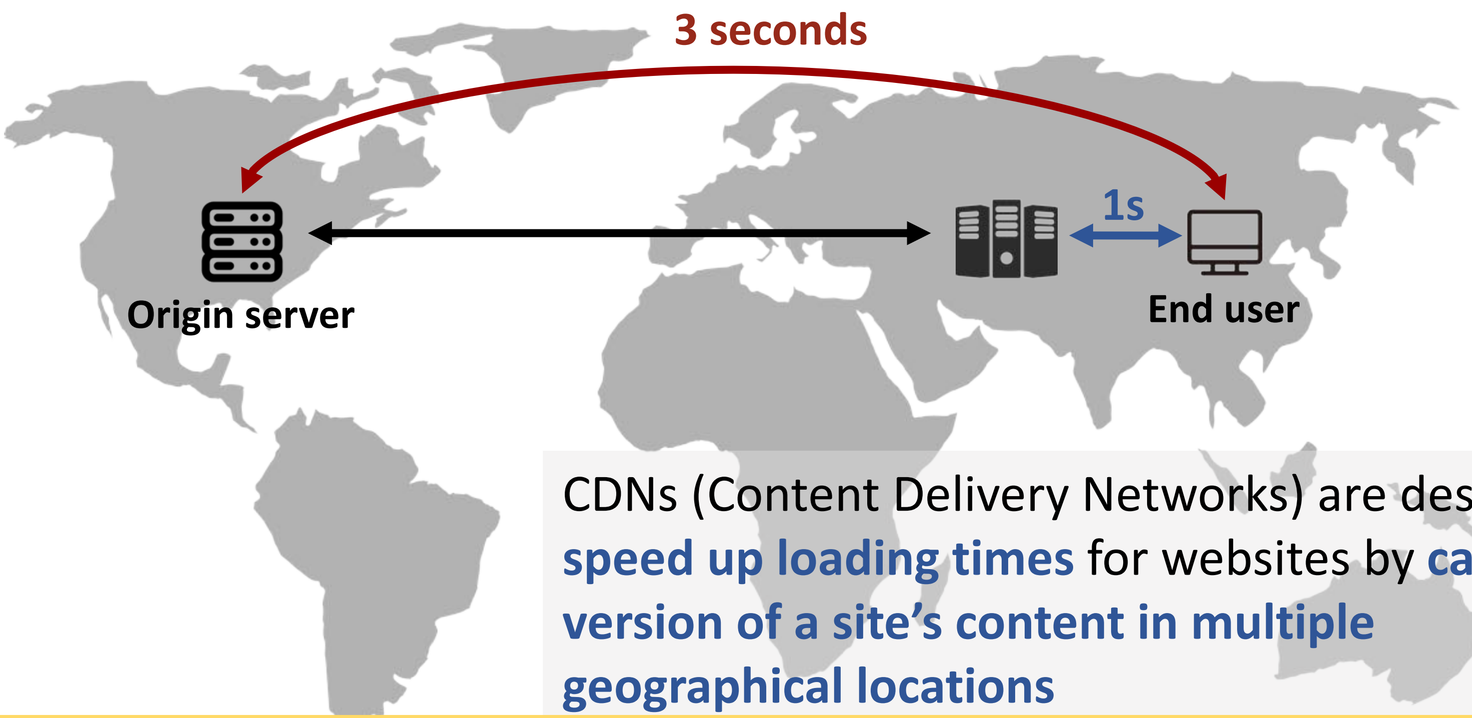
22nd ACM/IFIP International Middleware Conference, Industrial Track (Middleware Industry'21)

POLYGON:

A QUIC-Based CDN Server Selection System Supporting Multiple Resource Demands

Mengying Zhou, Tiancheng Guo, Yang Chen, Junjie Wan, Xin Wang

Everybody Uses CDNs

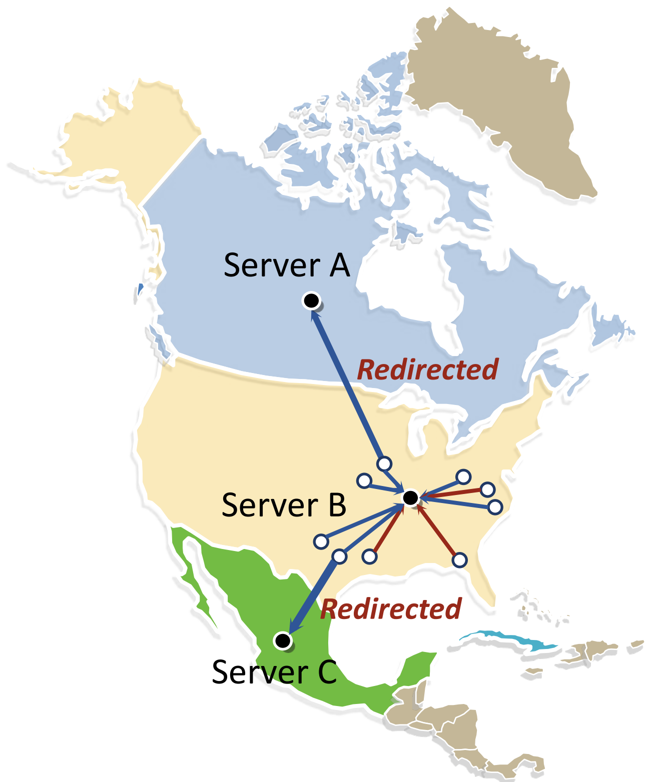


CDNs (Content Delivery Networks) are designed to **speed up loading times** for websites by **caching a version of a site's content in multiple geographical locations**

Is the smallest geographic distance enough?

CLOUDFLARE Akamai fastly

The Usefulness of Bandwidth



● CDN server ○ Client

- A: 100Mbps, 150ms - 1000 requests/s
- B: 1Gbp
- C: 100

- Two content types in the CDN requests:
  - A epic NBA moment video:
    - large size (15M), in the end of the browsing queue
  - A widely used JavaScript library:
    - small size (200K), in the front of the browsing queue

- According to the geographic distance, the CDN requests from **North America** will be **ALL** direct to **Server B**

Avg. Job Completion Time:
- Video: 273.3 s
- JS: 2.2 s

Exceed Load especially bandwidth

- Video requests are **prioritized for bandwidth**, not latency. If we redirect **20% of video requests** to **Server A / Server C**:

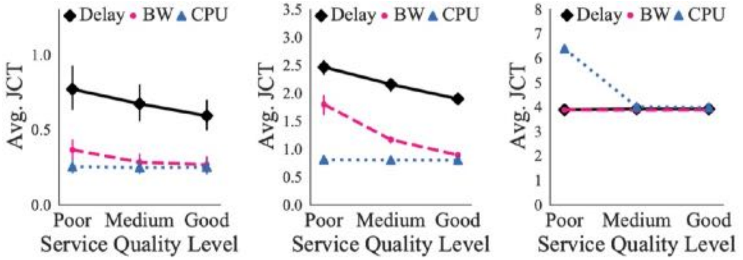
Avg. Job Completion Time:
- Video:
  - **Server A: 256.7 s**
  - **Server B: 265.2 s ↓**
  - **Server C: 242.8 s**

Bandwidth also matters for CDN selection

Delay, Bandwidth, and CPU All Matter

Types of requests sensitive to three common resource types

- Delay-sensitive (e.g. JavaScript file)
- Bandwidth-sensitive (e.g. video)
- CPU-sensitive (e.g. database query)



(a) Delay-sensitive (b) Bandwidth-sensitive (c) CPU-sensitive

Finding: each resource shows the most significant influence on its corresponding type of requests.

Priorities of Request	1st	2nd	3rd
Delay-sensitive	Delay	Bandwidth	CPU
Bandwidth-sensitive	Bandwidth	Delay	CPU
CPU-sensitive	CPU	Delay	Bandwidth

Considering resource priorities of different requests is necessary when selecting CDN servers

How Previous Solutions?

1. Centralized router manager considering the concurrent request load [Alzoubi et al., TWEB'11]
2. FastRoute [Flavel et al., NSDI'15]
   - Offloading traffic to other nodes by editing the DNS resolvers
   - Applied in Bing search engine [Calder et al., IMC'15]

