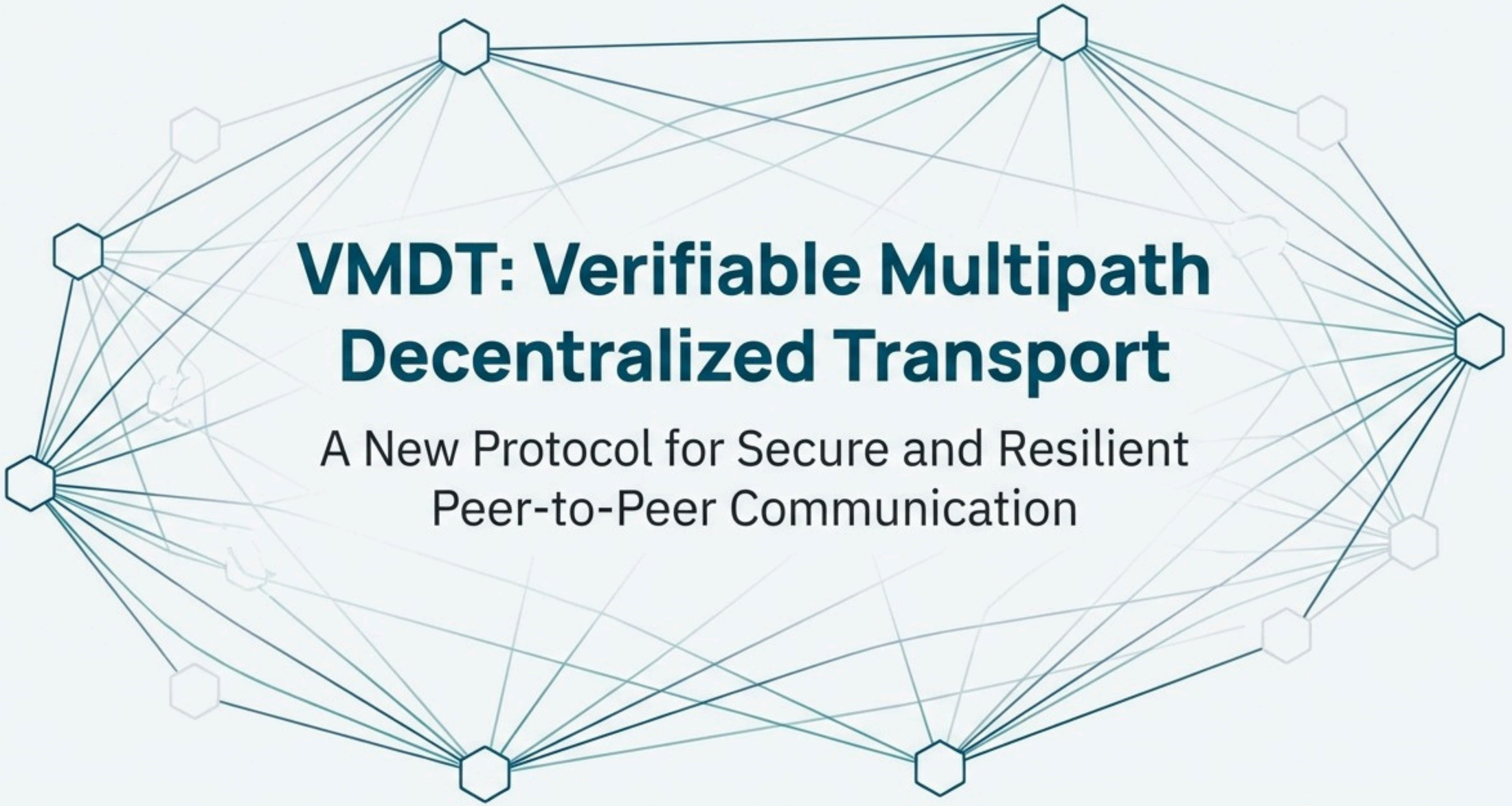
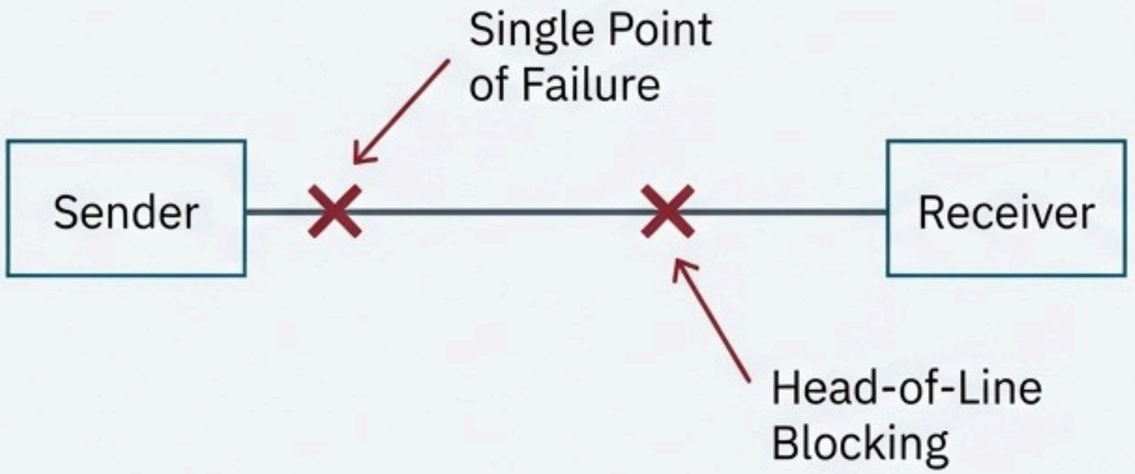


