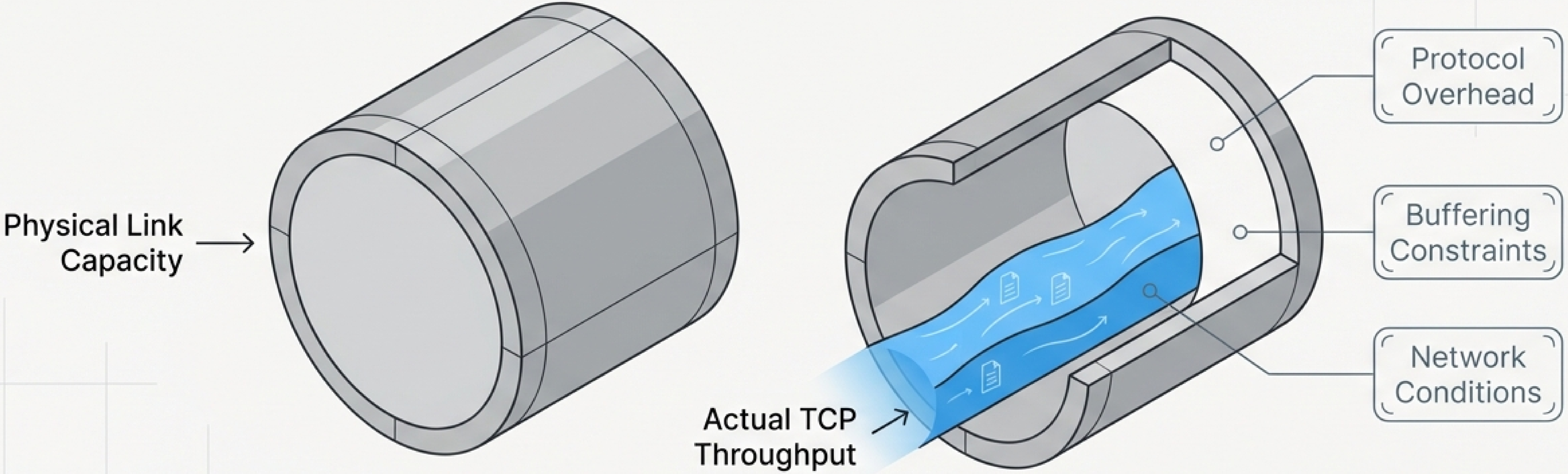
TCP Throughput Validation with iperf3 & RFC 6349

A Methodology for Measuring End-to-End Network Performance

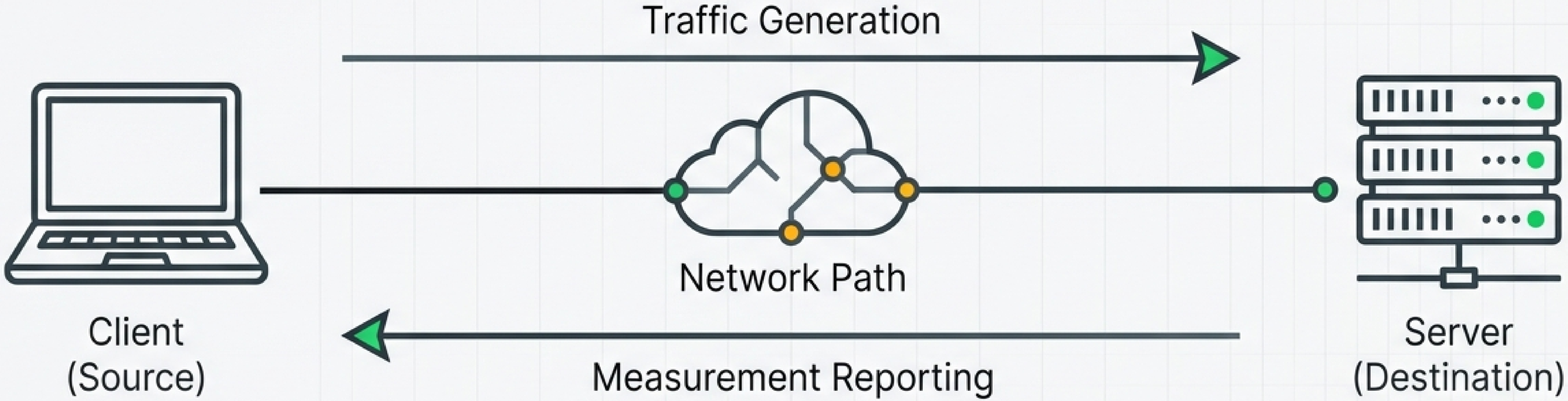
Engineered by Pras & Shreya

Link ‘Up’ ≠ Performance Verified



We are not testing the cable. We are validating the service delivered to the application.

The Architecture of iperf3



Definition

Open-source industry standard for active bandwidth measurement.

Mechanism

Client generates traffic; Server absorbs and measures.

