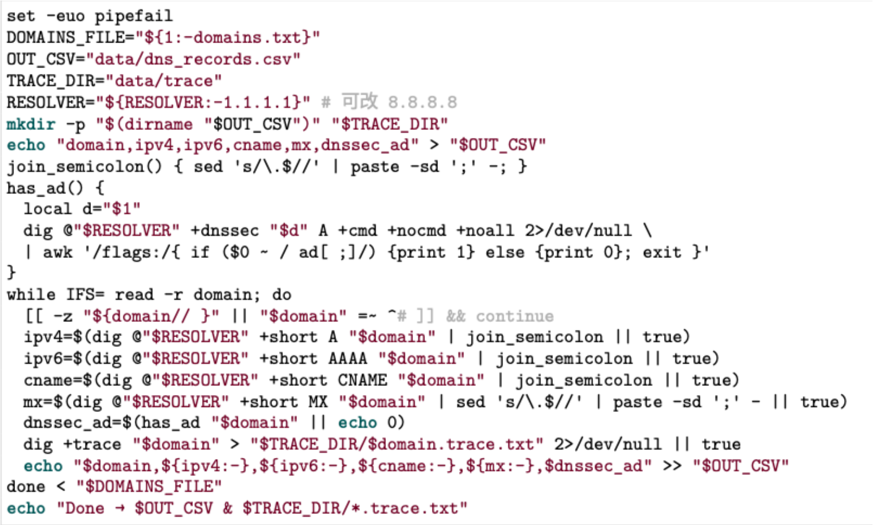
LSAP HW4 Report

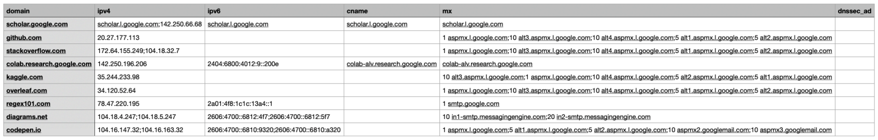
1 Investigating Modern Internet Infrastructure

(1) Domain Analysis

本研究針對十個常用網站，蒐集其 DNS 基本紀錄：A（IPv4）、AAAA（IPv6）、CNAME（正規名稱）、MX（郵件交換）。另以 `dig +dnssec` 檢查回應旗標是否包含 `ad`（Authenticated Data），據此判斷使用之遞迴解析器是否已完成 DNSSEC 簽章驗證。所有查詢統一透過公開遞迴解析器（例如 Cloudflare `1.1.1.1`）執行，以降低本機快取與 ISP/地區策略差異的影響；該解析器支援 DNSSEC 驗證，便於以 `ad` 作為驗簽指標。

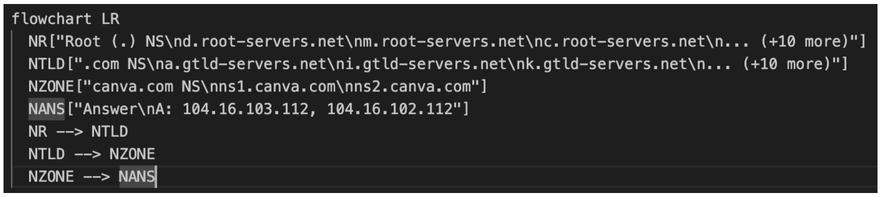


```
set -euo pipefail
DOMAINS_FILE="${1:-domains.txt}"
OUT_CSV="data/dns_records.csv"
TRACE_DIR="data/trace"
RESOLVER="${RESOLVER:-1.1.1.1}" # 可改 8.8.8.8
mkdir -p "$(dirname "$OUT_CSV")" "$TRACE_DIR"
echo "domain,ipv4,ipv6,cname,mx,dnssec_ad" > "$OUT_CSV"
join_semicolon() { sed 's/\.$//' | paste -sd ';' -; }
has_ad() {
  local d="$1"
  dig @"$RESOLVER" +dnssec "$d" A +cmd +nocmd +noall 2>/dev/null \
  | awk '/flags:/{ if ($0 ~ / ad[ ;]/) {print 1} else {print 0}; exit }'
}
while IFS= read -r domain; do
  [[ -z "${domain// }" || "$domain" =~ ^# ]] && continue
  ipv4=$(dig @"$RESOLVER" +short A "$domain" | join_semicolon || true)
  ipv6=$(dig @"$RESOLVER" +short AAAA "$domain" | join_semicolon || true)
  cname=$(dig @"$RESOLVER" +short CNAME "$domain" | join_semicolon || true)
  mx=$(dig @"$RESOLVER" +short MX "$domain" | sed 's/\.$//' | paste -sd ';' - || true)
  dnssec_ad=$(has_ad "$domain" || echo 0)
  dig +trace "$domain" > "$TRACE_DIR/$domain.trace.txt" 2>/dev/null || true
  echo "$domain,${ipv4:-},${ipv6:-},${cname:-},${mx:-},$dnssec_ad" >> "$OUT_CSV"
done < "$DOMAINS_FILE"
echo "Done → $OUT_CSV & $TRACE_DIR/*.trace.txt"
```