VMDT: Verifiable Multipath Decentralized Transport

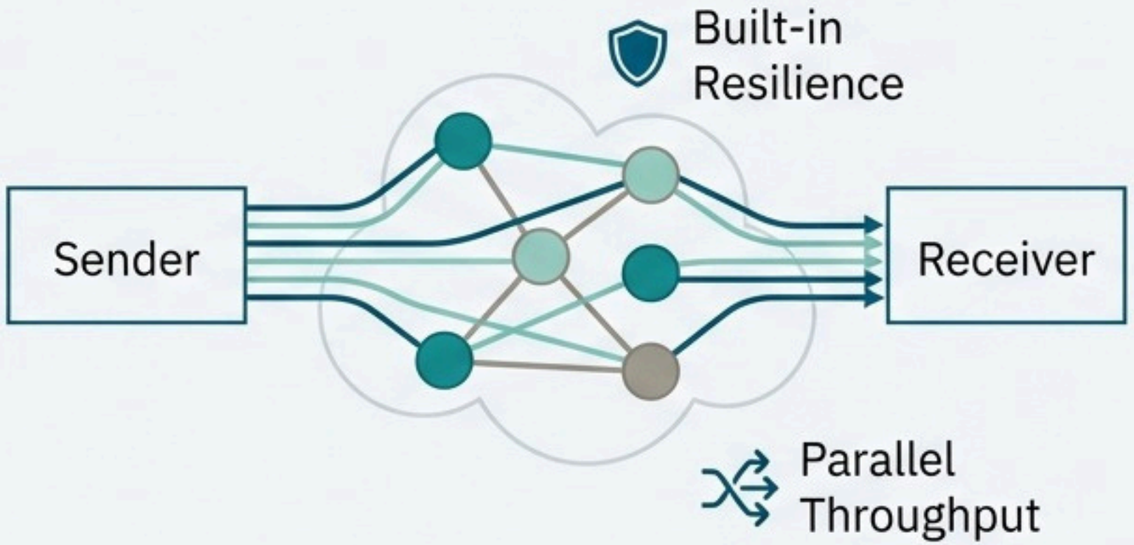
A New Protocol for Secure and Resilient Peer-to-Peer Communication

The Limits of Single-Path Communication

Traditional Protocols like TCP



VMDT's Multipath Approach



Traditional protocols are vulnerable to path failures and bottlenecks, a critical issue for modern decentralized systems.

The VMDT Solution: A Three-Pillar Approach

Verifiable

Uses cryptographic secret sharing to ensure data integrity and authenticity.