Versatility

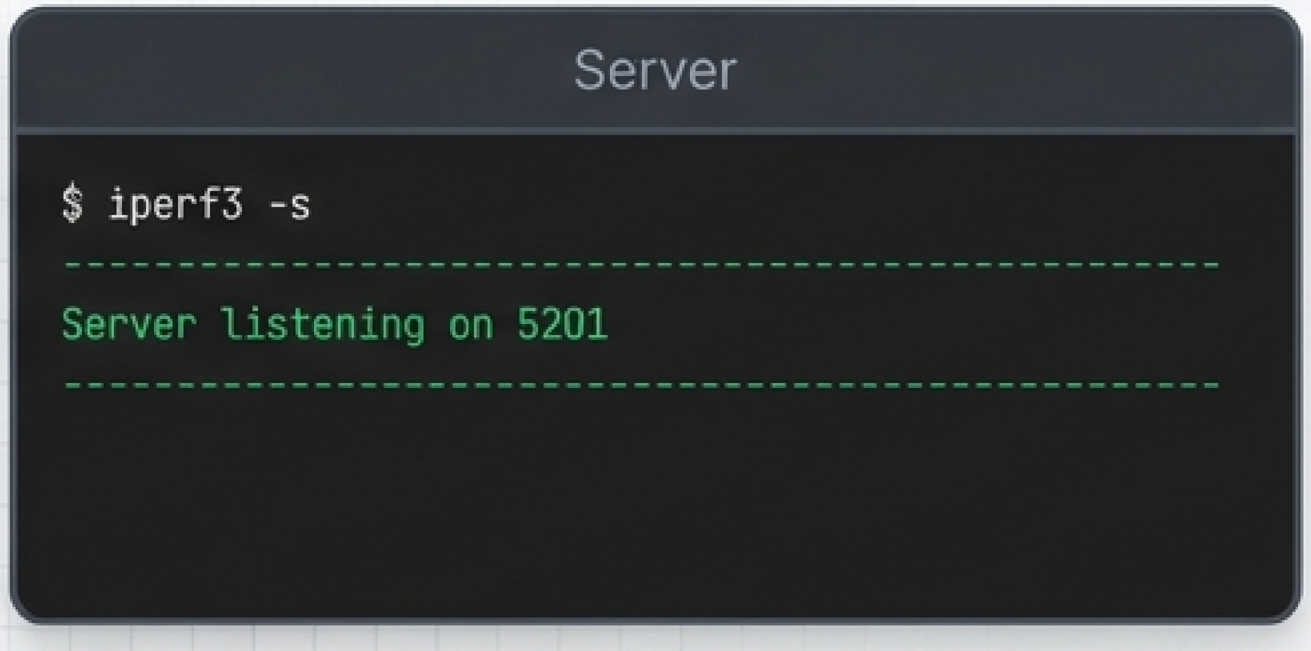
Supports TCP/UDP. Runs on Linux, Windows, macOS.

Pre-Flight Environment Verification

- ☑ **Connectivity:** Valid IP routing between endpoints.
- ☑ **Firewalls:** TCP port 5201 permitted (inbound/outbound).
- ☑ **Stability:** No active maintenance windows or congestion events.
- ☑ **Privileges:** Admin/Root access available for tuning.

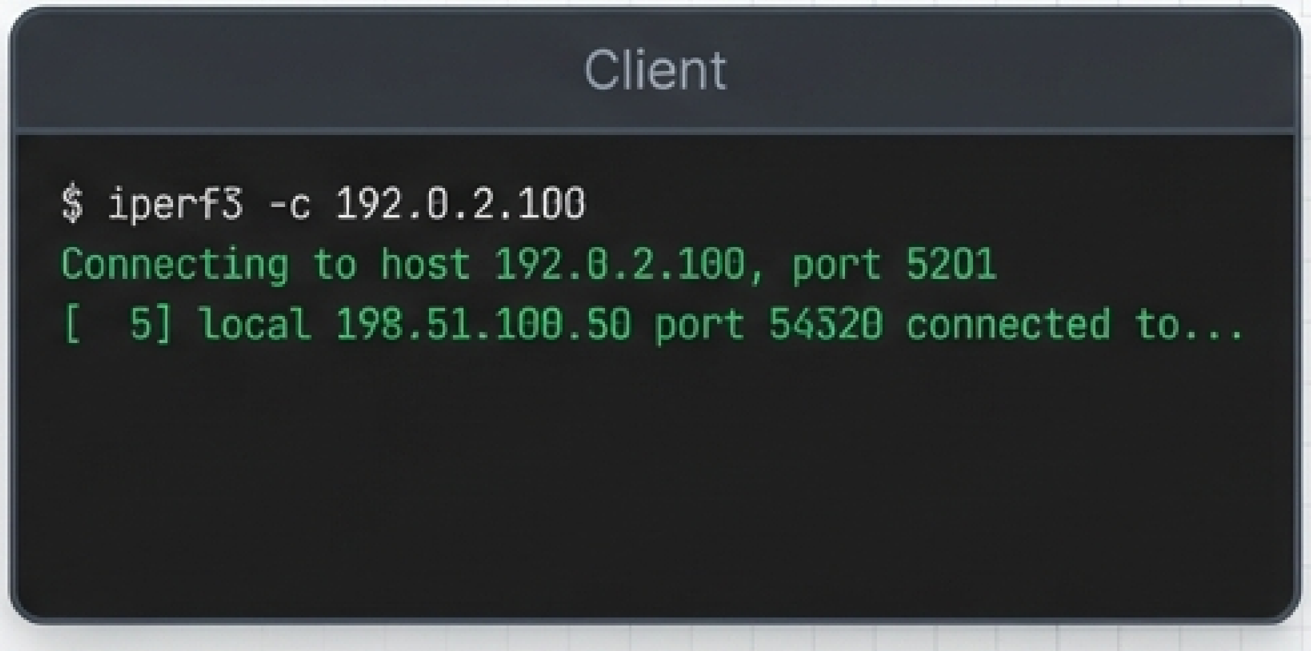
PRO TIP: Ensure `iperf3` versions match on both endpoints to avoid protocol compatibility errors.