domain	ipv4	ipv6	cname	mx	dnssec_ad
scholar.google.com	scholar.l.google.com;142.250.66.68	scholar.l.google.com	scholar.l.google.com	scholar.l.google.com	
github.com	20.27.177.113			1 aspmx.l.google.com;10 alt3.aspmx.l.google.com;10 alt4.aspmx.l.google.com;5 alt1.aspmx.l.google.com;5 alt2.aspmx.l.google.com	
stackoverflow.com	172.64.155.249;104.18.32.7			1 aspmx.l.google.com;10 alt3.aspmx.l.google.com;10 alt4.aspmx.l.google.com;5 alt1.aspmx.l.google.com;5 alt2.aspmx.l.google.com	
colab.research.google.com	142.250.196.206	2404:6800:4012:9::200e	colab-alv.research.google.com	colab-alv.research.google.com	
kaggle.com	35.244.233.98			10 alt3.aspmx.l.google.com;1 aspmx.l.google.com;10 alt4.aspmx.l.google.com;5 alt2.aspmx.l.google.com;5 alt1.aspmx.l.google.com	
overleaf.com	34.120.52.64			1 aspmx.l.google.com;10 alt3.aspmx.l.google.com;10 alt4.aspmx.l.google.com;5 alt1.aspmx.l.google.com;5 alt2.aspmx.l.google.com	
regex101.com	78.47.220.195	2a01:4f8:1c1c:13a4::1		1 smtp.google.com	
diagrams.net	104.18.4.247;104.18.5.247	2606:4700::6812:4f7;2606:4700::6812:5f7		10 in1-smtp.messagingengine.com;20 in2-smtp.messagingengine.com	
codepen.io	104.16.147.32;104.16.163.32	2606:4700::6810:9320;2606:4700::6810:a320		1 aspmx.l.google.com;5 alt1.aspmx.l.google.com;5 alt2.aspmx.l.google.com;10 aspmx2.googlemail.com;10 aspmx3.googlemail.com	

Figure 1: 10 個網站 DNS 紀錄彙整



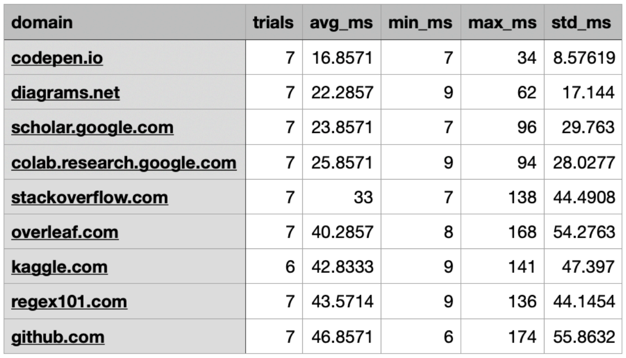
```
flowchart LR
  NR["Root (.) NS\nd.root-servers.net\nm.root-servers.net\nc.root-servers.net\n... (+10 more)"]
  NTLD[".com NS\na.gtld-servers.net\ni.gtld-servers.net\nk.gtld-servers.net\n... (+10 more)"]
  NZONE["canva.com NS\nns1.canva.com\nns2.canva.com"]
  NANS["Answer\nA: 104.16.103.112, 104.16.102.112"]
  NR --> NTLD
  NTLD --> NZONE
  NZONE --> NANS
```

Figure 2: visual diagram of canva DNS lookup path

(2) DNS Resolution Time Measurement

我們用 dig +stats 向指定遞迴解析器（預設 1.1.1.1）對 domains.txt 的 10 個網域各做多次查詢（預設 7 次），從輸出的「Query time」擷取每次解析耗時（毫秒），記錄 A（或設定 RECORD_TYPE=AAAA）。

```
set -euo pipefail
DOMAINS_FILE="${1:-domains.txt}"
OUT_RAW="data/dns_time_raw.csv"
OUT_SUM="data/dns_time_avg.csv"
RESOLVER="${RESOLVER:-1.1.1.1}"
RECORD_TYPE="${RECORD_TYPE:-A}"
TRIALS="${TRIALS:-7}"
mkdir -p data
echo "domain,trial,ms" > "$OUT_RAW"
while IFS= read -r domain; do
  [[ -z "${domain// }" || "$domain" =~ ^# ]] && continue
  for t in $(seq 1 "$TRIALS"); do
    out="$(dig @"$RESOLVER" +tries=1 +time=2 "$domain" "$RECORD_TYPE" +noall +answer +
        stats 2>/dev/null || true)"
    ms="$(printf "%s\n" "$out" | awk -F': ' '/^;; Query time:/{print $2}' | awk '{print $1
        }')"
    [[ -z "$ms" ]] && ms="-1"
    echo "$domain,$t,$ms" >> "$OUT_RAW"
    sleep "0.$(( RANDOM % 4 ))"
  done
done < "$DOMAINS_FILE"
awk -F, 'BEGIN{
  OFS=","; print "domain,trials,avg_ms,min_ms,max_ms,std_ms"
}
NR>1 {
  if ($3 >= 0) {
    n[$1]++; s[$1]+=$3; ss[$1]+=$3*$3;
    if (!($1 in min) || $3<min[$1]) min[$1]=$3;
    if (!($1 in max) || $3>max[$1]) max[$1]=$3;
  }
}
END{
  for (d in n) {
    avg = s[d]/n[d];
    var = (ss[d]/n[d]) - (avg*avg); if (var < 0) var = 0;
    std = sqrt(var);
    print d, n[d], avg, min[d], max[d], std;
  }
}' "$OUT_RAW" | sort -t, -k3,3n > "$OUT_SUM"
echo "Done:"
echo " - $OUT_RAW"
echo " - $OUT_SUM"
```