Multipath

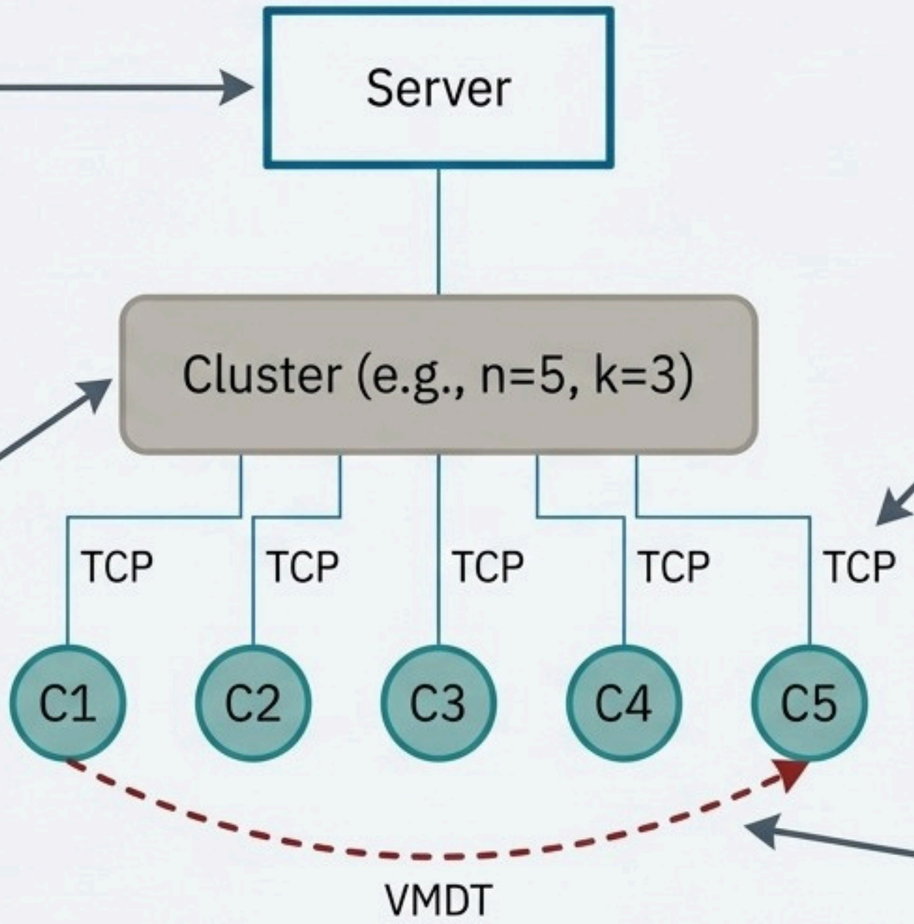
Distributes data across multiple network paths simultaneously for reliability and speed.

Decentralized

Operates in a peer-to-peer cluster with no single point of failure.

How VMDT Organizes Communication

A central coordinator for cluster management and peer discovery. Communication with peers is over TCP.