Executing the Baseline Test



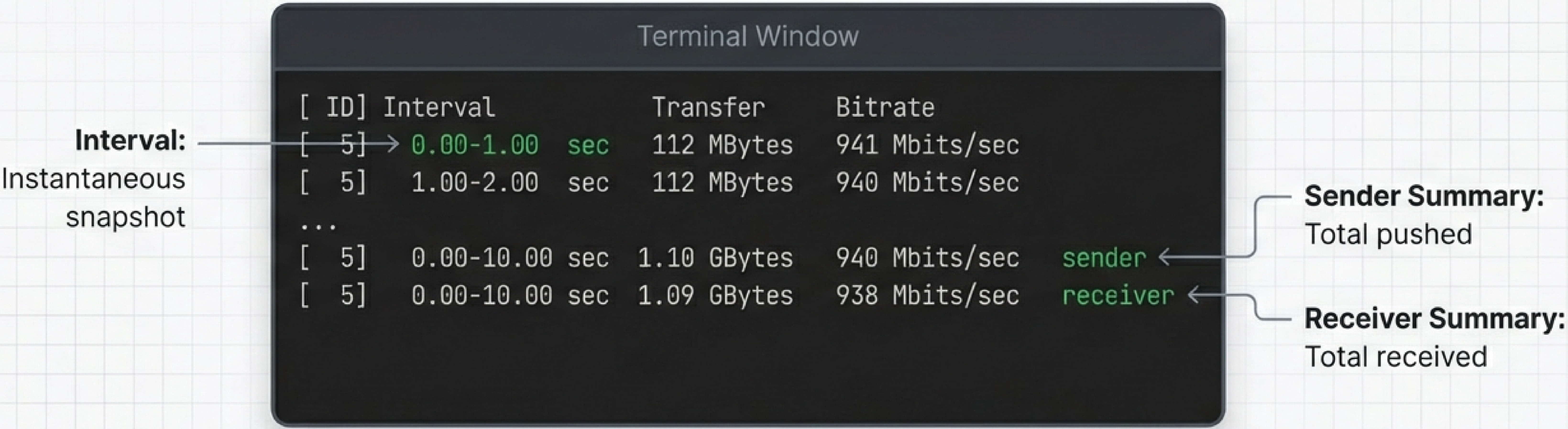
```
$ iperf3 -s
---------------------------------------------------
Server listening on 5201
---------------------------------------------------
```

The server listens; the client initiates.



```
$ iperf3 -c 192.0.2.100
Connecting to host 192.0.2.100, port 5201
[  5] local 198.51.100.50 port 54320 connected to...
```

Deciphering the Output



```
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-1.00   sec   112 MBytes   941 Mbits/sec
[  5]   1.00-2.00   sec   112 MBytes   940 Mbits/sec
...
[  5]   0.00-10.00 sec  1.10 GBytes   940 Mbits/sec   sender
[  5]   0.00-10.00 sec  1.09 GBytes   938 Mbits/sec   receiver
```

