TIME'S UP! KEEPING TRANSPORT PROTOCOLS REAL-TIME!

INTRODUCING NANOPING

NANOPING IS A DROP-IN SOLUTION TO PROVIDE REAL-TIME CONNECTIVITY - IN ANY APPLICATION AND SERVICE.

- NanoPing operates at the network packet level
- NanoPing is a plug and play application that allows pain free integration of our dedicated protocol
- NanoPing is available as a software application for immediate evaluation and use
- Dedicated PoC in which can run in your network with your traffic

Crafting real-time solutions on raw UDP, TCP, or QUIC is notoriously complex - and often feels impossible.

NanoPing solves this challenge by blending cutting-edge innovations in protocol design, FEC, networking, scheduling, and monitoring into a seamless, powerful solution.

With NanoPing, your team can shift focus from the headaches of connectivity to unlocking the full potential of your applications / services.

REAL-TIME COMMUNICATION IS BECOMING THE NORM

IN VIRTUAL REALITY,

Latency of over 58ms results in motion sickness

IN MULTIPLAYER GAMING,

Latency of over 100ms hurts gaming experience significantly

IN TELEDRIVING & TELE-OPERATION,

Latency of over 200ms results in collisions

IN AI / RPC,

Keep latency under 250 MS for timely remote monitoring intervention|

BUILDING REAL-TIME COMMUNICATION IS HARD

YOUR EXISTING STACK

Bespoke networking solutions are built on existing networking solutions (e.g. single link, TCP, etc), and implementations need to consider the existing infrastructure

YOUR USER'S UNRELIABLE NETWORK(S)

Mobile and wireless networks are a de-facto norm, but the underlying protocols have not caught up

IMPROVE POOR USER EXPERIENCE

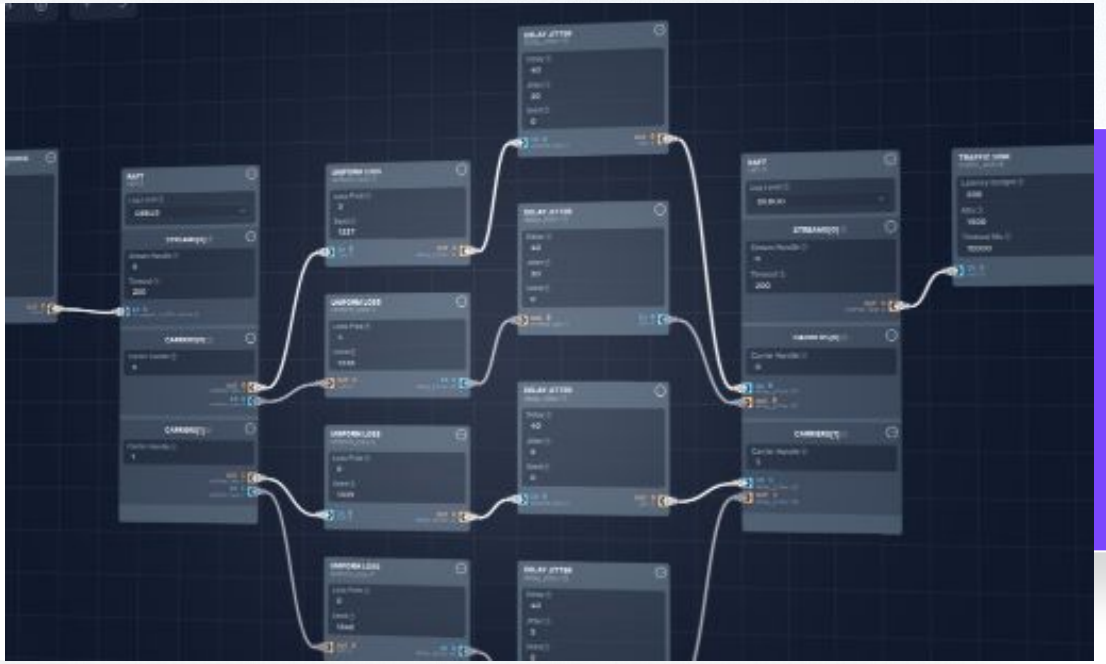
Applications usually do not have control over the network of the user, and trying to make the network itself reliable is not a viable option