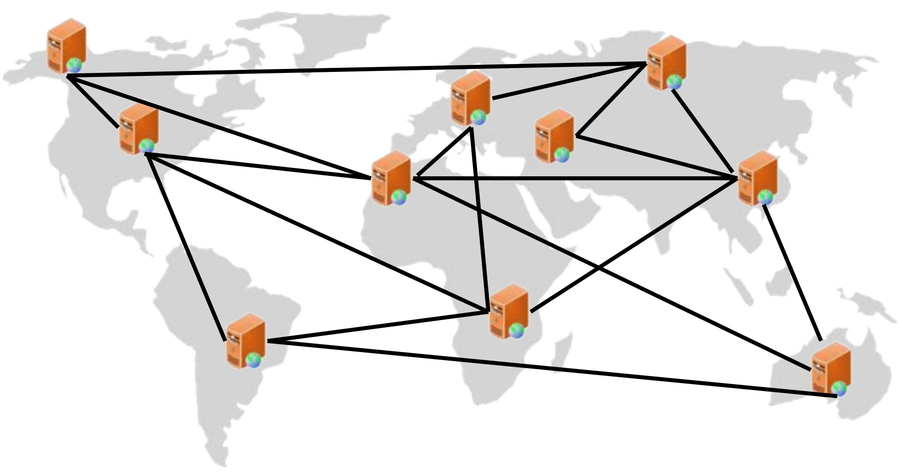
Shortcomings

- ***They treat the resource priority of different types of requests as the same***
- ***They are based on Anycast routing, which loses precise control over CDN server selection***

POLYGON:

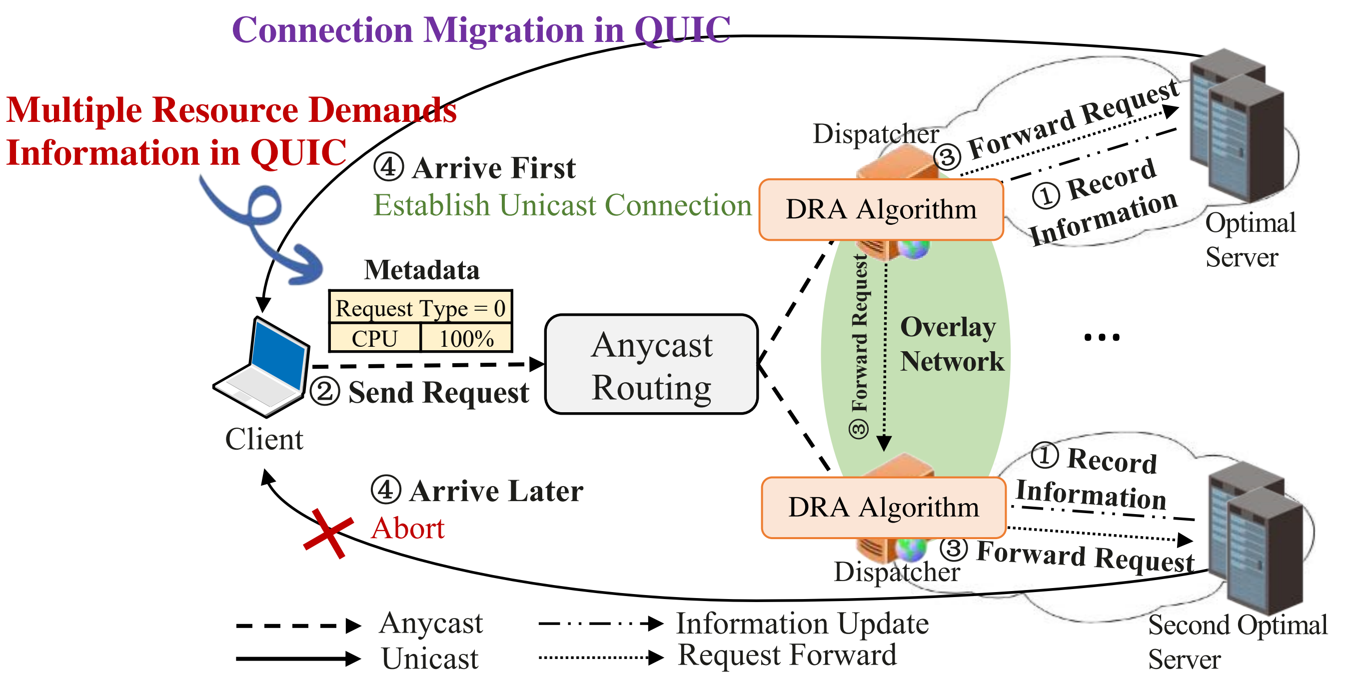
A QUIC-Based CDN Server Selection System Supporting Multiple Resource Demands

A set of dispatchers at strategic network locations:



1. Resource status collection
2. Server assignment with multiple resource demands
3. Forwarding requests to suitable CDN servers at a small cost