Interval: Instantaneous snapshot

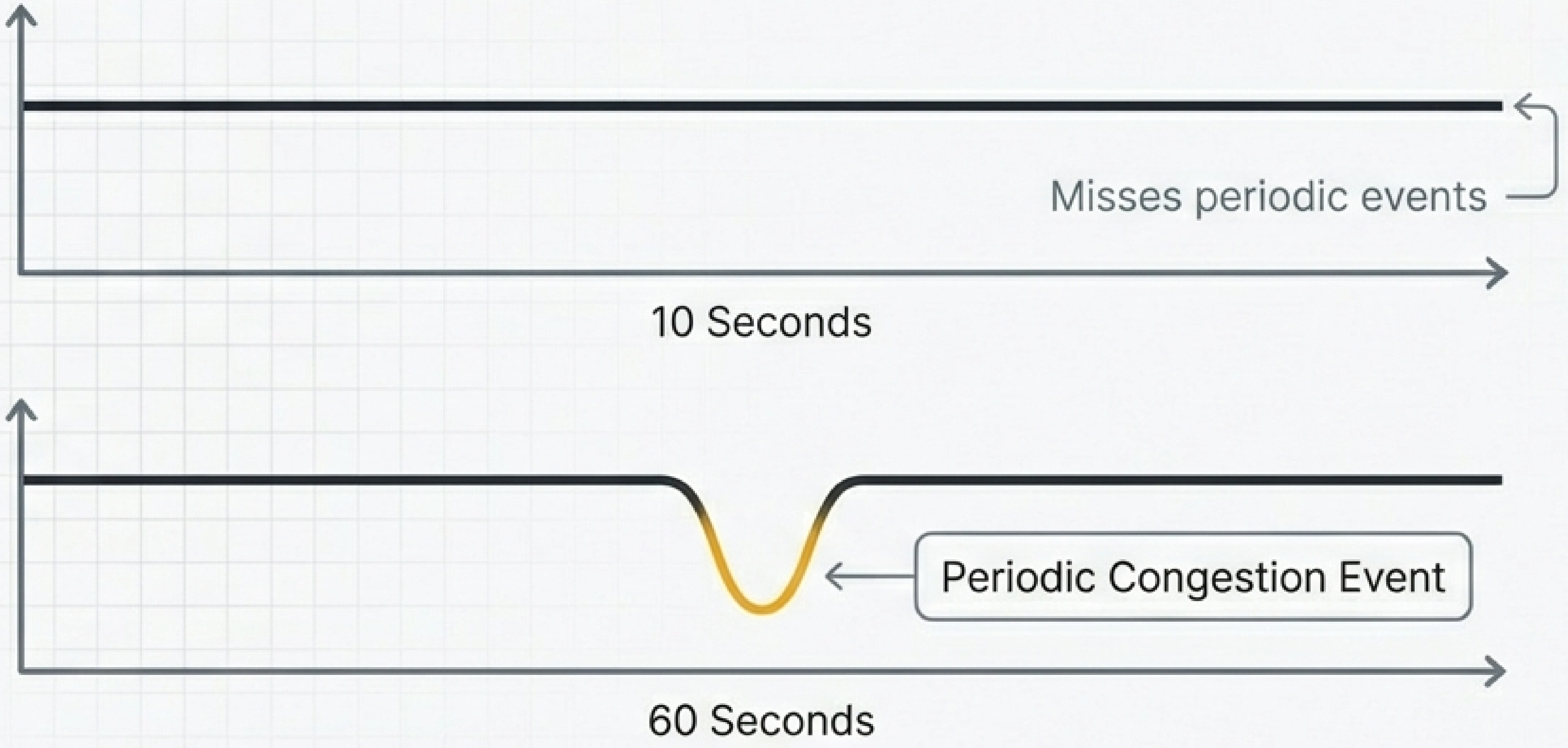
Sender Summary: Total pushed

Receiver Summary: Total received

940 Mbps on a 1 Gbps link is excellent. The gap is standard TCP overhead.

Refinement: Test Duration

```
$ iperf3 -c 192.0.2.100 -t 60
```



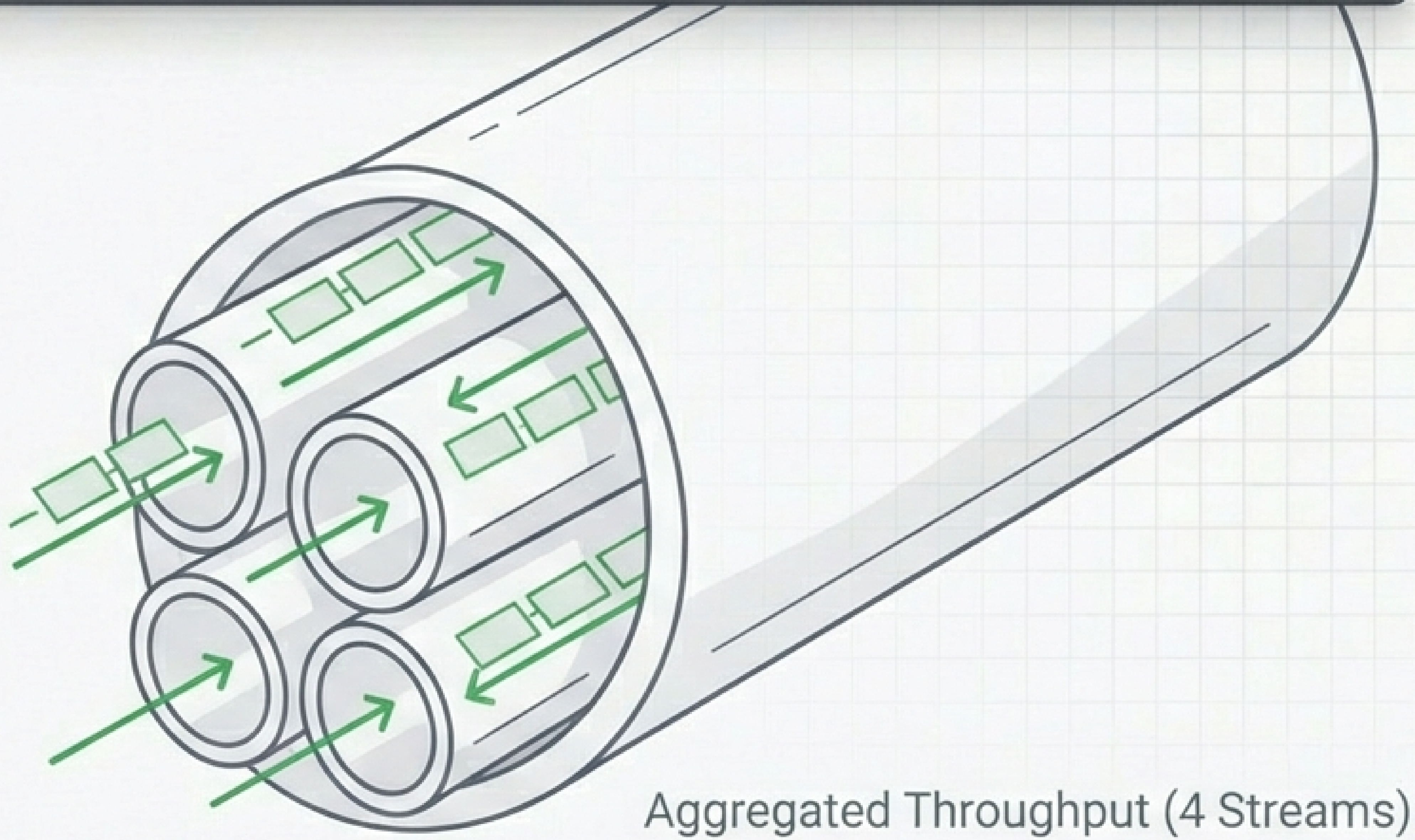
- Smooths out short-term variance.
- Identifies periodic traffic shaping.
- Provides a true picture of sustained throughput.