MAKING UNRELIABLE NETWORKS RELIABLE

For real-time applications, the requirements are simply greater. It's your job to need to accommodate any constraint (platform, network, use case, etc...) or you're at fault

BUILDING REAL-TIME COMMUNICATION IS HARD

```
user@nanoping sudo np up
Started. Serving dashboard on http://127.0.0.1:8081
```



THE FUNDAMENTALS

RLNC

Erasure coding ensures efficient recovery and utilization of multi-path

MONITORING

Extract and monitor important KPIs to ensure value delivered and to spot issues

INTEGRATION

Binary or software integration in end-user applications / services

LATENCY AWARE TRANSPORT PROTOCOL

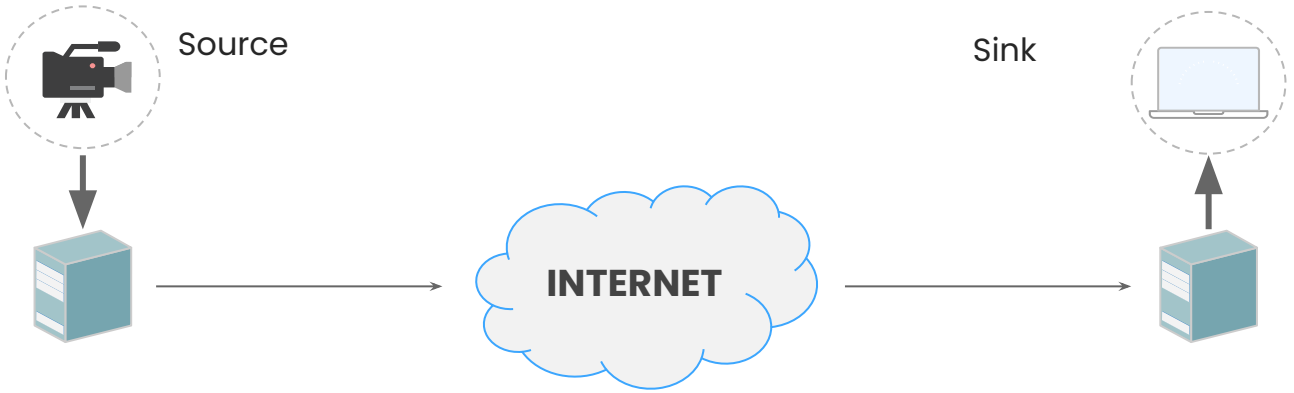
Tracking latency and reliability across multiple network paths (if available) - dynamically adjusting strategy to ensure reliable delivery

CUSTOMIZABILITY

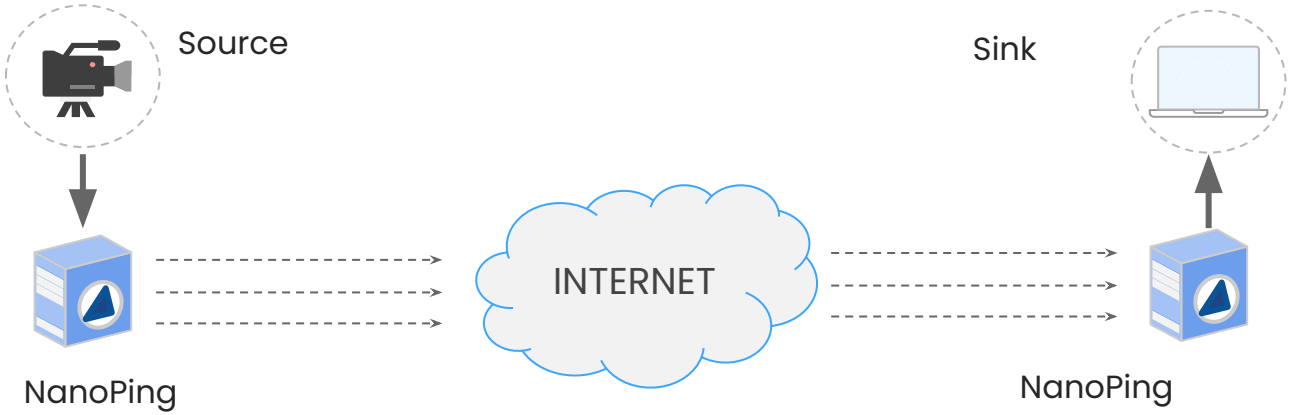
Adapt to different application requirements

