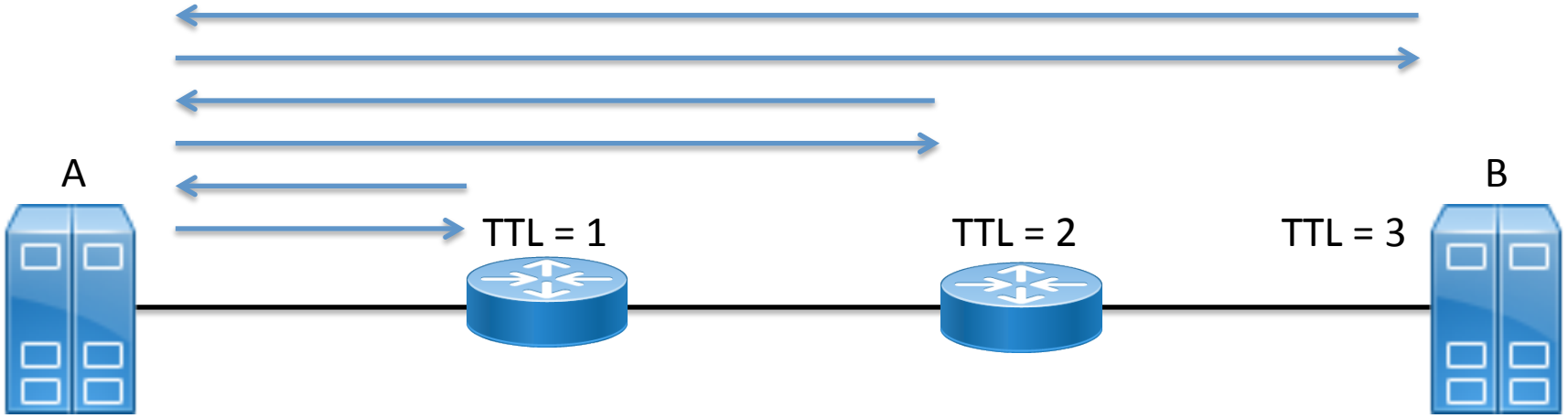
Reverse Traceroute: A surprisingly hard problem for something so simple