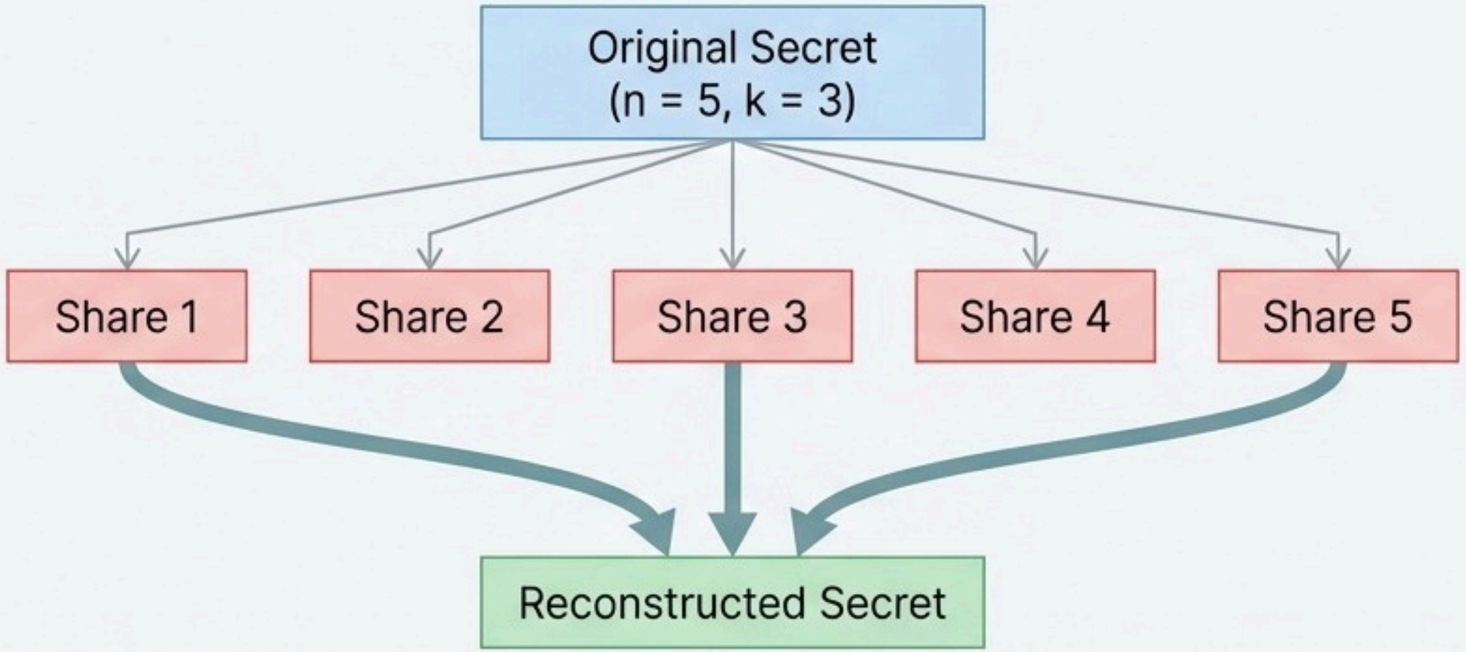


A logical group of peers that collaborate to transmit data.

Clients (Peers): The endpoints that send and receive messages.

Data Flow: Data, split into shares, travels peer-to-peer over multiple paths using UDP.

Security and Resilience Through Secret Sharing



Security

Fewer than k shares reveal nothing about the secret. (e.g., k-1 shares are useless).

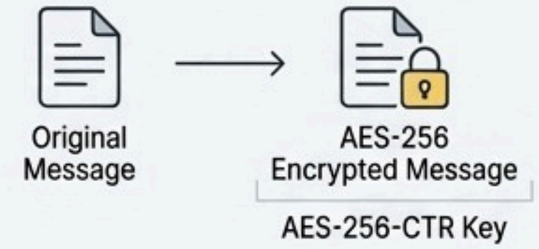
Loss Resistance

The system can tolerate the loss of up to n-k shares. (e.g., 5-3 = 2 lost shares).

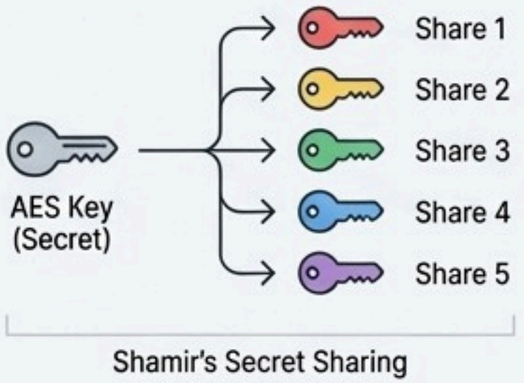
The Engine Behind VMDT: How SSMS Works

Phase 1: Splitting the Secret

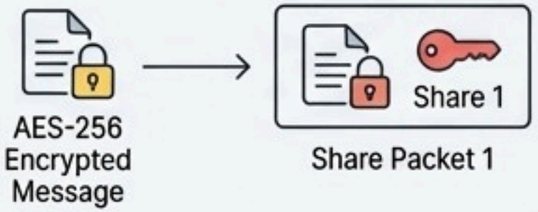
The original message is encrypted using a temporary AES-256-CTR key.



This AES key (the "secret") is split into n key shares using Shamir's Secret Sharing scheme.

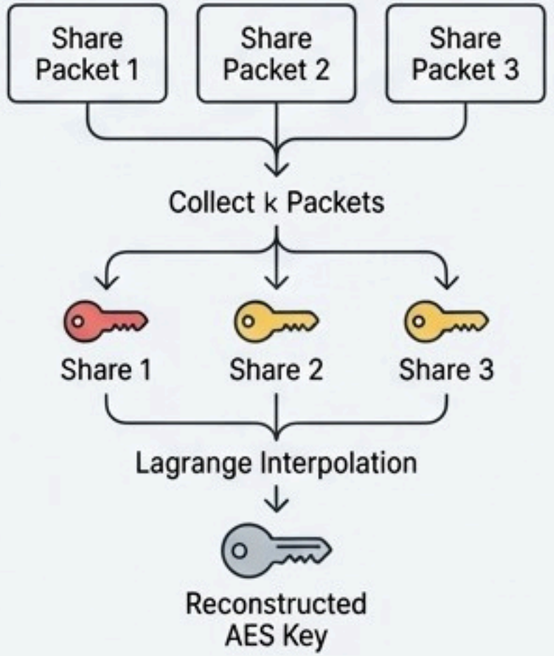


The final data packet for each share contains the encrypted message plus one unique key share.



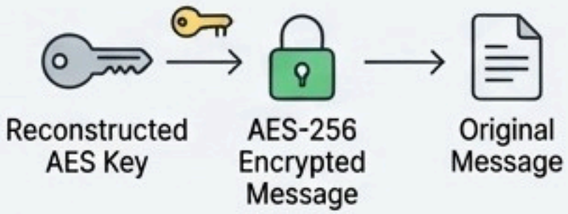
Phase 2: Reconstructing the Secret

The receiver collects at least k of these data packets.