NANOPING DATA-PLANE

TODAY



NANOPING



→ **TCP/UDP TRANSPORT**

- Not latency aware
- Not multi-path
- Only retransmissions
- No observability
- No stream prioritization
- No application feedback

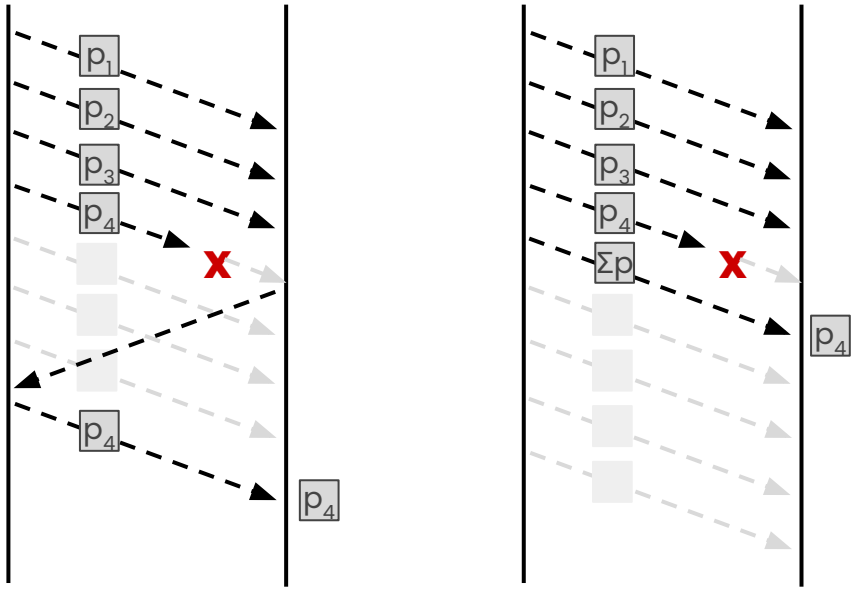
→ **TCP/UDP TRANSPORT**

⇢ **NANOPING TRANSPORT**

+ Latency aware
+ Multipath
+ Adaptive packet FEC/ARQ
+ Built-in observability
+ Stream priorities
+ Application feedback

TECH DEEP DIVE

One of the key reasons for using **ECC/FEC** is to **minimize per-packet delay** for latency sensitive applications.