Refinement: Parallel Streams

```
$ iperf3 -c 192.0.2.100 -P 4
```

Problem: High Bandwidth-Delay Product (BDP) links (WANs) often cannot be saturated by a single TCP stream due to window sizing.

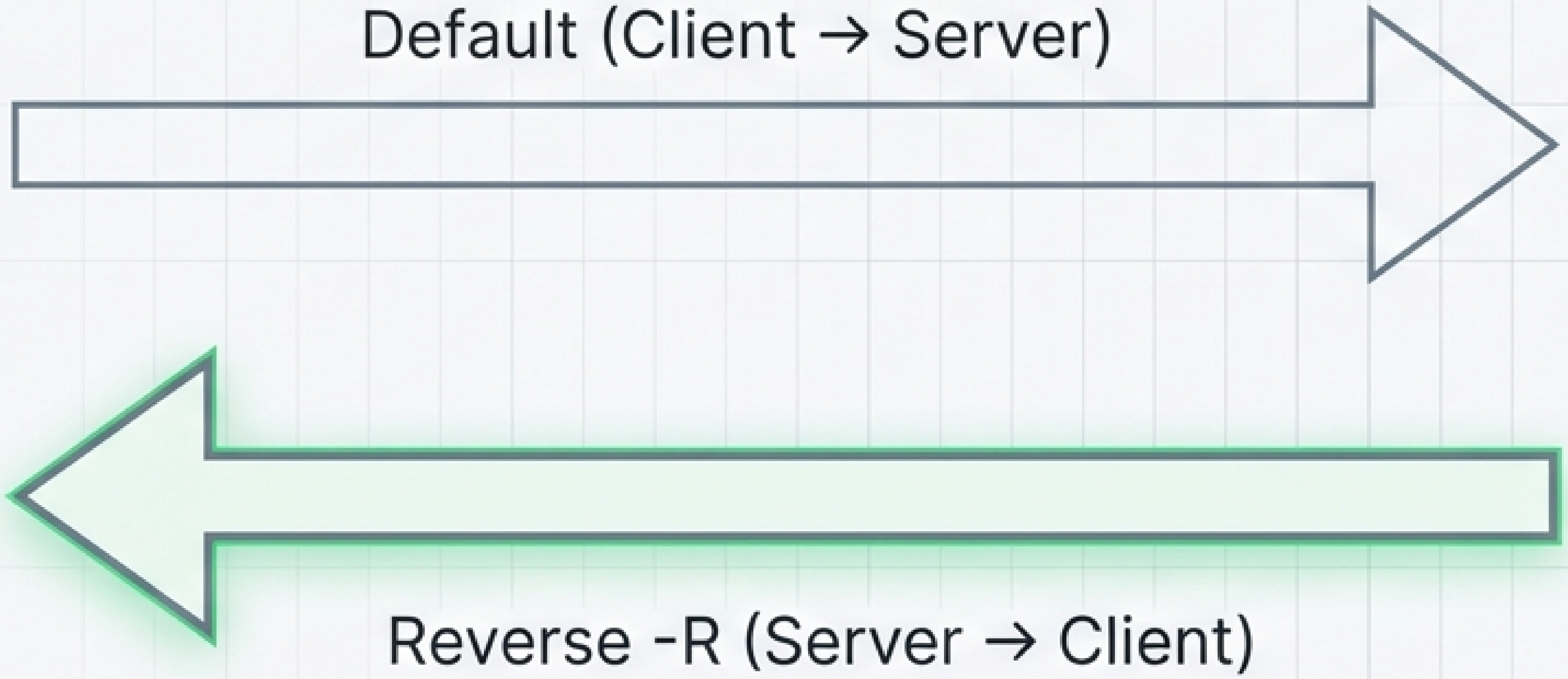
Solution: The `-P` flag runs multiple streams simultaneously to validate total capacity.