The k unique key shares are used to rebuild the original AES key using Lagrange interpolation.

The reconstructed AES key is used to decrypt the message.



Security by Design: No Single Point of Compromise

Confidentiality

An intermediate peer only holds encrypted ciphertext and a single, meaningless piece of the decryption key. They cannot read the original message.

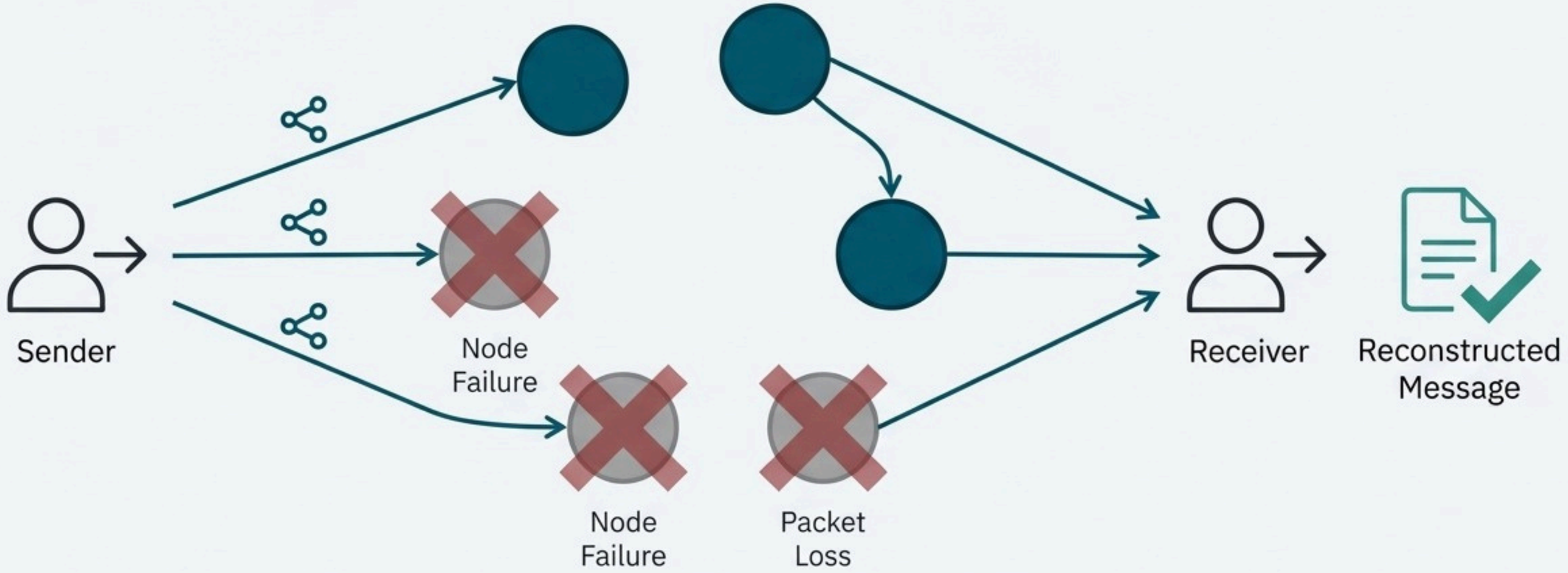
Data Integrity

The cryptographic nature of Shamir’s scheme makes unauthorized modification of shares mathematically detectable during the reconstruction phase.

Privacy

Because the full message is never transmitted over a single path, the system is inherently resistant to eavesdropping.