Polygon Workflow



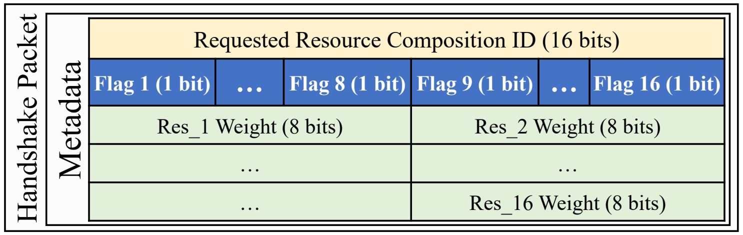
- Step 1: Collecting real-time resource status
- Step 2: Sending request and selecting server { Appending Resource Demand Info; Demand Restriction Allocation (DRA) Alg.
- Step 3: Forwarding request to suitable server { Redundant Forwarding; Fast Forwarding via Overlay Network
- Step 4: Establishing unicast connection and content transmission

QUIC Brings More Benefits

- *QUIC is proposed by Google to reduce the latency and enhance the ability in mobility*

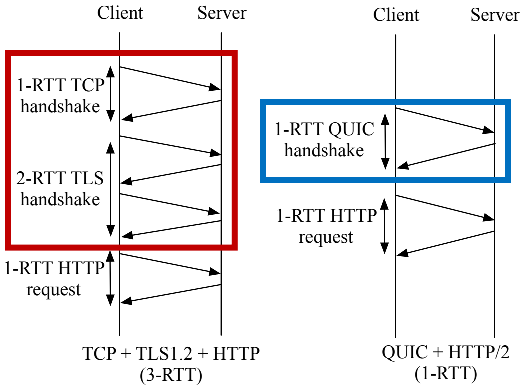
1. Metadata in QUIC

- Metadata block in handshake packet
- Both supporting the pre-defined classic resource compositions and customizable demand weights



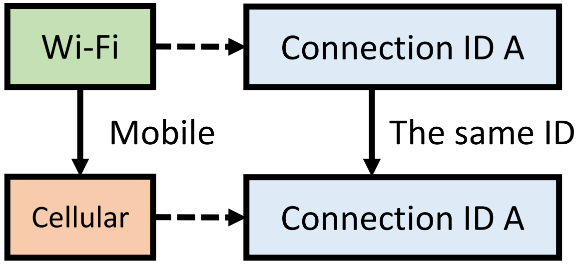
2. Zero-Latency Connection Establishment to Dispatchers

- QUIC reduces the handshake delay with 1-RTT/0-RTT rather than 3-RTT of TCP



3. Eliminating Re-Connection between Client and Server

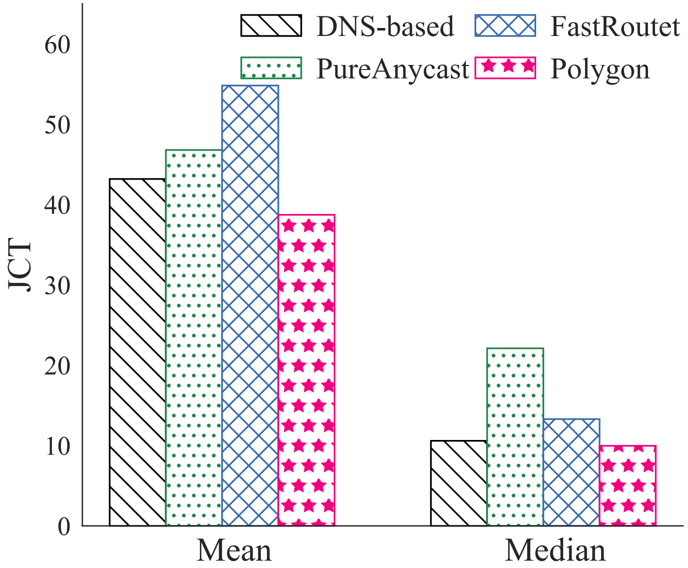
- Connection migration in QUIC guarantees the feasibility of the continuous connection even when the IP address changes
- One QUIC connection can accomplish sending, forwarding, and transmitting



