ECMP stability (Paris-style):

After running my differences script (see appendix) I get the following output that compares two different traceroute outputs to the same end host using the jsonl output.

The tracediff.py script computed a Jaccard similarity of 1.0000 between the two runs (--flow-id 101 vs. --flow-id 202), indicating the set of router IPs at every hop was identical. This result demonstrates that for this specific path to 8.8.8.8, changing the flow ID did not trigger any Equal-Cost Multi-Path (ECMP) load balancing, as the path remained perfectly stable.

Path Change detector:

After adding one new argument (save_path) to mytrace.py that saves a new jsonl file with the visited path for a tracerout, I then ran two traces to the same host (google.com) in different times of the day to see traces under different network conditions.

I then used the script in the appendix to comput the Jaccard similarity between the two runs. The similarity was equal to 1, indicating that the paths taken were identical. This similarity of 1.0 indicates that the path from my location to google.com is highly stable, showing no route changes even when tested at different times of the day under potentially different network load conditions

```
➜  proj3 git:(main) ✗ python3 tracediff.py trace_google_com_1763318114.jsonl
trace_google_com_1763321948.jsonl
Path Change Detector (Overall Similarity)
Overall Jaccard Similarity: 1.0000
Paths appear identical.

ECMP Stability Analysis (Per-Hop)
Hop  1: Similarity=1.0000
Hop  2: Similarity=1.0000
Hop  3: Similarity=1.0000
Hop  4: Similarity=1.0000
Hop  5: Similarity=1.0000
Hop  6: Similarity=1.0000
Hop  7: Similarity=1.0000
Hop  8: Similarity=1.0000
Hop  9: Similarity=1.0000
Hop 10: Similarity=1.0000
Hop 11: Similarity=1.0000
```

APPENDIX

```python
import json
import argparse
from collections import defaultdict
import sys

def jaccard_similarity(set1, set2):
   """Calculates the Jaccard similarity between two sets."""
   intersection = len(set1.intersection(set2))
   union = len(set1.union(set2))
   return intersection / union if union > 0 else 1.0

def load_hop_ips(filepath):
   """
   Reads a trace JSONL file and returns:
   1. A dict mapping {hop: {set of IPs}}
   2. An overall set of all unique IPs in the path
   """
   hops = defaultdict(set)
   all_ips = set()

   try:
       with open(filepath, 'r') as f:
           for line in f:
               try:
                   data = json.loads(line)
                   ip = data.get('router_ip')
                   if ip:
                       hop_num = data.get('hop')
                       hops[hop_num].add(ip)
                       all_ips.add(ip)
               except json.JSONDecodeError:
                   print(f"Warning: Skipping bad JSON line in {filepath}")
   except FileNotFoundError:
       print(f"Error: File not found: {filepath}")
       return None, None

   return hops, all_ips

def main():
   parser = argparse.ArgumentParser(description="Compare two traceroute JSONL files.")
```

```python
    parser.add_argument("file1", help="First traceroute JSONL file")
    parser.add_argument("file2", help="Second traceroute JSONL file")
    args = parser.parse_args()

    hops1, all_ips1 = load_hop_ips(args.file1)
    hops2, all_ips2 = load_hop_ips(args.file2)

    if hops1 is None or hops2 is None:
        sys.exit(1)

    print("Path Change Detector (Overall Similarity)")
    overall_sim = jaccard_similarity(all_ips1, all_ips2)
    print(f"Overall Jaccard Similarity: {overall_sim:.4f}")
    if overall_sim < 1.0:
        print("Difference detected!")
        print(f"  Path 1 unique IPs: {all_ips1 - all_ips2}")
        print(f"  Path 2 unique IPs: {all_ips2 - all_ips1}")
    else:
        print("Paths appear identical.")

    print("\nECMP Stability Analysis (Per-Hop) ")
    all_hops = set(hops1.keys()).union(set(hops2.keys()))

    for hop in sorted(all_hops):
        ips1 = hops1.get(hop, set())
        ips2 = hops2.get(hop, set())

        sim = jaccard_similarity(ips1, ips2)

        # Check for ECMP (more than 1 IP at a single hop)
        ecmp_str = ""
        if len(ips1) > 1 or len(ips2) > 1:
            ecmp_str = f"[ECMP Detected: |{len(ips1)}| vs |{len(ips2)}| IPs]"

        print(f"Hop {hop:2d}: Similarity={sim:.4f}  {ecmp_str}")
        if sim < 1.0:
            print(f"  └ Hop {hop} changed: {ips1} vs {ips2}")

if __name__ == "__main__":
    main()
```