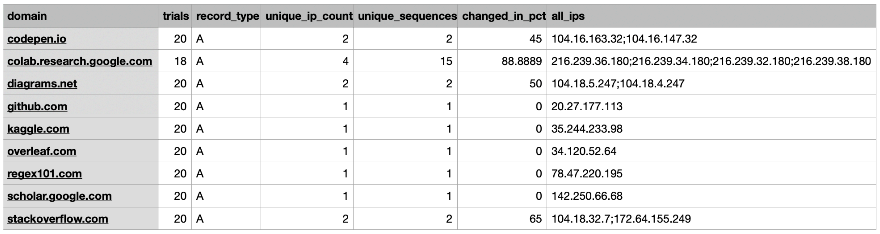
domain	trials	avg_ms	min_ms	max_ms	std_ms
codepen.io	7	16.8571	7	34	8.57619
diagrams.net	7	22.2857	9	62	17.144
scholar.google.com	7	23.8571	7	96	29.763
colab.research.google.com	7	25.8571	9	94	28.0277
stackoverflow.com	7	33	7	138	44.4908
overleaf.com	7	40.2857	8	168	54.2763
kaggle.com	6	42.8333	9	141	47.397
regex101.com	7	43.5714	9	136	44.1454
github.com	7	46.8571	6	174	55.8632

Figure 3: DNS resolution times 之平均

(3) DNS Load Balancing

對各網域連續查詢多次（A/AAAA），觀察是否出現不同的 IP 集合（unique_answers>1），或僅順序不同（order_varied=yes，常見 Round-Robin），以偵測 DNS 層級的負載平衡。

```
#!/usr/bin/env bash
set -euo pipefail
DOMAINS_FILE="${DOMAINS_FILE:-domains.txt}"
OUT_RAW="data/dns_lb_raw.csv"
OUT_SUM="data/dns_lb_summary.csv"
RESOLVER="${RESOLVER:-1.1.1.1}"
RECORD_TYPE="${RECORD_TYPE:-A}"
TRIALS="${TRIALS:-20}"
SUBNET_OPT="${SUBNET:++subnet=${SUBNET}}"
mkdir -p data
echo "domain,trial,ips_ordered" > "$OUT_RAW"
run_for_domain() {
  d="$1"
  [ -z "${d// }" ] && return
  case "$d" in \#*) return;; esac
  for t in $(seq 1 "$TRIALS"); do
    ans="$(dig @"$RESOLVER" $SUBNET_OPT +tries=1 +time=2 "$d" "$RECORD_TYPE" +noall +
        answer 2>/dev/null || true)"
    ips="$(printf "%s\n" "$ans" | awk -v rt="$RECORD_TYPE" '$4==rt {print $5}' | sed 's/\.
        $//' | paste -sd ';' -)"
    [ -z "$ips" ] && ips="(no-answer)"
    echo "$d,$t,$ips" >> "$OUT_RAW"
    sleep "0.$((RANDOM%3))"
  done
}
if [ -n "${DOMAINS:-}" ]; then
  set -- $DOMAINS
  for d in "$@"; do run_for_domain "$d"; done
else
  while IFS= read -r d; do run_for_domain "$d"; done < "$DOMAINS_FILE"
fi
awk -F, -v RT="$RECORD_TYPE" '
BEGIN{OFS=","; print "domain,trials,record_type,unique_ip_count,unique_sequences,
    changed_in_pct,all_ips"}
NR>1{
  dom=$1; seq=$3; cnt[dom]++
  key=dom SUBSEP seq
  if(!(key in seqseen)) seqseen[key]=1
  n=split(seq,a,/;/)
  for(i=1;i<=n;i++){ ip=a[i]; if(ip!="" && ip!="(no-answer)") ipseen[dom SUBSEP ip]=1 }
  if(!(dom in first)) first[dom]=seq
  if(seq!=first[dom]) chg[dom]++
}
END{
  for(k in ipseen){ split(k,p,SUBSEP); d=p[1]; ip=p[2]; if(!(d in uipcount)) uipcount[d
      ]=0; if(!( (d SUBSEP ip) in touched)){ touched[d SUBSEP ip]=1; uiplist[d]=(uiplist[
      d] ";" ip); uipcount[d]++ } }
  for(k in seqseen){ split(k,p,SUBSEP); d=p[1]; useq[d]++ }
  for(d in cnt){
    ips = (d in uiplist)? substr(uiplist[d],2) : ""
    u = (d in uipcount)? uipcount[d] : 0
    us = (d in useq)? useq[d] : 0
    p = (chg[d]+0) * 100.0 / cnt[d]
    print d, cnt[d], RT, u, us, p, ips
  }
}
```



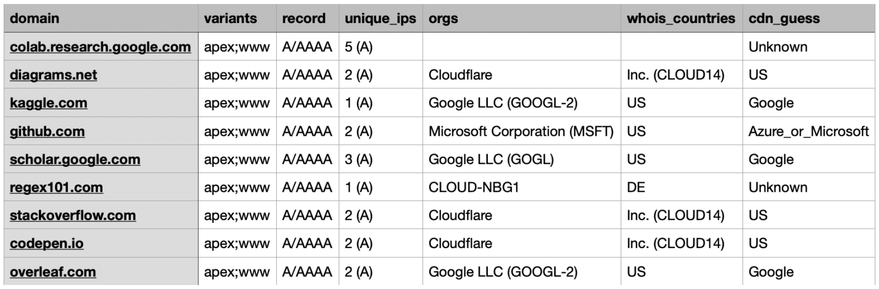
domain	trials	record_type	unique_ip_count	unique_sequences	changed_in_pct	all_ips
codepen.io	20	A	2	2	45	104.16.163.32;104.16.147.32
colab.research.google.com	18	A	4	15	88.8889	216.239.36.180;216.239.34.180;216.239.32.180;216.239.38.180
diagrams.net	20	A	2	2	50	104.18.5.247;104.18.4.247
github.com	20	A	1	1	0	20.27.177.113
kaggle.com	20	A	1	1	0	35.244.233.98
overleaf.com	20	A	1	1	0	34.120.52.64
regex101.com	20	A	1	1	0	78.47.220.195
scholar.google.com	20	A	1	1	0	142.250.66.68
stackoverflow.com	20	A	2	2	65	104.18.32.7;172.64.155.249