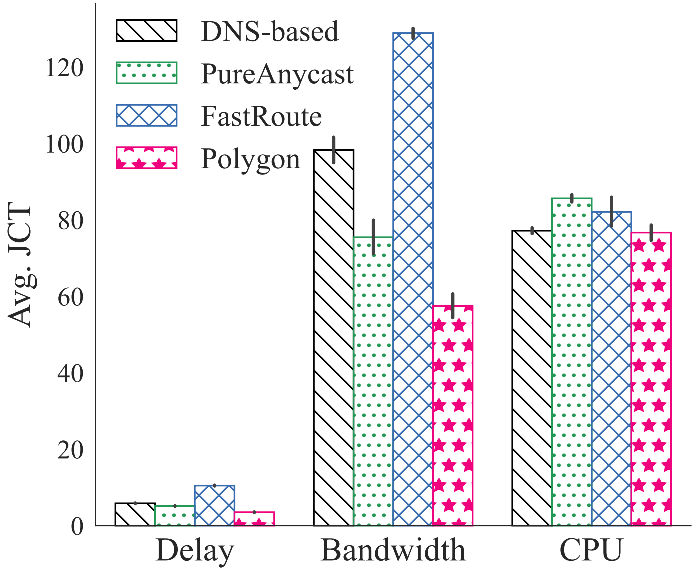
Migration without awareness

Less Job Completion Time

- **Comparison of job completion time performance among DNS-based, PureAnycast-based, FastRoute, and Polygon**



(a) The mean and median values of job completion time



(b) The average job completion time of requests sensitive to different resource types

1. Polygon **achieves less job completion time** in terms of either the mean or median value
2. Overall improvement benefits from each type of requests, especially **bandwidth-sensitive requests**

Higher Resource Utilization Efficiency

- **The number of requests completed in the same 2 hours and the average resource cost to complete per request**

Method	# BW Req.	BW Cost / # BW Req.	# CPU Req.	CPU Cost / # CPU Req.
DNS-based	1570	7.04	421	0.74
PureAnycast	1915	6.31	576	0.60
FastRoute	497	12.56	380	0.90
Polygon	***2166***	***4.71***	***619***	***0.49***
Max improvement	*↑13%*	*↓25%*	*↑7%*	*↓18%*

1. Polygon makes better use of the unoccupied servers and **alleviates the resource preemption** in crowded regions
2. **64%** of CPU-sensitive requests and **34%** of bandwidth-sensitive requests are redirected to **other regions**

POLYGON:

A QUIC-Based CDN Server Selection System Supporting Multiple Resource Demands

1. Requests in different application scenarios would have different resource type priorities
2. Polygon is a QUIC-based CDN server selection system that supports multiple types of resource demands
3. A real-world evaluation demonstrates the significant improvement in job completion time and resource utilization efficiency

Thanks for your listening!

22nd ACM/IFIP International Middleware Conference, Industrial Track (Middleware Industry'21)

POLYGON:

A QUIC-Based CDN Server Selection System Supporting Multiple Resource Demands

Mengying Zhou, Tiancheng Guo, Yang Chen, Junjie Wan, Xin Wang

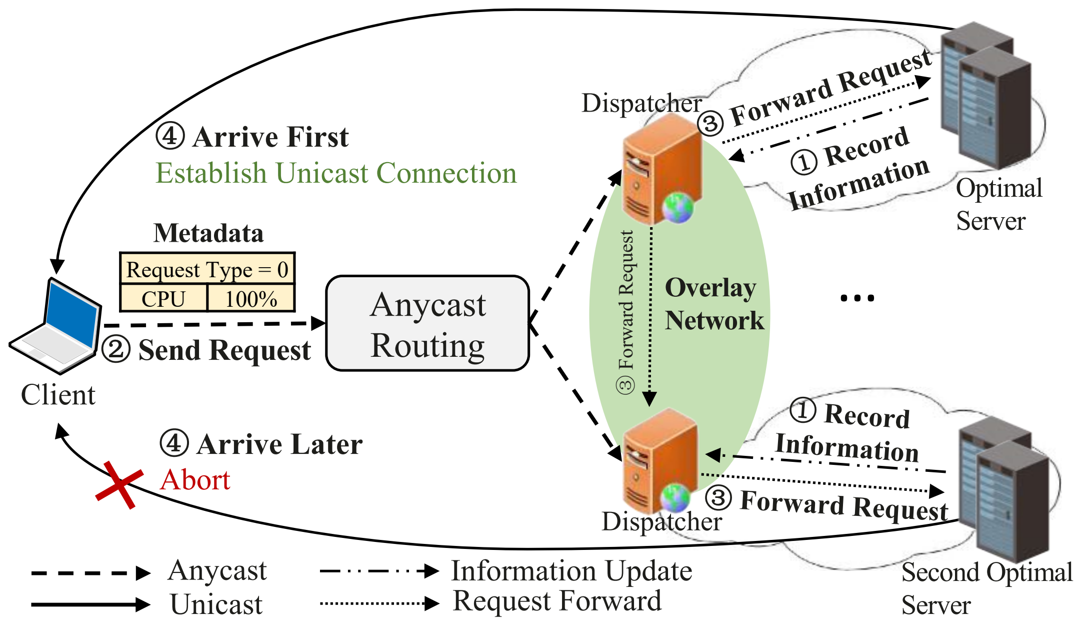
(Technical Details)