Aggregated Throughput (4 Streams)

Refinement: Directionality

```
$ iperf3 -c 192.0.2.100 -R
```

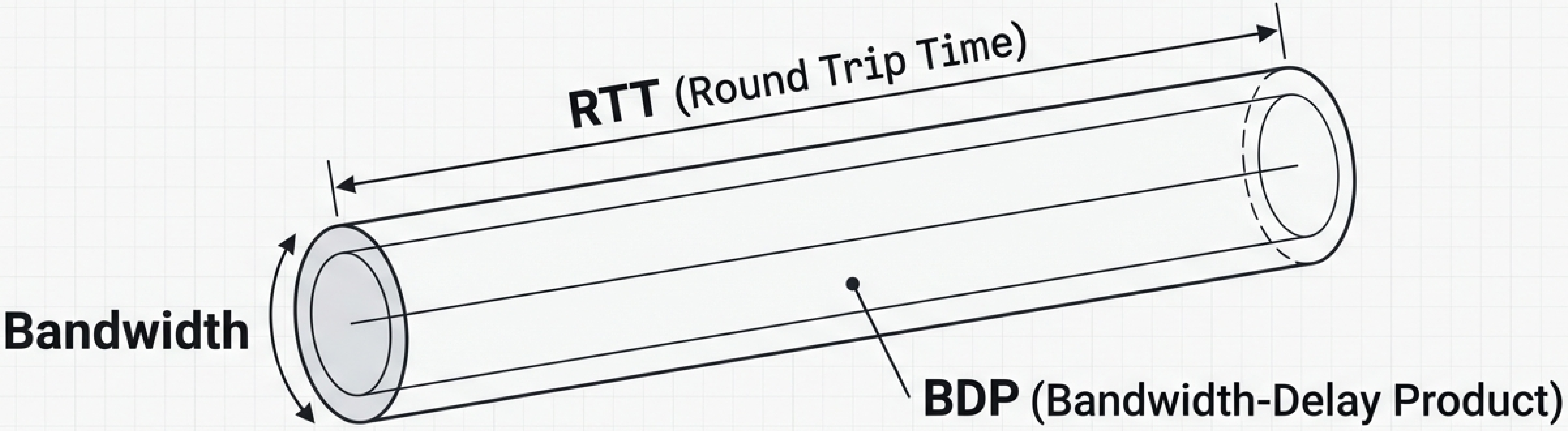


Default (Client → Server)

Reverse -R (Server → Client)

- **The Trap:** Do not assume symmetry. **Fiber and Cable often have different upload/download speeds.**
- **Action:** Always test both directions to catch asymmetric congestion or shaping.