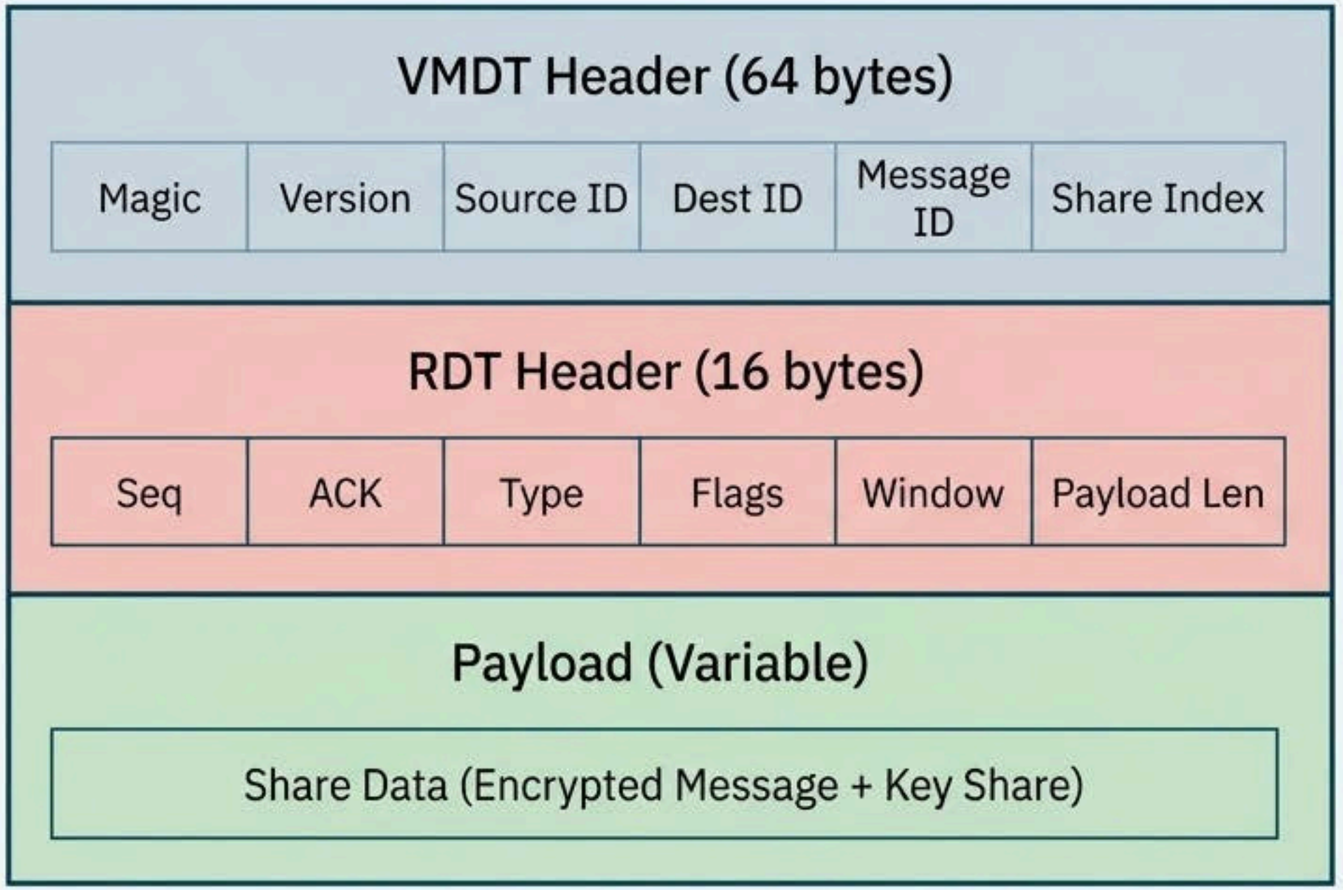
Loss Resistance: Built-in Fault Tolerance



VMDT can withstand up to $n-k$ node or path failures without data loss.
As long as the k threshold is met, the message is successfully delivered.

Anatomy of a VMDT Data Packet

The data packet is structured with dedicated headers for routing, context, and reliability.



VMDT Header (64 bytes)

Contains identifiers for the message, source, destination, and share index to establish context for the share.

RDT Header (16 bytes)

Manages reliable delivery over UDP. Contains fields like Sequence Number, ACK Number, and Flags.

Payload (Variable)

Contains the actual share data generated by the SSMS process (encrypted message + key share).

A Dual-Layer Protocol Design

Feature	Control Layer	Data Layer
Purpose	Cluster Management, Peer Discovery, Coordination	High-Speed Share Transmission
Transport	TCP	UDP
Rationale	Guarantees reliable, in-order delivery of critical commands.	Low overhead and flexibility for a custom Reliable Data Transfer (RDT) implementation.
Format	Human-readable ASCII messages	Binary headers and payload

Orchestrating the Peers with Control Messages

Cluster Management

`CREATE_CLUSTER|<client_id>|<n>|<k>`

`JOIN_CLUSTER|<client_id>|<cluster_id>`

Peer Communication

`GET_CLIENT_INFO|<client_id>|<target_client_id>`

`REGISTER_P2P_PORT|<client_id>|<port>`

Share Management

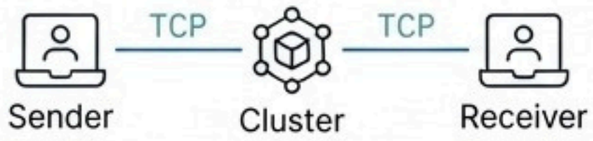
`SHARE_POOL_ADD|<msg_id>|...`

`SHARE_POOL_CLAIM|<msg_id>|...`

> A simple, text-based command system running over TCP coordinates all peer activity before, during, and after data transfer.