1. Requests in different application scenarios would have different resource type priorities

2. Polygon is a QUIC-based CDN server selection system that supports multiple types of resource demands

3. A real-world evaluation demonstrates the significant improvement in job completion time and resource utilization efficiency

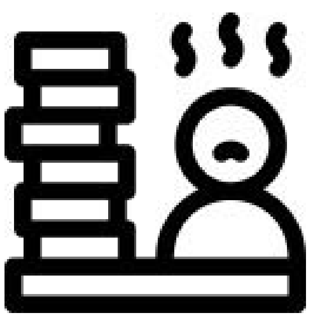
Design Overview

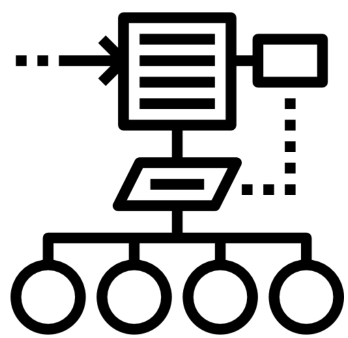


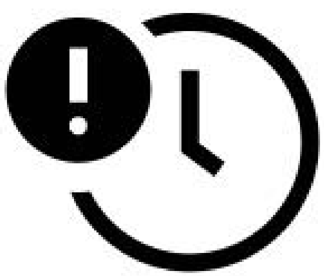
- Step 1: Collecting real-time resource status
- Step 2: Sending request and selecting server
- Step 3: Forwarding request to suitable server
- Step 4: Establishing unicast connection and content transmission

Practical Challenges

- Diversity of resource status
- Effective demand delivery and robust selection
- Extra delay for connecting and forwarding



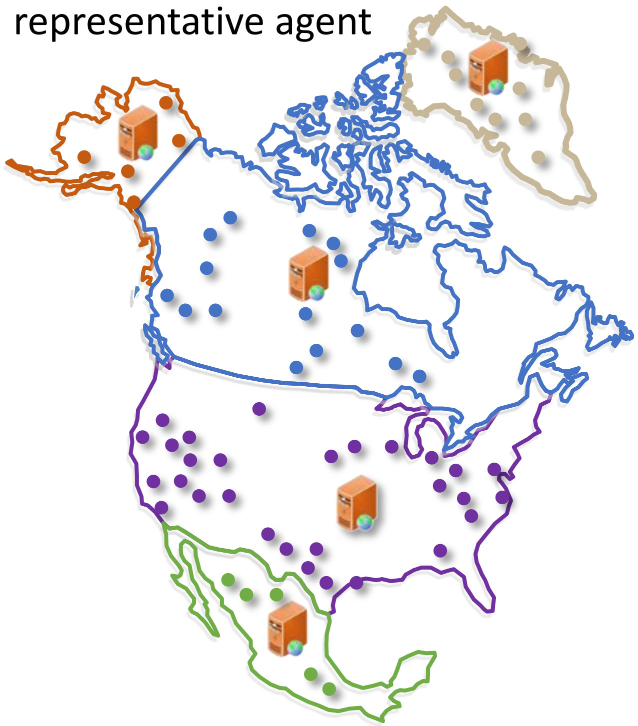




Collection of Resource Status

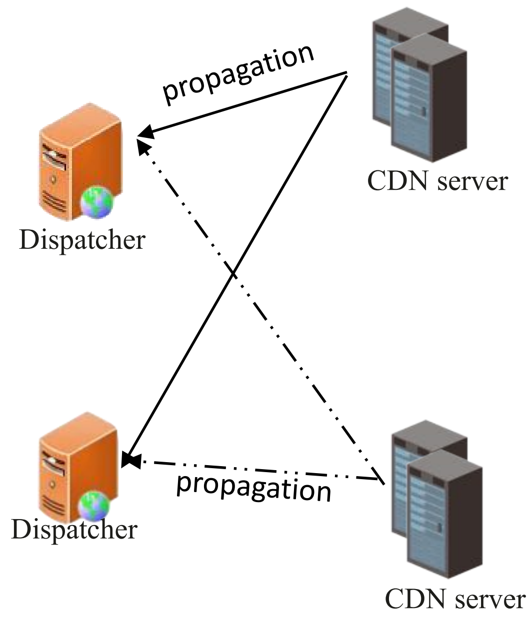
Delay & Bandwidth:

- Provided by the representative agent in each area
- One dispatcher in each region acts as a representative agent



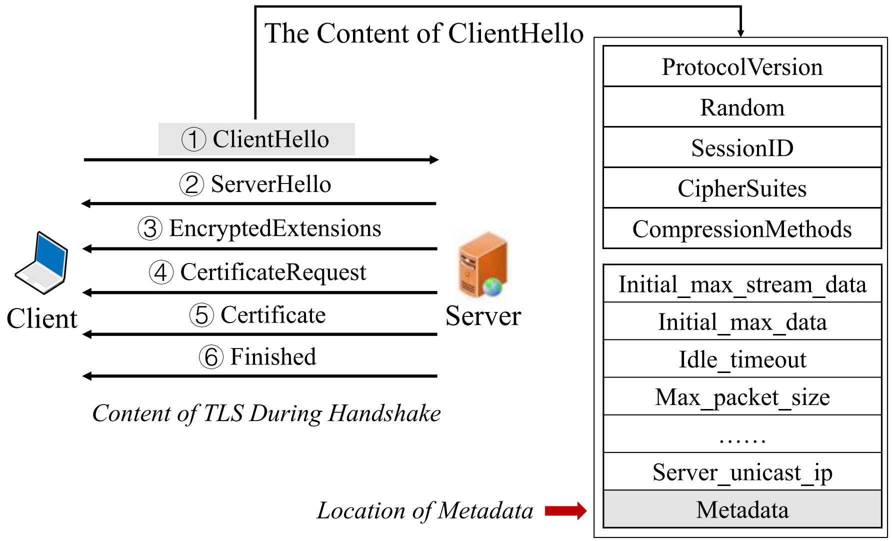
CPU:

- Collected by each server
- CPU capability: *idle rate* × *number of CPU cores* × *CPU clock frequency*

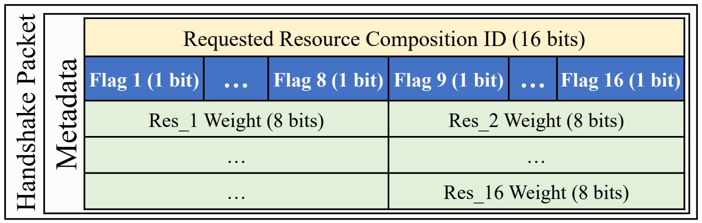


- Collection interval: Delay: 1h, Bandwidth: 1.5s, CPU: 1.5s

Diagram of Metadata Implementation



The design of metadata



- Polygon uses a metadata block within the *ClientHello* packet to specify resource demands

Sending Request with Resource Demands

1. Pre-defined classic resource compositions

e.g. Requested Resource Composition
① ID=1: only sensitive to delay
② ID=2: only sensitive to bandwidth
③ ID=3: only sensitive to CPU
…

Handshake Packet — Metadata

Requested Resource Composition ID (16 bits)					
Flag 1 (1 bit)	**…**	**Flag 8 (1 bit)**	**Flag 9 (1 bit)**	**…**	**Flag 16 (1 bit)**
Res_1 Weight (8 bits)			Res_2 Weight (8 bits)		
…			…		
…			Res_16 Weight (8 bits)		

2. Customizable demand weights

Requested Resource Composition ID = **0**															
1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1
11110 (Decimal 30, parsed for flag 1)								10100 (Decimal 20, parsed for flag 3)							
1010 (Decimal 10, parsed for flag 15)								101000 (Decimal 40, parsed for flag 16)							

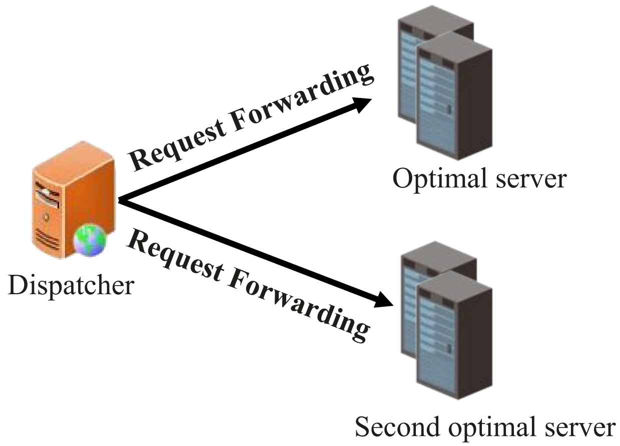
e.g. ID=0, and 30% for delay, 20% for bandwidth, 10% for CPU, 40% for loss