Rolf Winter

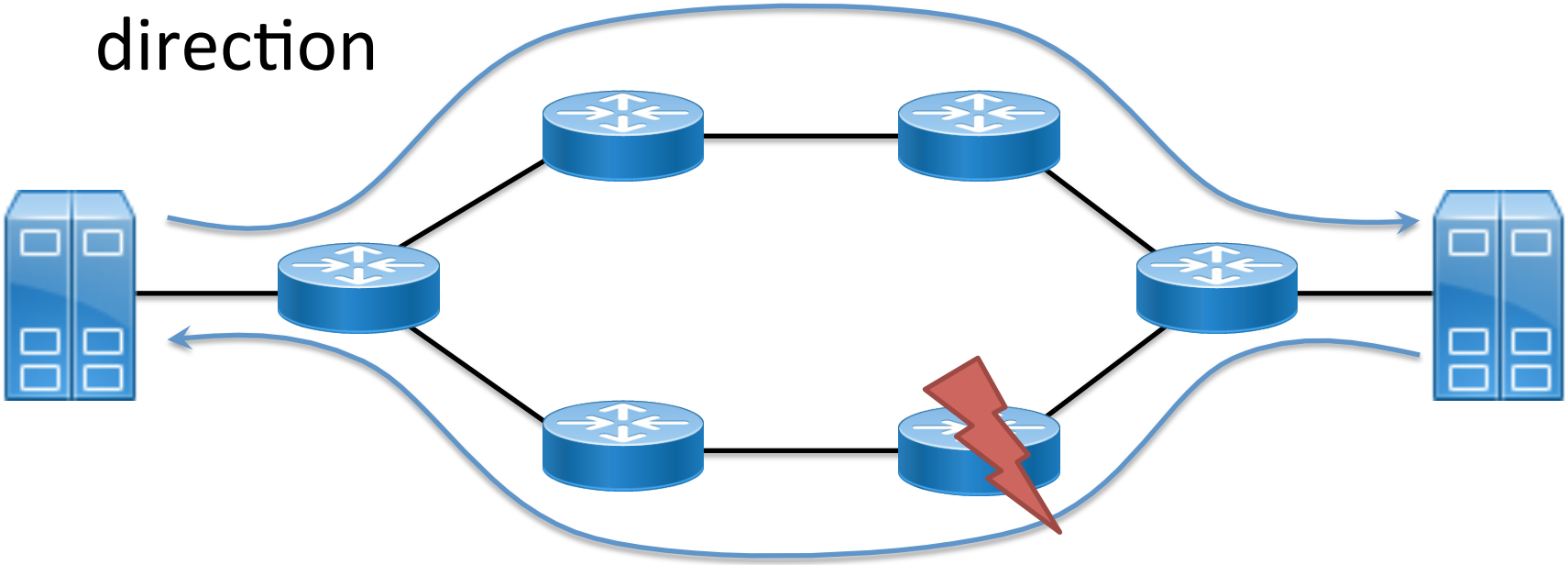
TR in the early Internet

- Most paths on the Internet were symmetric
- → Reverse path known, when the forward path is known
- → no need for a reverse TR



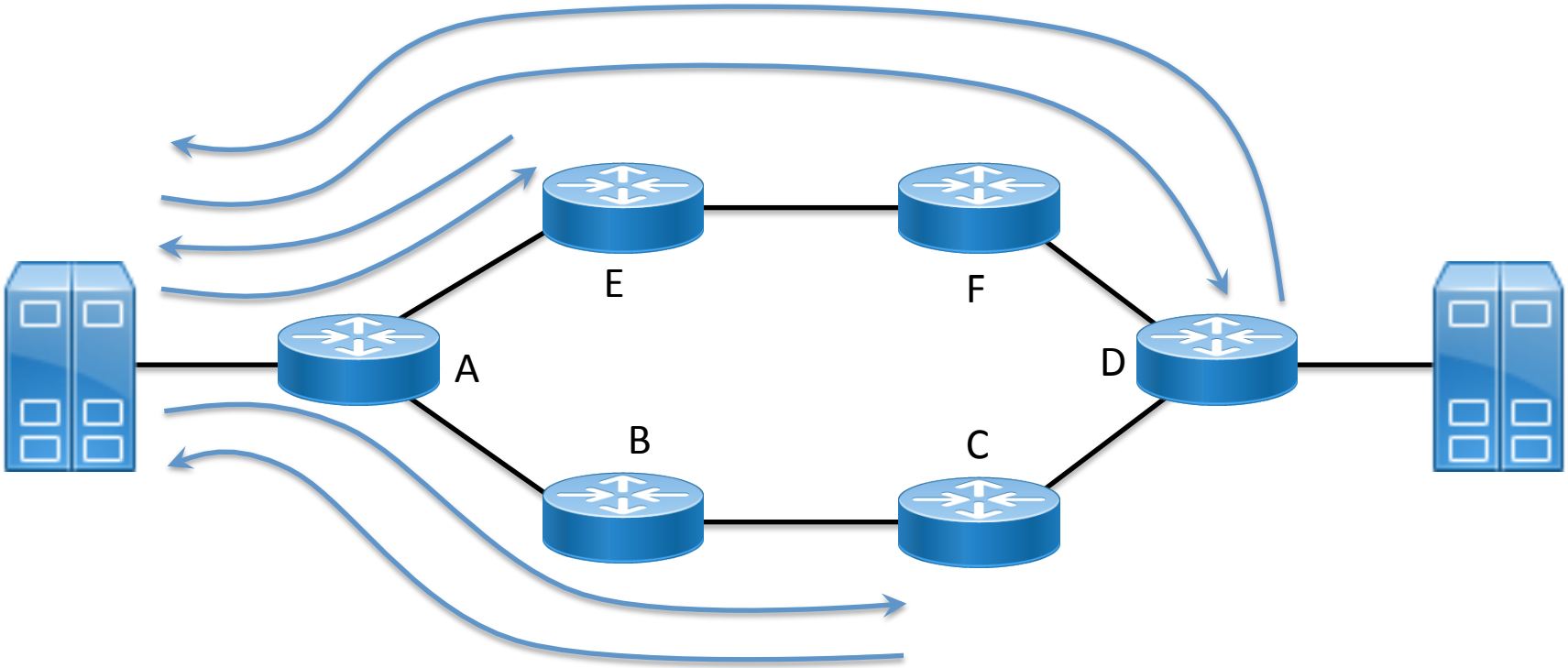
The Internet is more complex today

- Paths are largely asymmetric
- There is (likely) no one path between two points on the Internet
- Problems might be on the “invisible” reverse direction



… and paths are load-balanced, too

- Resulting TR path segment A-E-C-D does not exist!