Figure 4: different IP responses

(4) CDN Identification

判斷各網站是否位於 CDN 後方，並辨識供應商（Cloudflare、Akamai、Fastly、CloudFront 等）；同時蒐集可佐證之「邊緣節點(Edge POP)」代碼(例如 CF-Ray 的 -TPE、CloudFront 的 X-Amz-Cf-Pop)。

```
chmod +x scripts/cdn_detect.sh
bash scripts/cdn_detect.sh
```

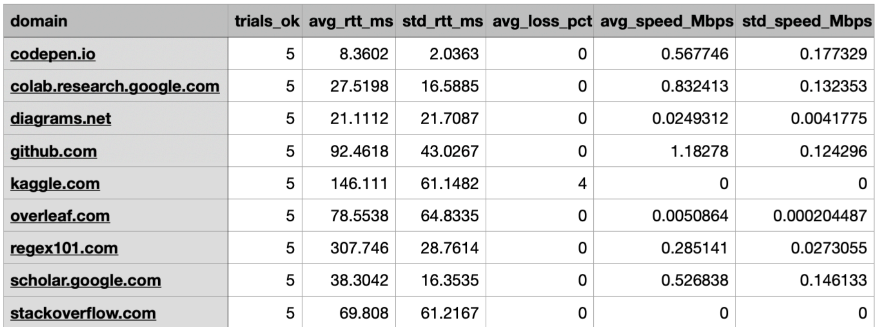


domain	variants	record	unique_ips	orgs	whois_countries	cdn_guess
colab.research.google.com	apex;www	A/AAAA	5 (A)			Unknown
diagrams.net	apex;www	A/AAAA	2 (A)	Cloudflare	Inc. (CLOUD14)	US
kaggle.com	apex;www	A/AAAA	1 (A)	Google LLC (GOOGL-2)	US	Google
github.com	apex;www	A/AAAA	2 (A)	Microsoft Corporation (MSFT)	US	Azure_or_Microsoft
scholar.google.com	apex;www	A/AAAA	3 (A)	Google LLC (GOGL)	US	Google
regex101.com	apex;www	A/AAAA	1 (A)	CLOUD-NBG1	DE	Unknown
stackoverflow.com	apex;www	A/AAAA	2 (A)	Cloudflare	Inc. (CLOUD14)	US
codepen.io	apex;www	A/AAAA	2 (A)	Cloudflare	Inc. (CLOUD14)	US
overleaf.com	apex;www	A/AAAA	2 (A)	Google LLC (GOOGL-2)	US	Google