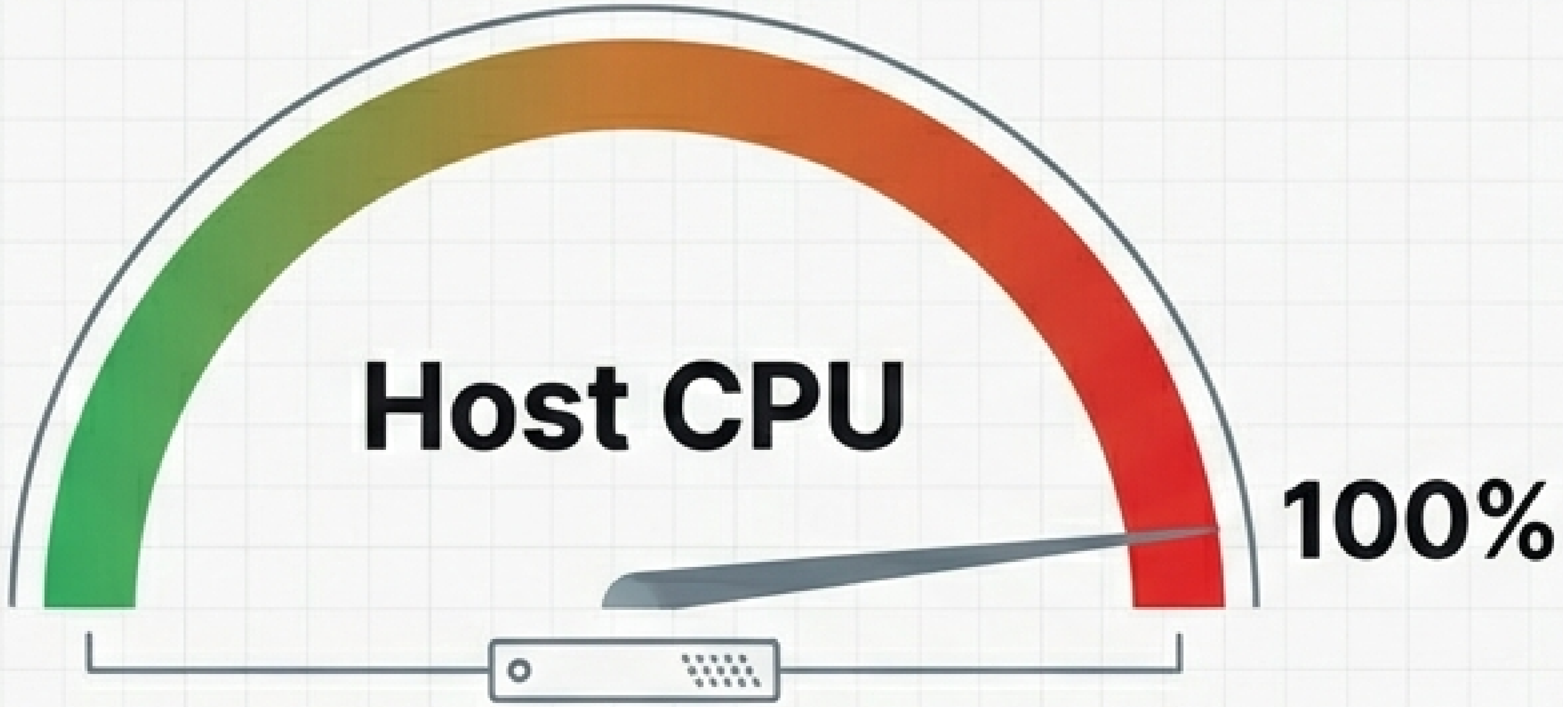
The Physics of TCP: RFC 6349



$$BDP = \text{Link Bandwidth} \times RTT$$

To fill the pipe, you must have enough data ‘in flight’.
If the TCP window < BDP, you cannot saturate the link.

Isolate the Variable: Host Hygiene



> "If the CPU is pegged, you are measuring the server's limit, not the network's."

1. **CPU Saturation:** Critical for 10G+ links.
2. **NIC Offloads:** Enable TSO/GSO/GRO via ethtool.

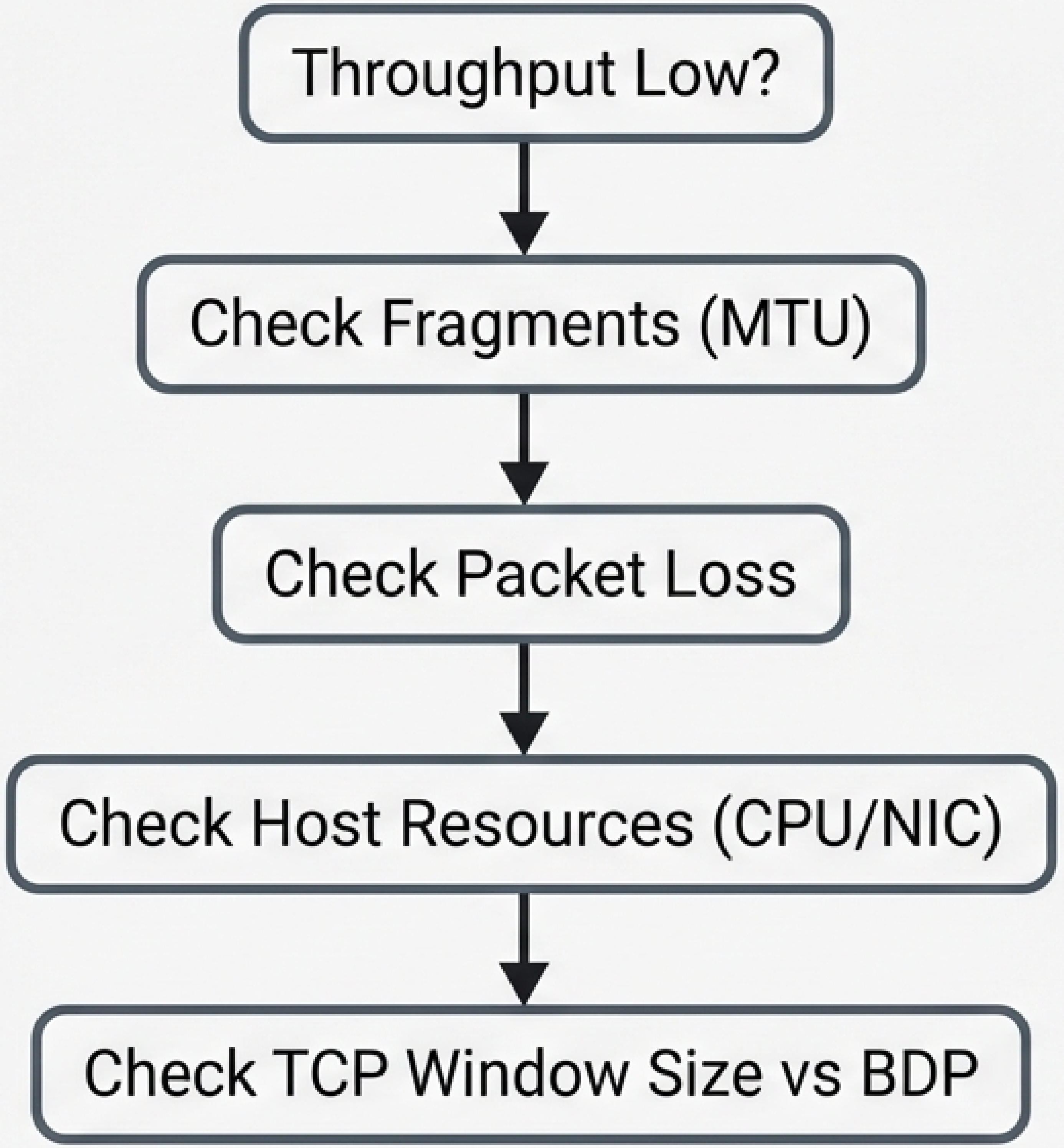
```
$ sudo ethtool -K eth0 tso on gso on gro on
```