Requirements for rTR - I

- It has to work without control over the host performing the reverse traceroute
  - What makes ping and traceroute so successful, is that these tools work without control over the host replying to the messages sent
  - It should be implemented as part of the OS, so that it is universally available and no additional software needs to be constantly running on a host

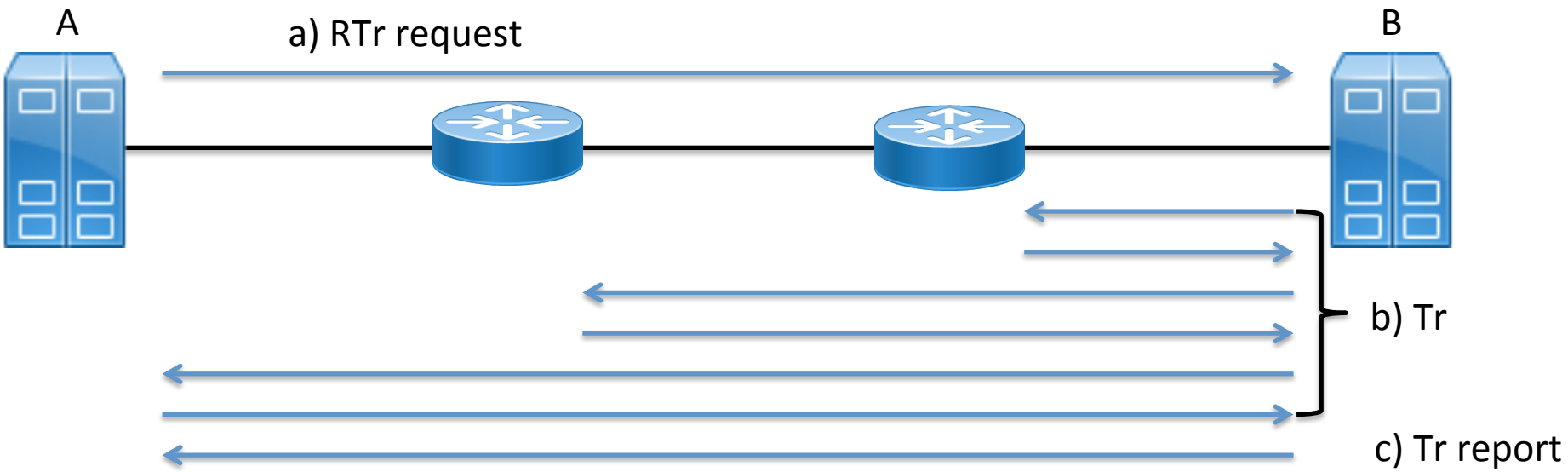
Requirements for rTR - I

- Reverse Traceroute has to “handle” asymmetric, load-balanced paths
  - Because they are a reality
- Reverse Traceroute has to be able to both infer the path and measure RTTs

rTR – first try

a) Host A sends rTR Message to host B
b) B performs a TR
c) B sends a TR report (hops and RTT) to A

No option!
- Attack vector
- Stateful



Cons:
- Amplification Attacks made easy
- B needs to keep state
- The Tr report could be > 1 pkt