Demand Restriction Allocation (DRA) Algorithm

Two aspects determine the server score:

1. Amount of currently available resources
2. Maximum capability of resources

Calculate servers' scores and pick up the optimal and the second optimal servers

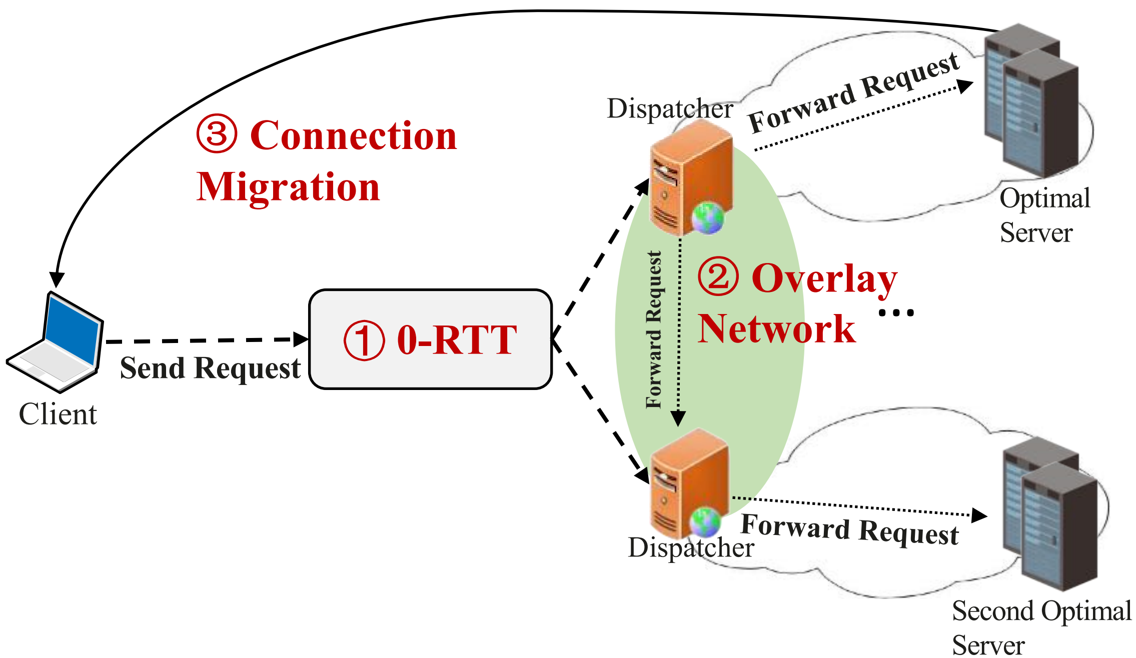


Redundant Forwarding

- Avoiding the potential sharp capacity degradation of the optimal node
- Being triggered when there is no much difference between the two nodes

Request Forwarding by Dispatchers

- **Reducing the extra delay** by the introduction of dispatchers
  ① Zero-latency connection establishment to dispatchers
  ② Fast forwarding via overlay network
  ③ Eliminating re-connection between client and server



Experiment Configuration

- Testbed deployment
  - 5 dispatchers and 5 CDN servers located in five continents
  - 10 clients: Asia (3), North America (3), Europe (2), Australia (1), and South America (1)
- Virtual machine setting
  - Conducted on Google Cloud Platform
  - Ubuntu 18.04 LTS, one standard vCPU and 3.75 GB memory
- Simulated requests
  - Delay-sensitive: frontpages of Alexa Top 500 Sites
  - Bandwidth-sensitive: video file (5MB), visiting 10 times
  - CPU-sensitive: 100 random queries in a database with 1 million entries
  - The ratio of each type of requests: 4:4:1

POLYGON:

A QUIC-Based CDN Server Selection System Supporting Multiple Resource Demands

Mengying Zhou, Tiancheng Guo, Yang Chen, Junjie Wan, Xin Wang

Thanks for your listening!

myzhou19@fudan.edu.cn
https://mengyingzhou.github.io