3. **Traffic Isolation:** Ensure a clean path with no competing workloads.

Command Reference: Power User Flags

Flag	Function
`-p [n]`	**Port**. Bypass default 5201 if blocked.
`-J`	**JSON.** Output for automation and parsing.
`-O [n]`	**Omit.** Ignore first *n* seconds (skip TCP slow-start).
`-w [n]`	**Window.** Manual TCP window tuning.

Troubleshooting Low Throughput



Throughput Low?

Check Fragments (MTU)

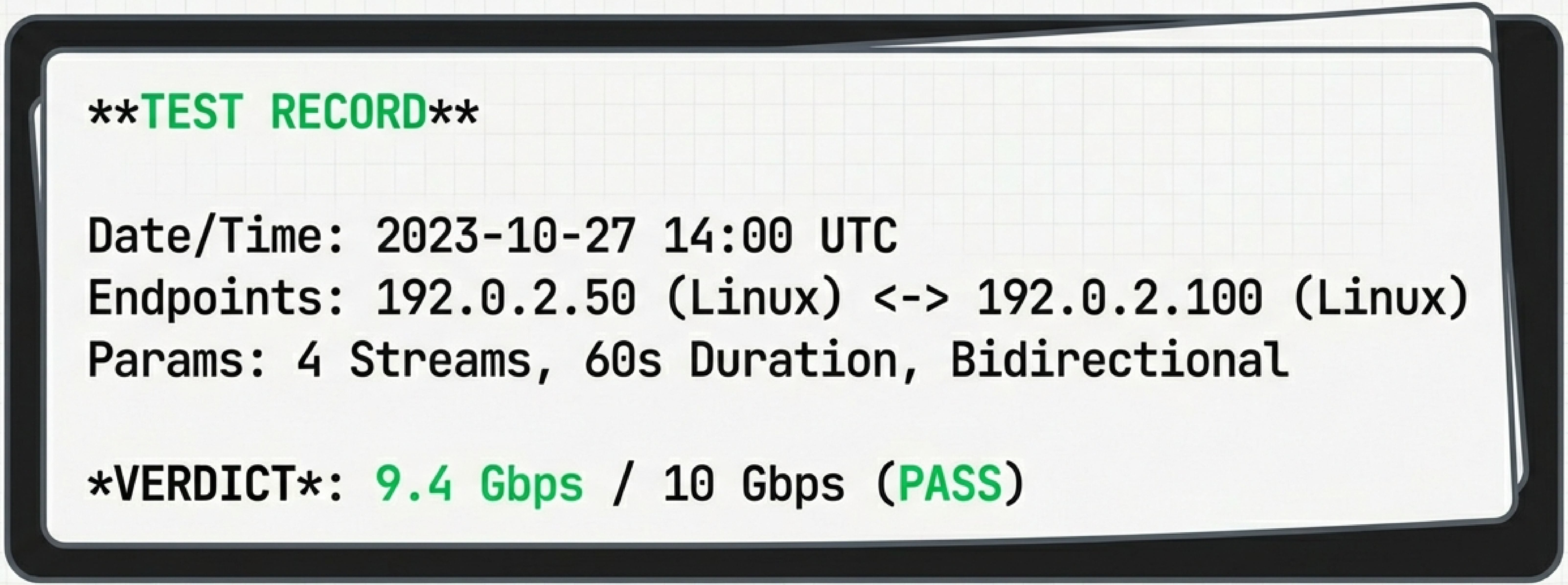
Check Packet Loss

Check Host Resources (CPU/NIC)

Check TCP Window Size vs BDP

**Reality Check: The "Network Probllem" is often an undersized VM or misconfigured NIC.

Verification & Reporting



```
**TEST RECORD**

Date/Time: 2023-10-27 14:00 UTC
Endpoints: 192.0.2.50 (Linux) <-> 192.0.2.100 (Linux)
Params: 4 Streams, 60s Duration, Bidirectional

*VERDICT*: 9.4 Gbps / 10 Gbps (PASS)
```

Build a body of evidence over time. Consistency is key.

Summary & Resources

1. **Test Bi-directionally**: Do not assume symmetry.
2. **Calculate BDP**: Know your physics.
3. **Verify Host Health**: Don't bottleneck on CPU.
4. **Document Results**: Validated vs. SLA.

Resources

- iperf.fr (Documentation)
- **IETF RFC 6349** (Framework)

PRASLAB