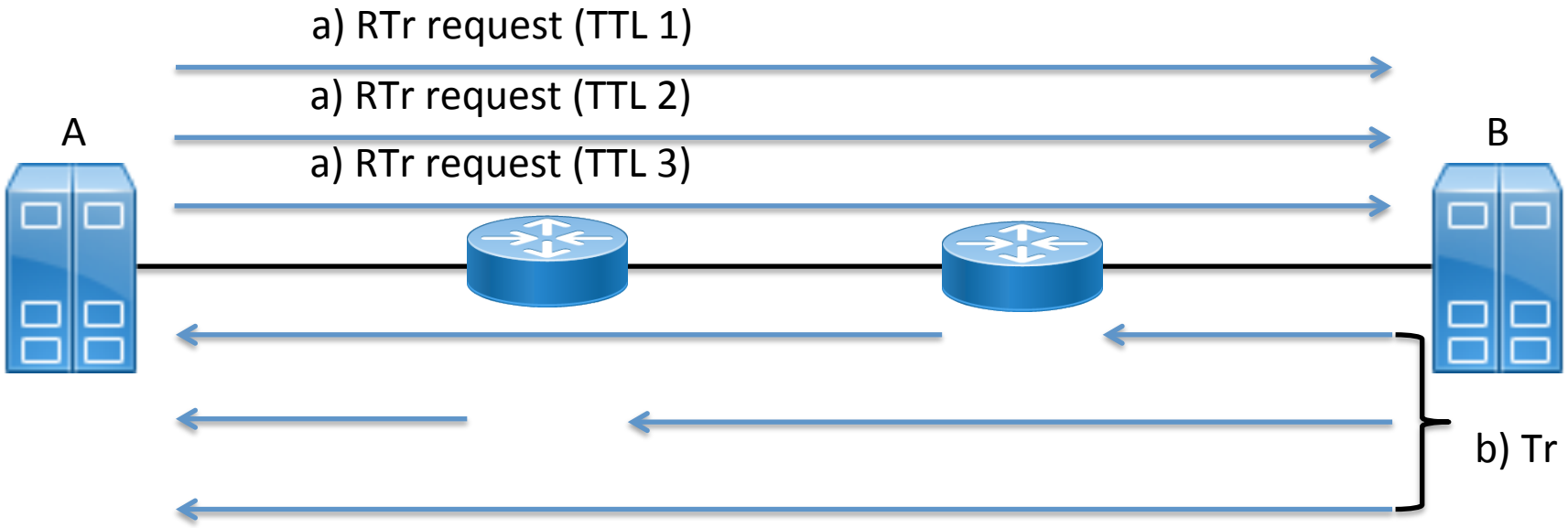
Requirements for rTR - II

- Reverse Traceroute should not be usable as a DoS tool, neither for the host nor for the
  - In order to protect the host, reverse traceroute should be stateless
  - It should not send large amounts of packets for a single packet (protect the network from amplification attacks).
- Not fulfilling these requirements will likely lead to filtering of reverse traceroute messages as it will be seen as a security threat by network operators.

rTR – second try

a) Host A sends RTr Message for each hop to host B
b) B performs a Tr spoofing the sender to be

Candidate, but
- Spoofing (yuk)



Cons:
- IP address spoofing (BCP 38)
- No explicit RTT information (but implicit by calculating RTT variations)

Improvements:
- One Request per TTL-probe (DoS protection)
- IP address spoofing (reduces amounts of packets, plus stateless)

Requirements for rTR - III

- Reverse Traceroute should not rely on "hackery" or features that are not universally accepted or deployed
  - Source address spoofing, IP options, extension headers are not part of the solution