Figure 5: CDN provider and edge server locations

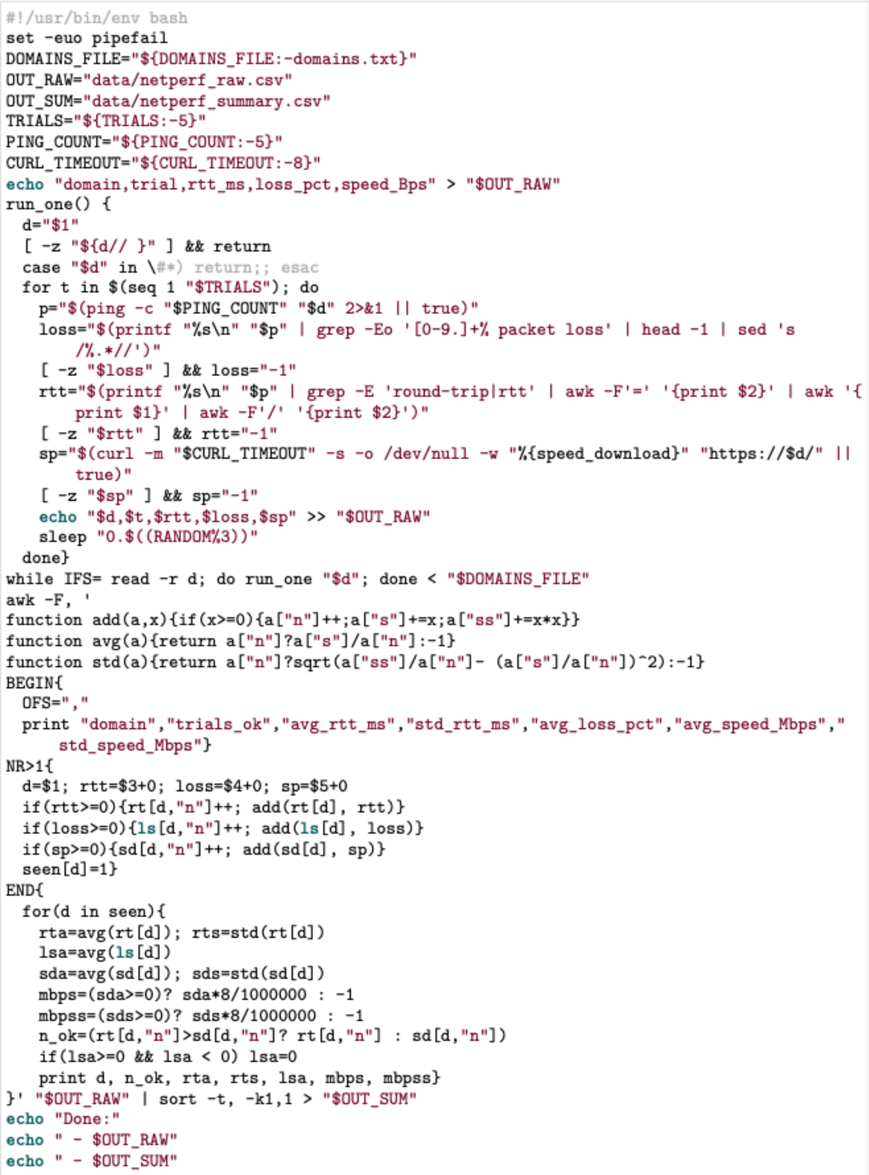
(5) Network Performance Monitoring

針對十個網站量測三項網路效能指標：**平均往返延遲（RTT, ms）、封包遺失率（%）、下載吞吐（Mbps）**，並彙整成表格。



domain	trials_ok	avg_rtt_ms	std_rtt_ms	avg_loss_pct	avg_speed_Mbps	std_speed_Mbps
codepen.io	5	8.3602	2.0363	0	0.567746	0.177329
colab.research.google.com	5	27.5198	16.5885	0	0.832413	0.132353
diagrams.net	5	21.1112	21.7087	0	0.0249312	0.0041775
github.com	5	92.4618	43.0267	0	1.18278	0.124296
kaggle.com	5	146.111	61.1482	4	0	0
overleaf.com	5	78.5538	64.8335	0	0.0050864	0.000204487
regex101.com	5	307.746	28.7614	0	0.285141	0.0273055
scholar.google.com	5	38.3042	16.3535	0	0.526838	0.146133
stackoverflow.com	5	69.808	61.2167	0	0	0

Figure 6: Network Performance

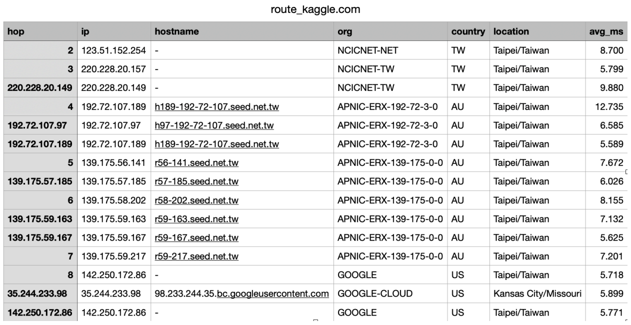


```
#!/usr/bin/env bash
set -euo pipefail
DOMAINS_FILE="${DOMAINS_FILE:-domains.txt}"
OUT_RAW="data/netperf_raw.csv"
OUT_SUM="data/netperf_summary.csv"
TRIALS="${TRIALS:-5}"
PING_COUNT="${PING_COUNT:-5}"
CURL_TIMEOUT="${CURL_TIMEOUT:-8}"
echo "domain,trial,rtt_ms,loss_pct,speed_Bps" > "$OUT_RAW"
run_one() {
  d="$1"
  [ -z "${d// }" ] && return
  case "$d" in \#*) return;; esac
  for t in $(seq 1 "$TRIALS"); do
    p="$(ping -c "$PING_COUNT" "$d" 2>&1 || true)"
    loss="$(printf "%s\n" "$p" | grep -Eo '[0-9.]+% packet loss' | head -1 | sed 's
        /%.*//')"
    [ -z "$loss" ] && loss="-1"
    rtt="$(printf "%s\n" "$p" | grep -E 'round-trip|rtt' | awk -F'=' '{print $2}' | awk '{
        print $1}' | awk -F'/' '{print $2}')"
    [ -z "$rtt" ] && rtt="-1"
    sp="$(curl -m "$CURL_TIMEOUT" -s -o /dev/null -w "%{speed_download}" "https://$d/" ||
        true)"
    [ -z "$sp" ] && sp="-1"
    echo "$d,$t,$rtt,$loss,$sp" >> "$OUT_RAW"
    sleep "0.$((RANDOM%3))"
  done}
while IFS= read -r d; do run_one "$d"; done < "$DOMAINS_FILE"
awk -F, '
function add(a,x){if(x>=0){a["n"]++;a["s"]+=x;a["ss"]+=x*x}}
function avg(a){return a["n"]?a["s"]/a["n"]:-1}
function std(a){return a["n"]?sqrt(a["ss"]/a["n"]- (a["s"]/a["n"])^2):-1}
BEGIN{
  OFS=","
  print "domain","trials_ok","avg_rtt_ms","std_rtt_ms","avg_loss_pct","avg_speed_Mbps","
      std_speed_Mbps"}
NR>1{
  d=$1; rtt=$3+0; loss=$4+0; sp=$5+0
  if(rtt>=0){rt[d,"n"]++; add(rt[d], rtt)}
  if(loss>=0){ls[d,"n"]++; add(ls[d], loss)}
  if(sp>=0){sd[d,"n"]++; add(sd[d], sp)}
  seen[d]=1}
END{
  for(d in seen){
    rta=avg(rt[d]); rts=std(rt[d])
    lsa=avg(ls[d])
    sda=avg(sd[d]); sds=std(sd[d])
    mbps=(sda>=0)? sda*8/1000000 : -1
    mbpss=(sds>=0)? sds*8/1000000 : -1
    n_ok=(rt[d,"n"]>sd[d,"n"]? rt[d,"n"] : sd[d,"n"])
    if(lsa>=0 && lsa < 0) lsa=0
    print d, n_ok, rta, rts, lsa, mbps, mbpss}
}' "$OUT_RAW" | sort -t, -k1,1 > "$OUT_SUM"
echo "Done:"
echo " - $OUT_RAW"
echo " - $OUT_SUM"
```

