Important note

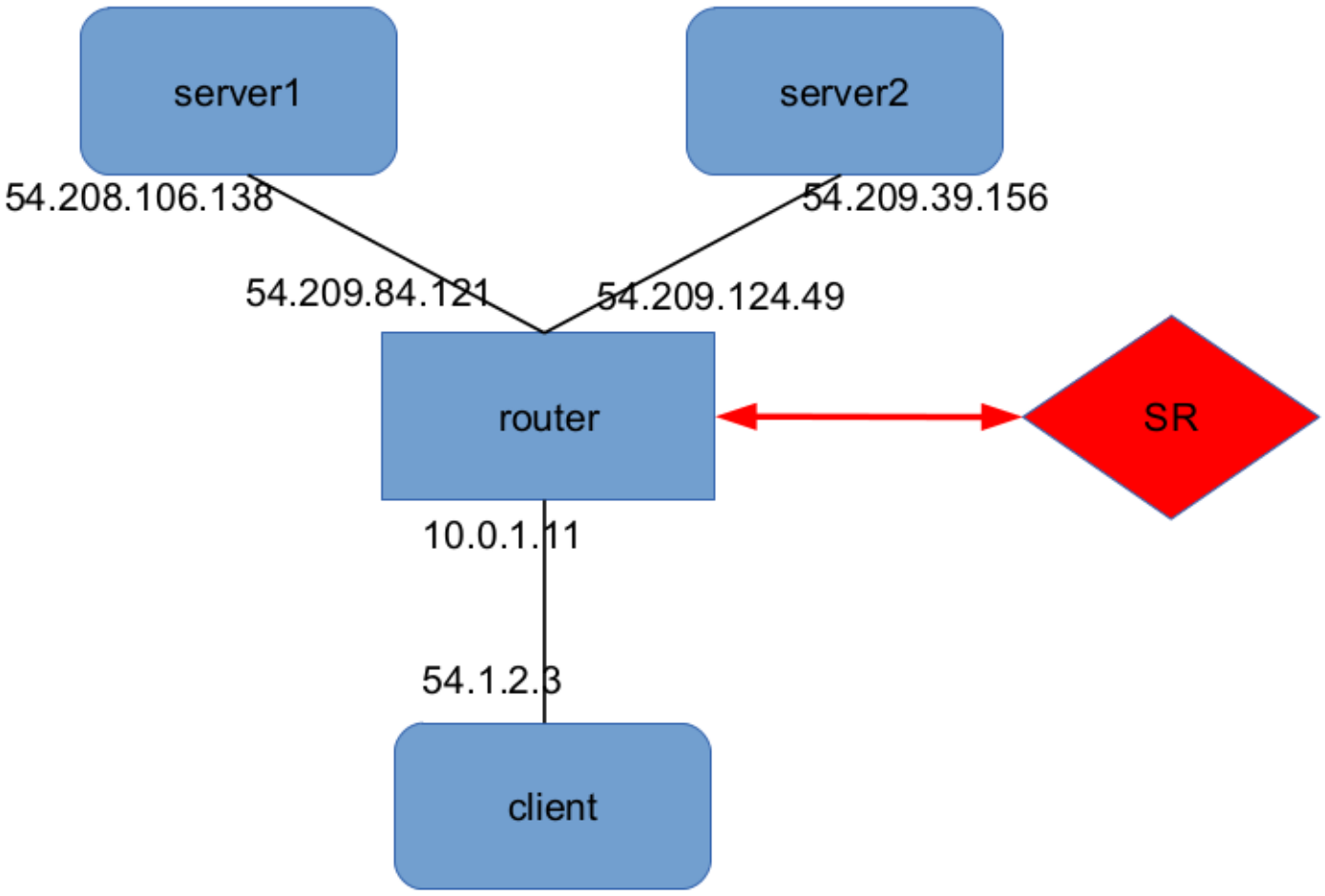
The test descriptions below remain vague on purpose. For example, we do not specify which of the router's interfaces receives the incoming packet, or which source IP / destination IP we are using. You can, however, find out by looking at the .pcap file included in the feedback email. This .pcap file is a capture of all the packets sent and received by your router during the phase 1 of the tests (read below to understand what phase 1 is). Use Wireshark to read the packet capture.

To help you in your debugging, we provide a **reference .pcap file**, i.e. the .pcap produced by the reference solution. You can download it here.

Note that for each of your submissions, the exact same tests will be performed in the exact same order. However, your code should be generic and should not make any specific assumptions, other than those already made in the starter code.