rTR – third try

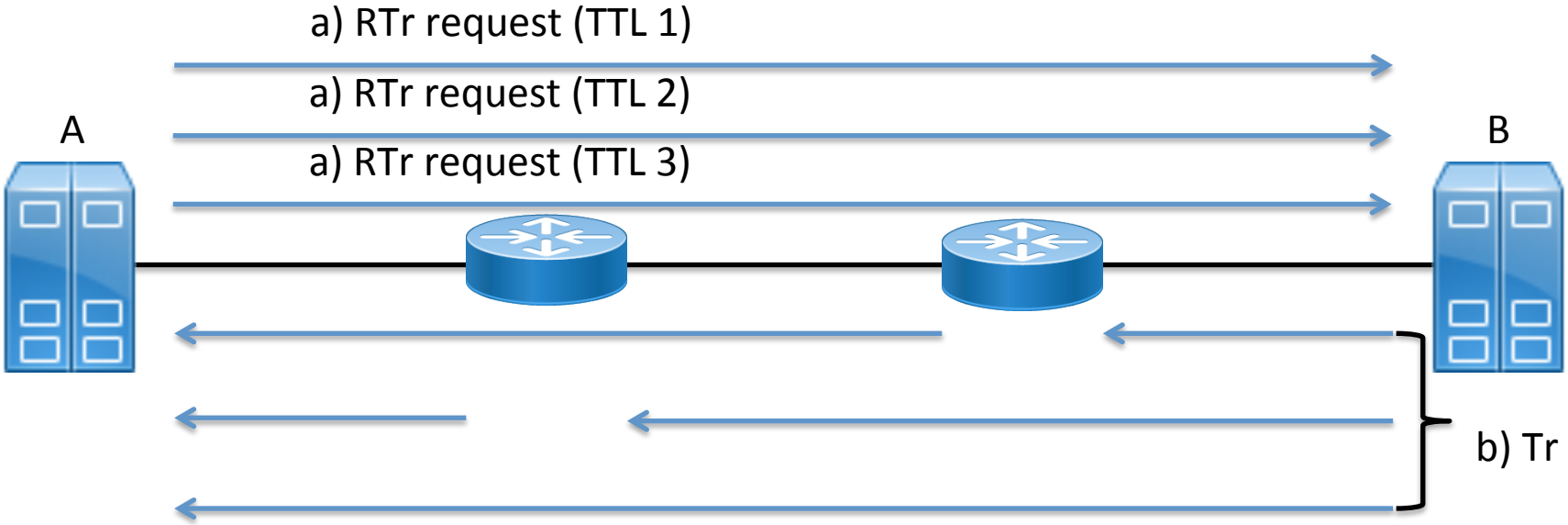
Candidate, but
- Router changes

Reverse Traceroute option B3:

a) Host A sends RTr Message for each hop to host B

b) B performs TR and includes original sender in msg (not spoofing, in payload)

c) Router extract orig source and send the response



Cons:
- Requires router modifications
- No explicit RTT information (but implicit by calculating RTT variations)

Improvement:
- No spoofing, but original address inside of the ICMP message

Requirements for rTR - IV

- Routers should remain untouched
  - Things will become much more difficult if routers are involved (deployment)

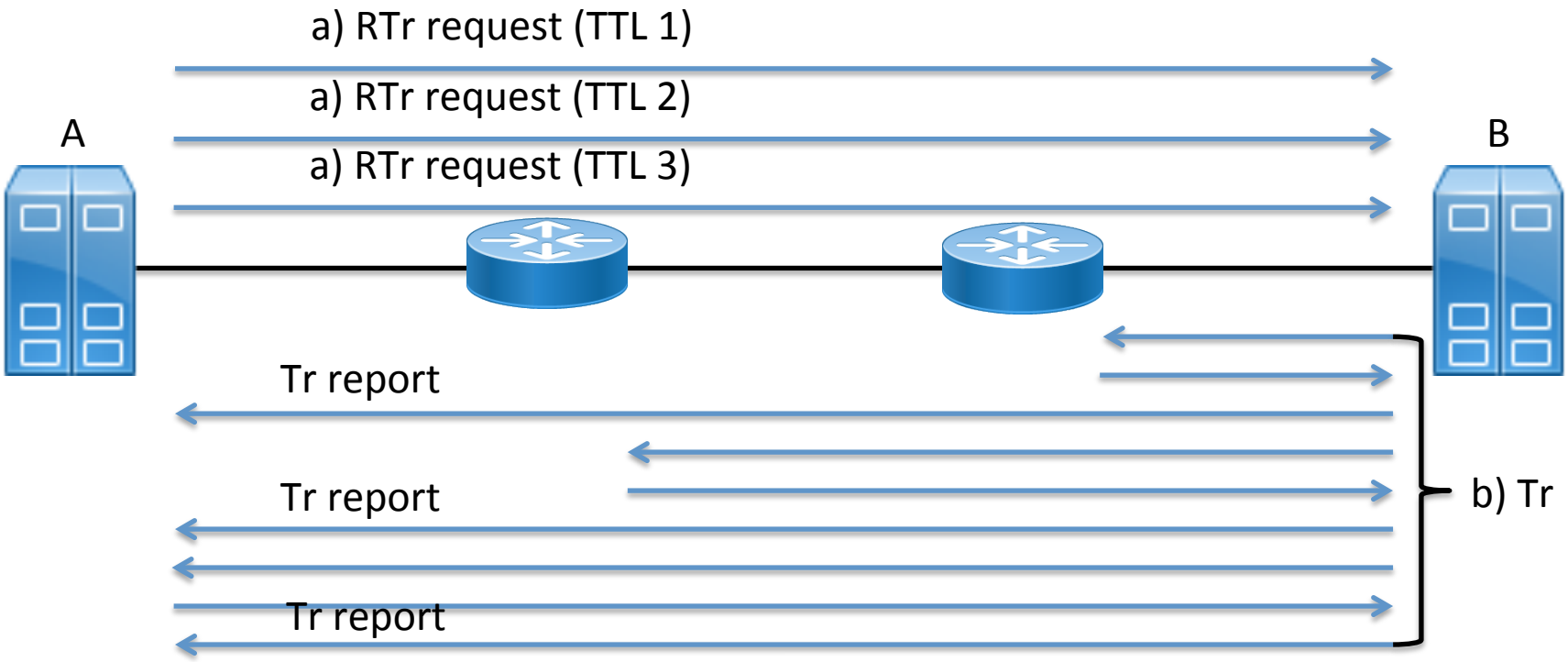
rTR – fourth try

Reverse Traceroute option C:

a) Host A sends RTr Message for each hop to host B

b) B performs a Tr spoofing and sends a report per hop

Candidate, but
- Heavier on host B



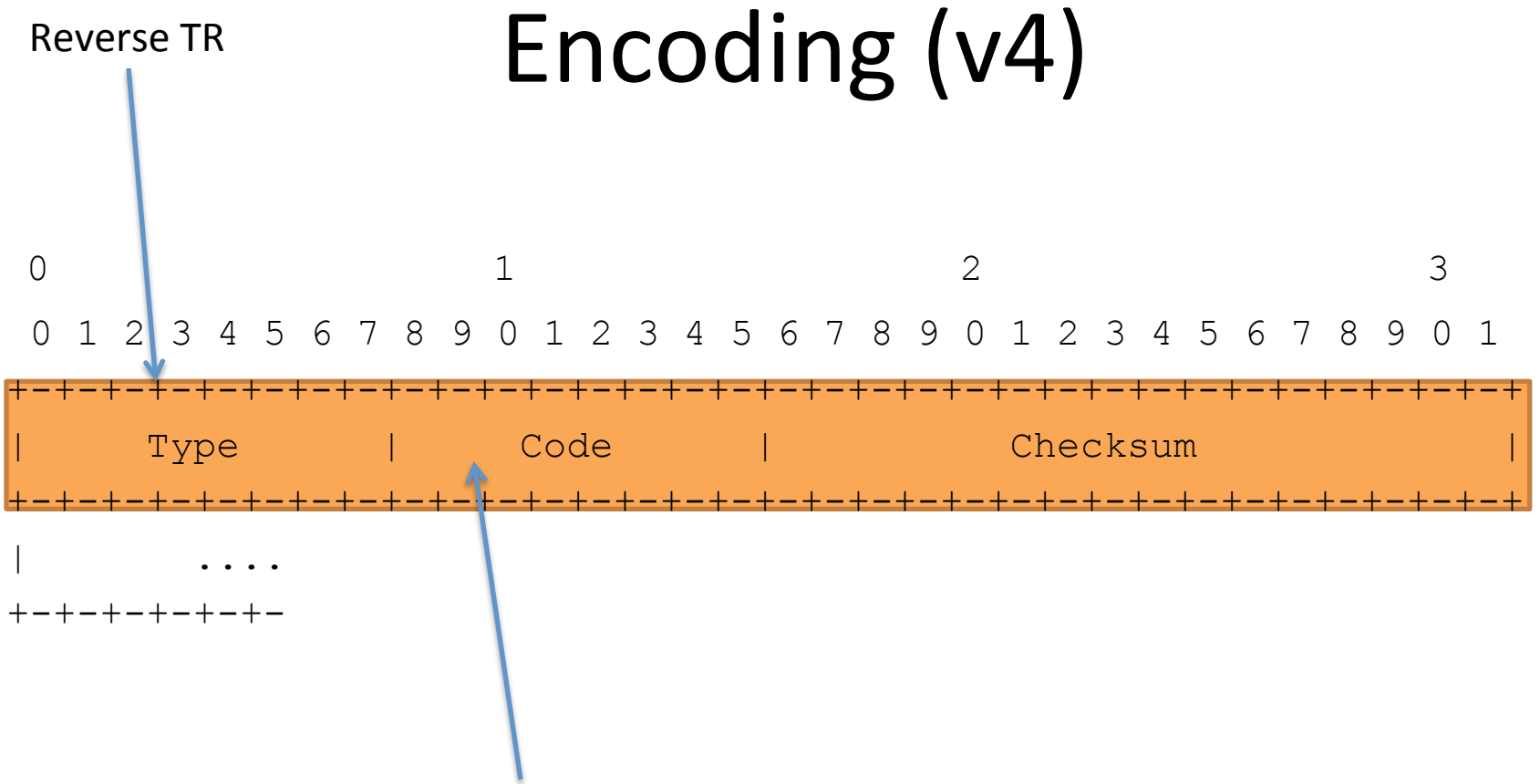
Cons:
- Slightly more work for B
- Could be stateful for B (depends on implementation)

Improvement:
- Works without router support

Requirements for rTR - V

- Reverse Traceroute should be easily policable if a network operator choses to do so
  - Really want to be friendly to network operations
  - Should use new ICMP types so that they can be filtered/policed at the edge of a network or by the end systems.