(6) Network Routing Path Analysis

traceroute 量測從本機到 cloudflare.com 的實際路由，擷取每一跳的回應時間；再用自動化腳本將輸出解析為表格，計算各 hop 的平均往返延遲（ms）。同時對每個 IP 做反向 DNS 與 IP 資訊查詢，補齊 **Hostname / Organization（ISP） / Country / City / 經緯度 **。

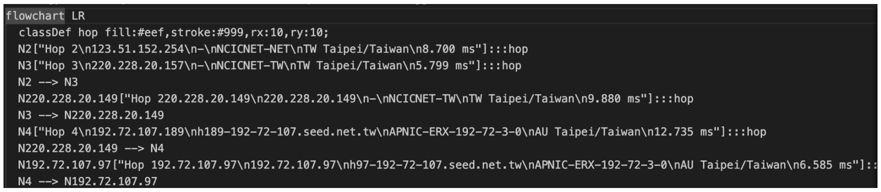
```
#!/usr/bin/env bash
set -euo pipefail
D="${1:?usage: route_analyze.sh <domain>}"
OUT_CSV="data/route_${D}.csv"
OUT_MMD="report/route_${D}.mmd"
TR="$(command -v traceroute || true)"
[ -z "$TR" ] && { echo "traceroute not found"; exit 1; }
echo "hop,ip,hostname,org,country,location,avg_ms" > "$OUT_CSV"
$TR -n -q 3 -w 2 "$D" | awk 'NR>1' | while read -r line; do
  hop="$(echo "$line" | awk "{print \$1}")"
  ip="$(echo "$line" | grep -Eo '([0-9]{1,3}\.){3}[0-9]{1,3}' | head -1)"
  if [ -z "$ip" ]; then
    echo "$line" | grep -q '\* \* \*' && ip="*"
  fi
  if [ "$ip" = "*" ] || [ -z "$ip" ]; then
    hn="*"; org=""; ctry=""; loc=""; avg="-1"
  else
    hn="$(dig +short -x "$ip" | sed 's/\.$//' | head -1)"
    [ -z "$hn" ] && hn="-"
    ms_raw="$(echo "$line" | grep -Eo '[0-9]+\.[0-9]+ ms' | awk '{print $1}')"
    if [ -z "$ms_raw" ]; then avg="-1"; else
      c=0; s=0; echo "$ms_raw" | while read -r v; do c=$((c+1)); s=$(awk -v a="$s" -v b="
          $v" 'BEGIN{printf "%.6f", a+b}'); done
      avg="$(awk -v s="$s" -v c="$c" 'BEGIN{ if(c>0) printf "%.3f", s/c; else print "-1"
          }')"
    fi
    wf="$(whois "$ip" 2>/dev/null || true)"
    org="$(printf "%s\n" "$wf" | awk -F': *' 'tolower($1)~/^(orgname|org-name|owner|
         organization|descr|netname)$/ {print $2; exit}')"
    ctry="$(printf "%s\n" "$wf" | awk -F': *' 'tolower($1)~/^country$/ {print $2; exit}')"
    [ -z "$org" ] && org="-"; [ -z "$ctry" ] && ctry="-"
    ji="$(curl -m 3 -s "https://ipinfo.io/$ip" || true)"
    city="$(printf "%s" "$ji" | sed -n 's/.*"city"[[:space:]]*:[[:space:]]*"\([^"]*\)"
         .*/\1/p' | head -1)"
    region="$(printf "%s" "$ji" | sed -n 's/.*"region"[[:space:]]*:[[:space:]]*"\([^"]*\)"
         .*/\1/p' | head -1)"
    if [ -n "$city$region" ]; then loc="$city/$region"; else loc="-"; fi
  fi
  echo "$hop,$ip,$hn,$org,$ctry,$loc,$avg" >> "$OUT_CSV"
done
echo "flowchart LR" > "$OUT_MMD"
echo " classDef hop fill:#eef,stroke:#999,rx:10,ry:10;" >> "$OUT_MMD"
i=0
while IFS=, read -r hop ip hn org ctry loc avg; do
  [ "$hop" = "hop" ] && continue
  label="Hop ${hop}\n${ip}\n${hn}\n${org}\n${ctry} ${loc}\n${avg} ms"
  echo " N${hop}[\"$label\"]:::hop" >> "$OUT_MMD"
  if [ "$i" -gt 0 ]; then prev=$((hop-1)); echo " N${prev} --> N${hop}" >> "$OUT_MMD"; fi
  i=$((i+1))
done < "$OUT_CSV"
echo "Done"
echo "$OUT_CSV"
echo "$OUT_MMD"
```