Network topology and routing table

The testing environment we use is slightly different from the one you have on your own VM. The IP addresses are different, and we had to add some dummy entries to the routing table to test for error cases (Host / Net Unreachable):



```
8.8.8.8 8.8.8.8 255.255.255.255 eth2
10.0.0.0 10.0.1.1 255.255.0.0 eth3
54.0.0.0 10.0.1.1 255.0.0.0 eth3
54.208.106.138   54.208.106.138 255.255.255.255  eth1
54.208.106.0 54.208.106.138 255.255.255.254 eth3
54.209.39.156   54.209.39.156 255.255.255.255  eth2
54.209.39.0 54.209.39.156 255.255.255.254 eth3
```

Description of the tests

We have 21 different tests. We run 2 phases of tests (r1 and r2). For each phase we run the 21 tests, so each test is run twice (once is phase r1 and once in phase r2). In phase r1, we do not restart your router between tests, which means that if your router segfaults during one test, it will automatically fail all subsequent tests. In phase r2, on the other hand, the router is restarted after each test, which means that if you segfault or if your router is storing bad state, it should not affect subsequent tests.

All tests are worth the same amount of points.

In the feedback email you have received, you will notice that half the test names have the suffix '-r1' and the other half have the suffix '-r2'. Suffix '-r1' means that this is the phase 1 result and suffix '-r2' means phase 2.

- *init*: untars the archive and starts grader
- *compile*: compiles the code by running 'make'
- *ICMP-Shows-ARP-1*: verifies that when the router receives an ICMP echo request, it initiates an ARP request, instead of simply swapping source IP and destination IP in the reply.
- *ARP-Request-[1-2]*: sends an ARP request to one of the router's interfaces and waits for a matching ARP reply.
- *ARP-Caching-1*: sends two successive -identical- ICMP echo requests to one of the router's interfaces, and makes sure that only the first one triggers an ARP request. Notes: 1) We make sure that this is the first time that the router has to send an IP packet to this address, thus making an ARP request necessary. 2) The second ICMP echo request is only sent after the first ICMP echo reply has been sent by the router, so there is no race condition with the ARP request / reply.
- *ICMP-Request-Forwarding-[1-2]*: sends an ICMP echo request to the router. This request is not addressed to the router, which needs to forward it (the route is known). We check that the forwarded packet is identical to the original one, apart from the modified IP header.
- *ICMP-Reply-Forwarding-1*: same as above but the packet forwarded is now an ICMP echo reply instead of an ICMP echo request.
- *ICMP-Reply-[1-2]*: checks that the router correctly replies to ICMP echo requests addressed to one of its interfaces. This is different from earlier tests (e.g. ICMP-Shows-ARP-1), which was only interested in the ARP packets involved.
- *ICMP-TTL-Decrease-1*: sends an IP packet to the router (in our case an ICMP echo request) for forwarding and makes sure that the router correctly decrements the TTL field.
- *ICMP-Reply-TTL1-1*: sends an ICMP echo request addressed to one of the router's interfaces with a TTL of 1, and verifies that the router is generating a reply.
- *TCP-Forwarding-1*: same as ICMP-Request-Forwarding and ICMP-Reply-Forwarding,

but this time with a TCP packet instead of an ICMP packet. In other words, checks that the router correctly forwards TCP traffic.
- *UDP-Forwarding-1*: same as above but this time we send a UDP packet instead of a TCP one.
- *Port-Unreachable-[1-2]*: sends a TCP packet addressed to one of the router's interfaces, expects an ICMP port unreachable message.
- *Port-Unreachable-3*: same as above but with a UDP packet.
- *Net-Unreachable-1*: sends an ICMP packet to the router for forwarding, but makes sure that the router has no entry for the IP address. Expects an ICMP net unreachable message.
- *Net-Unreachable-2*: same as above but with a TCP packet instead of an ICMP one.
- *Host-Unreachable-1*: sends an ICMP packet to the router for forwarding. The router has a matching entry for the destination IP in its routing table, but the next hop is not replying to ARP requests. Expects an ICMP host unreachable message after roughly 5 seconds (5 timed out ARP requests).
- *Time-Exceed-1*: sends an ICMP packet to the router for forwarding, but with a TTL of 1. The router should drop the packet and generate an ICMP Time Exceed message.
- *Time-Exceed-2*: same as above but with a UDP packet instead of an ICMP one.

FAQ

- My code is failing most of the tests and I don't see why.

Check your checksums in the .pcap file with Wireshark. The grader systematically dismisses packets with a wrong IP checksum or a wrong ICMP checksum.