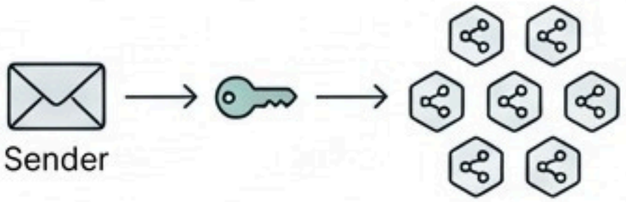
The Lifecycle of a Message in VMDT

1. Setup



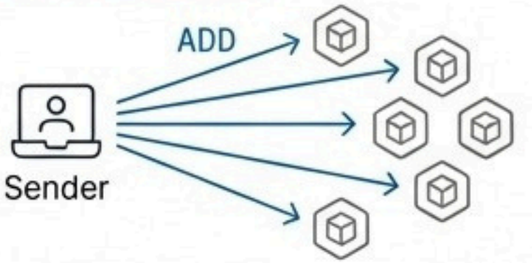
Sender and Receiver join a cluster via TCP control messages.

2. Preparation



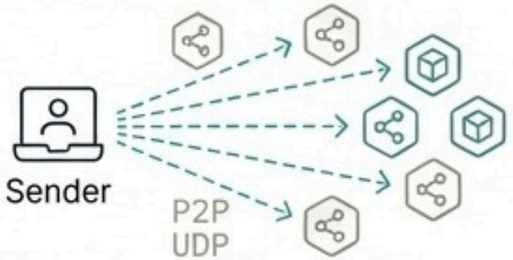
The Sender's client encrypts the message and splits the key into *n* shares using SSMS.

3. Distribution



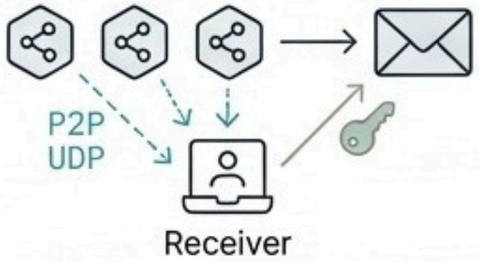
Sender broadcasts a `SHARE_POOL_ADD` control message to all cluster peers.

4. Claim & Transfer



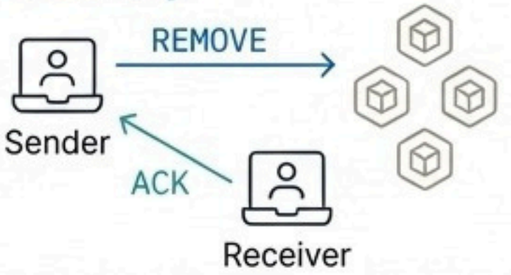
The Receiver broadcasts `SHARE_POOL_CLAIM` messages for *k* Peers transfer the claimed shares

5. Reconstruction



The Receiver's client uses the *k* shares to rebuild the AES key and decrypt the message. to the Receiver via P2P UDP.

6. Cleanup



The Receiver sends an acknowledgment, and the Sender broadcasts a `SHARE_POOL_REMOVE` message.

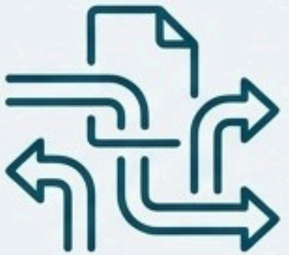
How VMDT Compares to Existing Protocols

Feature	TCP	QUIC	VMDT
Pathing	Single Path	Single Path (with migration)	**True Multipath**
Fault Tolerance	Low (path failure is total)	Moderate (fast reconnect)	**High (k out of n resilience)**
Architecture	Centralized (Client-Server)	Centralized (Client-Server)	**Decentralized (Peer-to-Peer)**
Privacy	None by default (requires TLS)	Encrypted by default	**Privacy by Architectural Design**

Where VMDT Excels: Practical Use Cases

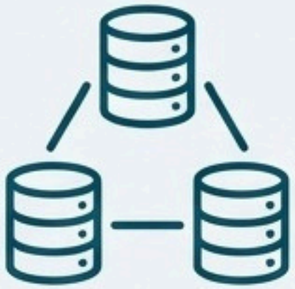
Secure Messaging

For confidential communication in untrusted or hostile network environments.