route_kaggle.com

hop	ip	hostname	org	country	location	avg_ms
2	123.51.152.254	-	NCICNET-NET	TW	Taipei/Taiwan	8.700
3	220.228.20.157	-	NCICNET-TW	TW	Taipei/Taiwan	5.799
220.228.20.149	220.228.20.149	-	NCICNET-TW	TW	Taipei/Taiwan	9.880
4	192.72.107.189	h189-192-72-107.seed.net.tw	APNIC-ERX-192-72-3-0	AU	Taipei/Taiwan	12.735
192.72.107.97	192.72.107.97	h97-192-72-107.seed.net.tw	APNIC-ERX-192-72-3-0	AU	Taipei/Taiwan	6.585
192.72.107.189	192.72.107.189	h189-192-72-107.seed.net.tw	APNIC-ERX-192-72-3-0	AU	Taipei/Taiwan	5.589
5	139.175.56.141	r56-141.seed.net.tw	APNIC-ERX-139-175-0-0	AU	Taipei/Taiwan	7.672
139.175.57.185	139.175.57.185	r57-185.seed.net.tw	APNIC-ERX-139-175-0-0	AU	Taipei/Taiwan	6.026
6	139.175.58.202	r58-202.seed.net.tw	APNIC-ERX-139-175-0-0	AU	Taipei/Taiwan	8.155
139.175.59.163	139.175.59.163	r59-163.seed.net.tw	APNIC-ERX-139-175-0-0	AU	Taipei/Taiwan	7.132
139.175.59.167	139.175.59.167	r59-167.seed.net.tw	APNIC-ERX-139-175-0-0	AU	Taipei/Taiwan	5.625
7	139.175.59.217	r59-217.seed.net.tw	APNIC-ERX-139-175-0-0	AU	Taipei/Taiwan	7.201
8	142.250.172.86	-	GOOGLE	US	Taipei/Taiwan	5.718
35.244.233.98	35.244.233.98	98.233.244.35.bc.googleusercontent.com	GOOGLE-CLOUD	US	Kansas City/Missouri	5.899
142.250.172.86	142.250.172.86	-	GOOGLE	US	Taipei/Taiwan	5.771

Figure 7: kaggle’s traceroute record



```
flowchart LR
  classDef hop fill:#eef,stroke:#999,rx:10,ry:10;
  N2["Hop 2\n123.51.152.254\n-\nNCICNET-NET\nTW Taipei/Taiwan\n8.700 ms"]:::hop
  N3["Hop 3\n220.228.20.157\n-\nNCICNET-TW\nTW Taipei/Taiwan\n5.799 ms"]:::hop
  N2 --> N3
  N220.228.20.149["Hop 220.228.20.149\n220.228.20.149\n-\nNCICNET-TW\nTW Taipei/Taiwan\n9.880 ms"]:::hop
  N3 --> N220.228.20.149
  N4["Hop 4\n192.72.107.189\nh189-192-72-107.seed.net.tw\nAPNIC-ERX-192-72-3-0\nAU Taipei/Taiwan\n12.735 ms"]:::hop
  N220.228.20.149 --> N4
  N192.72.107.97["Hop 192.72.107.97\n192.72.107.97\nh97-192-72-107.seed.net.tw\nAPNIC-ERX-192-72-3-0\nAU Taipei/Taiwan\n6.585 ms"]::
  N4 --> N192.72.107.97
```