Encoding (v4)



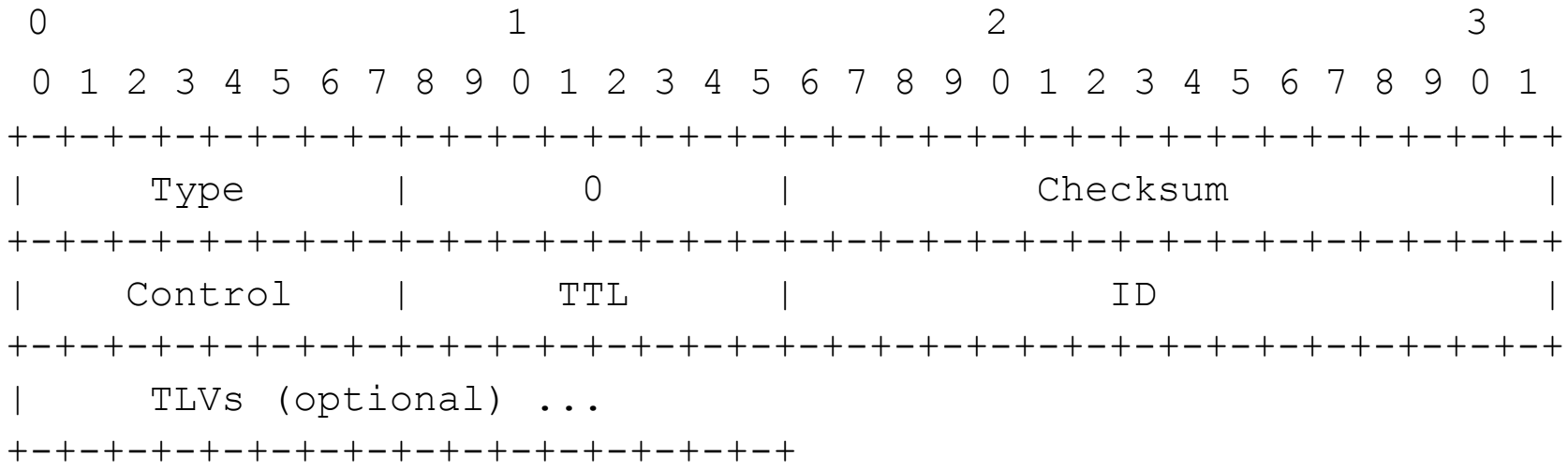
Message type:

- Request (Code 0)
- Probe (Code 1)
- Report (Code 2)

Requirements for rTR - VI

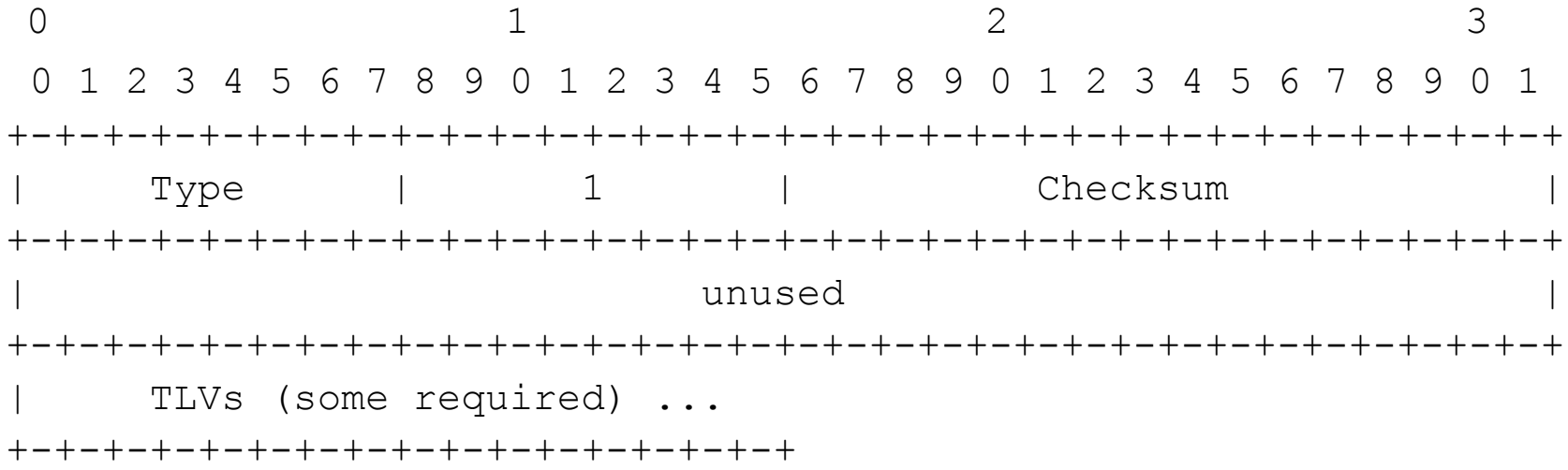
- Since load balancing practices might change or features might be needed in the future, rTR should be extensible
  - Use TLVs inside the ICMP message body
  - Leave some reserved bits inside the ICMP header, where it make sense