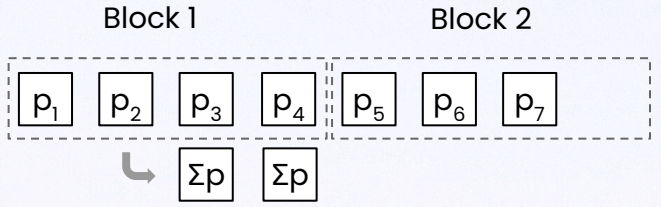


ARQ RECOVERY LATENCY: 1 RTT per retransmission

ECC/FEC RECOVERY LATENCY: Distance to the repair packet

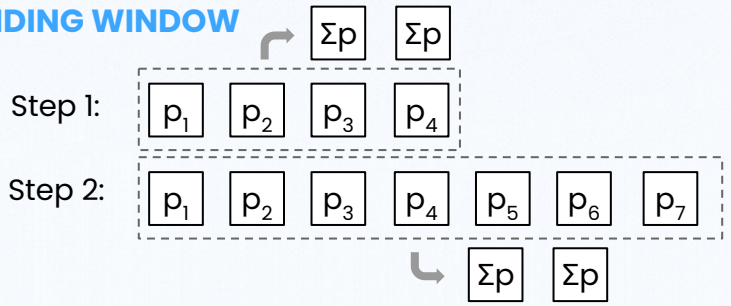
- By interleaving repair packets ECC/FEC based schemes can provide repair for packet loss faster than retransmissions.
- Picking the right algorithm is crucial! Fundamentally two types of ECC/FEC schemes exist:
  1. Block based (e.g. Reed Solomon)
  2. Sliding window based

BLOCK BASED



- Repair only possible at end of block
- Repair only covers packets in one block
- Time to repair dictated by block size

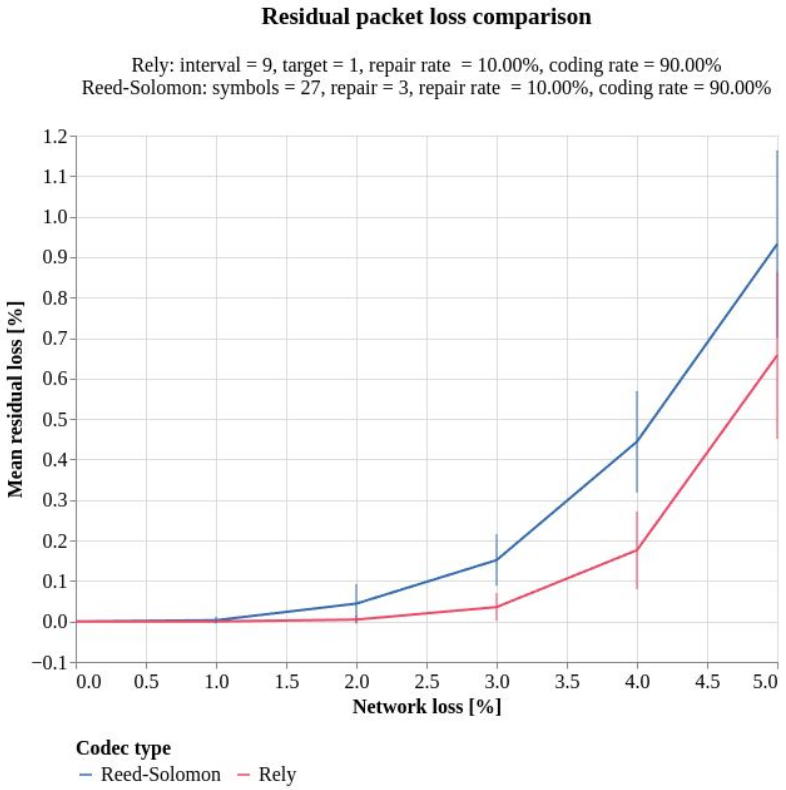
SLIDING WINDOW



- Repair can be generated at any time to ensure proper protection
- Repair can cover any packet in window

COMPARISON

Sliding window coding provides better or equal performance for (1) packet loss protection (2) per packet delay:



IMPLEMENTATION BENEFITS

- Using block ECC/FEC all repair and computations is performed at the end of the block - causing computational and bandwidth spikes
- Using sliding window repair can be evenly distributed providing a smooth computations and bandwidth usage
- Using block ECC/FEC requires management of multiple encoders and decoders per stream
- Sliding window codes naturally fit real-time streaming / packet-processing use-cases - with a single encoder/decoder per stream
- Using block ECC/FEC low-throughput applications require small blocks (reducing packet loss protection)
- Sliding window codes are elastic and seamlessly adapts to the traffic pattern (from low to high throughput)
- Using block ECC/FEC low-throughput applications require small blocks (reducing packet loss protection)
- Sliding window codes require no fragmentation of input data


nanoping