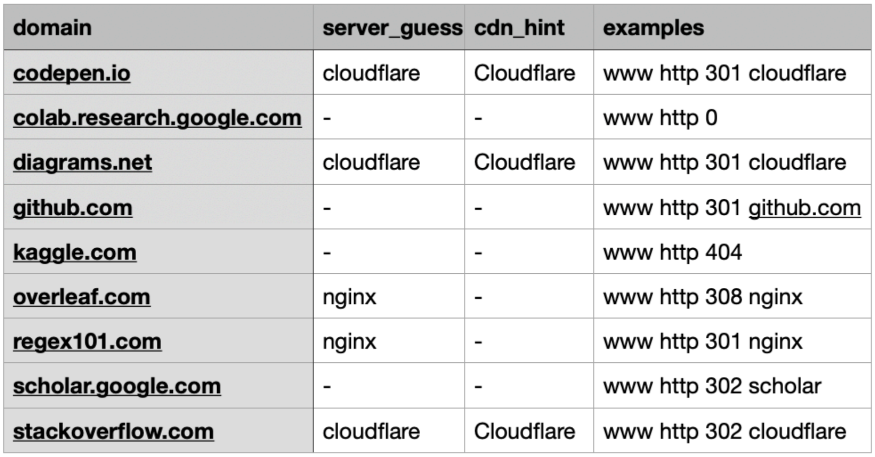
Figure 8: kaggls’s route diagram（節錄）

(7) Backend Server Investigation

我們以 curl -I -L 擷取各網站最終回應標頭（follow redirects），從 Server、X-Powered-By、Via 與常見雲端／CDN 指紋（如 CF-Ray, CloudFront, Fastly, gws, ESF 等）判斷背後的 Web 伺服器與雲邊緣層。流程自動化：逐一對清單中的網域發請求，解析標頭欄位並彙整為 CSV，比對「伺服器核心（Nginx/Apache/LiteSpeed/OpenResty 等）」與「CDN ／雲端代理（Cloudflare、Google Frontend、Akamai、CloudFront、Vercel 等）」兩個層面。

```
#!/usr/bin/env bash
set -euo pipefail
DOMAINS_FILE="${DOMAINS_FILE:-domains.txt}"
OUT_RAW="data/backend_raw.csv"
OUT_SUM="data/backend_summary.csv"
TIMEOUT="${TIMEOUT:-8}"
UA="${UA:-Mozilla/5.0}"
SCHEMES="${SCHEMES:-https http}"
VARIANTS="${VARIANTS:-apex www}"
echo "domain,variant,scheme,status,server,via,x_powered_by,cdn_hint,app_hint" > "$OUT_RAW
    "
fetch(){
  u="$1"; h="$(curl -m "$TIMEOUT" -A "$UA" -sS -I -L -k "$u" -D - -o /dev/null || true)"
  st="$(printf "%s" "$h" | awk 'BEGIN{RS="\r?\n\r?\n"} NR==1{print}' | awk 'toupper($0) ~
      /^HTTP/{code=$2} END{print code+0}')"
  sv="$(printf "%s" "$h" | awk -F': *' 'BEGIN{IGNORECASE=1} tolower($1)=="server"{print
     $2; exit}')"
  via="$(printf "%s" "$h" | awk -F': *' 'BEGIN{IGNORECASE=1} tolower($1)=="via"{print $2;
      exit}')"
  xp="$(printf "%s" "$h" | awk -F': *' 'BEGIN{IGNORECASE=1} tolower($1)=="x-powered-by"{
     print $2; exit}')"
  cf="$(printf "%s" "$h" | awk -F': *' 'BEGIN{IGNORECASE=1} tolower($1)=="cf-ray"{print "
     Cloudflare"; exit}')"
  ak="$(printf "%s" "$h" | awk -F': *' 'BEGIN{IGNORECASE=1} tolower($1)~/^server-timing$
     /{if(tolower($2)~/(ak_)/) print "Akamai"}')"
  fa="$(printf "%s" "$h" | awk -F': *' 'BEGIN{IGNORECASE=1} tolower($1)=="x-served-by"{if
     (tolower($2)~/(fastly)/) print "Fastly"}')"
  ec="$(printf "%s" "$h" | awk -F': *' 'BEGIN{IGNORECASE=1} tolower($1)~/^(x-ec-)/{print
     "Edgio_Limelight"; exit}')"
  lb="$(printf "%s" "$h" | awk -F': *' 'BEGIN{IGNORECASE=1} tolower($1)=="server"{if(
     tolower($2)~/(gws|tsa)/) print "Google"; else if(tolower($2)~/(cloudfront)/) print
     "AWS_CloudFront"; else if(tolower($2)~/(azure)/) print "Azure"}')"
  ch="${cf:-}${ak:+;$ak}${fa:+;$fa}${ec:+;$ec}${lb:+;$lb}"; ch="$(printf "%s" "$ch" | sed
      's/^;*//;s/;*$//' | tr ';' '\n' | sort -u | paste -sd ';' -)"
  app="$(printf "%s" "$sv;$xp" | tr '[:upper:]' '[:lower:]')"
  app=$(printf "%s" "$app" | awk -F';' '{for(i=1;i<=NF;i++){s=$i; if(s~/nginx/) a="nginx"
     ; if(s~/apache/) a=(a?a";":"")"apache"; if(s~/litespeed/) a=(a?a";":"")"litespeed";
      if(s~/iis/) a=(a?a";":"")"microsoft_iis"; if(s~/gws/) a=(a?a";":"")"google_gws";
     if(s~/tsa/) a=(a?a";":"")"google_tsa"; if(s~/cloudflare/) a=(a?a";":"")"cloudflare"
```

```
      ; if(s~/cloudfront/) a=(a?a";":"")"aws_cloudfront"; if(s~/ats|apache traffic server
      /) a=(a?a";":"")"apache_traffic_server"}; print a; exit}')
  echo "$st|$sv|$via|$xp|$ch|$app"
}
run_one(){
  d="$1"; v="$2"; s="$3"; t="$d"; [ "$v" = "www" ] && t="www.$d"; u="$s://$t/"
  r="$(fetch "$u")"; st="${r%%|*}"; rest="${r#*|}"; sv="${rest%%|*}"; rest="${rest#*|}";
      vi="${rest%%|*}"; rest="${rest#*|}"; xp="${rest%%|*}"; rest="${rest#*|}"; cdn="${
      rest%%|*}"; app="${rest#*|}"
  echo "$d,$v,$s,${st:-},${sv:-},${vi:-},${xp:-},${cdn:-},${app:-}" >> "$OUT_RAW"
}
while IFS= read -r d; do
  [ -z "${d// }" ] && continue
  case "$d" in \#*) continue;; esac
  for v in $VARIANTS; do for s in $SCHEMES; do run_one "$d" "$v" "$s"; done; done
done < "$DOMAINS_FILE"
awk -F, 'BEGIN{OFS=","; print "domain,server_guess,cdn_hint,examples"}
NR>1{
  k=$1
  if($9!=""){split($9,a,/;/); for(i in a) app[k":"a[i]]=1}
  if($8!=""){split($8,c,/;/); for(i in c) cdn[k":"c[i]]=1}
  ex[k]=$2" " $3 " " $4 " " $5
  seen[k]=1
}
END{
  for(d in seen){
    sg=""; sep=""
    for(x in app){ n=split(x,p,":"); if(p[1]==d){ sg=sg sep p[2]; sep=";"} }
    cg=""; sep=""
    for(x in cdn){ n=split(x,p,":"); if(p[1]==d){ cg=cg sep p[2]; sep=";"} }
    if(sg=="") sg="-"; if(cg=="") cg="-"
    print d,sg,cg,ex[d]
  }
}' "$OUT_RAW" | sort -t, -k1,1 > "$OUT_SUM"
echo "Done:"
echo " - $OUT_RAW"
echo " - $OUT_SUM"
```



domain	server_guess	cdn_hint	examples
codepen.io	cloudflare	Cloudflare	www http 301 cloudflare
colab.research.google.com	-	-	www http 0
diagrams.net	cloudflare	Cloudflare	www http 301 cloudflare
github.com	-	-	www http 301 github.com
kaggle.com	-	-	www http 404
overleaf.com	nginx	-	www http 308 nginx
regex101.com	nginx	-	www http 301 nginx
scholar.google.com	-	-	www http 302 scholar
stackoverflow.com	cloudflare	Cloudflare	www http 302 cloudflare

Figure 9: route diagram