More Encoding (v4) - Request



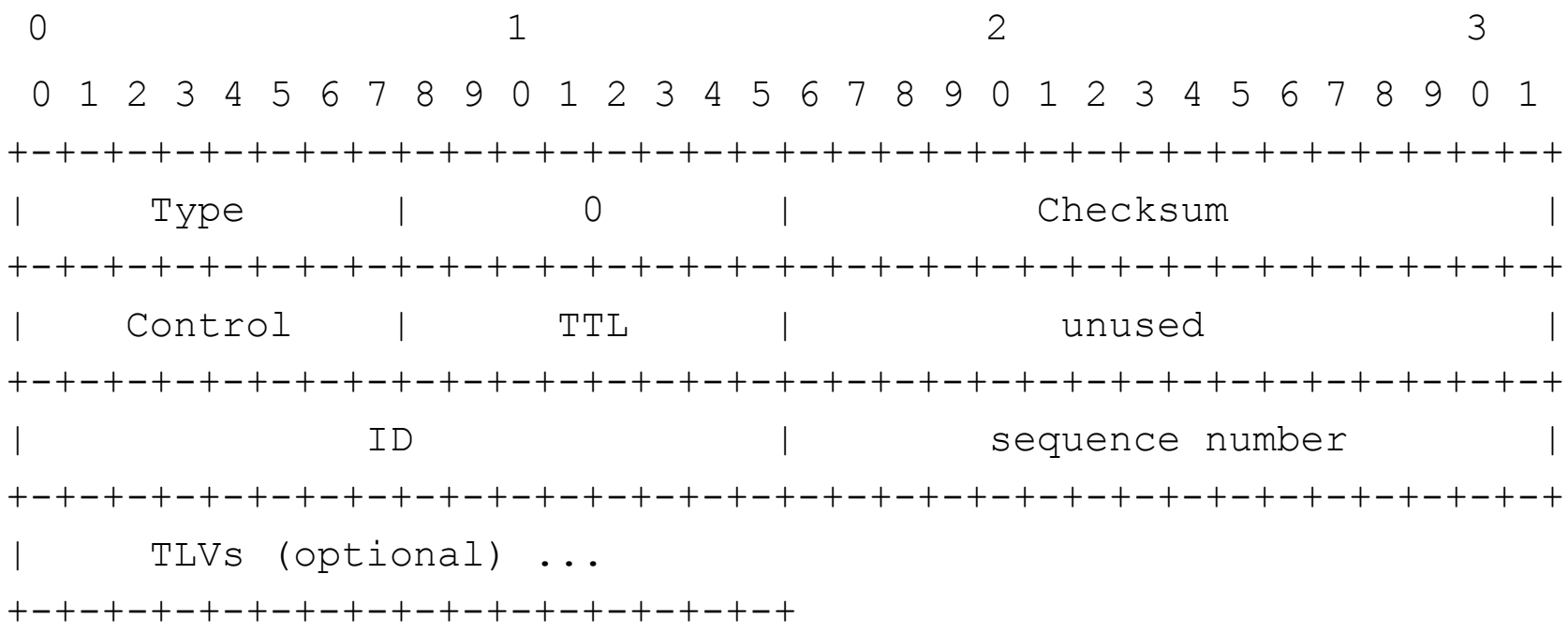
- Control bits could be (request load balancing control for TR, number of TTL probes requested…), TTL is the requested TTL for the TR packet, ID set by the requestor and echoed in the report, seq number set by the node performing the rTR (if multiple probes were sent)

More Encoding (v4) - Probe



- TLVs: timestamp, original requestors IP (to potentially extend rTR in the future), ID, seq-number – everything to make this stateless

Even more Encoding (v4) - Report



- TLV: one report TLV per probe sent