Resilient File Transfer

Ensuring critical file delivery despite unstable network connections or node failures.



Distributed Storage Systems

To store data redundantly and securely across a decentralized network of nodes.

Privacy-Preserving Applications

For any application where protecting the content and metadata of communications is paramount.

Acknowledging the Trade-Offs and Limitations

Increased Overhead

Splitting data into n shares increases the total amount of data transmitted across the network.

No Built-in Authentication

The current version lacks robust client authentication or authorization mechanisms.

Protocol Complexity

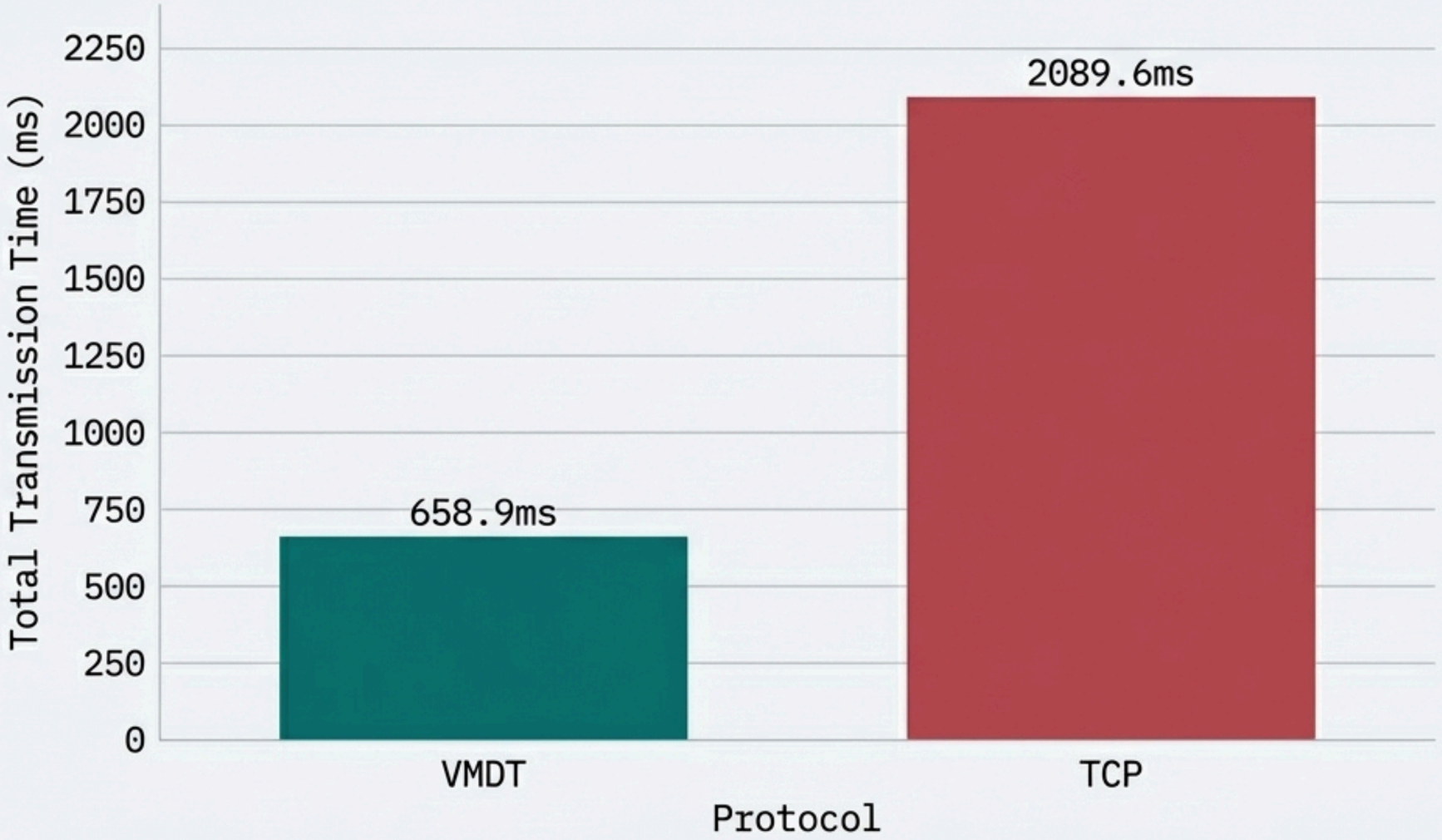
Inherently more complex to manage than a standard point-to-point TCP connection.

NAT Traversal Challenges

As a P2P protocol, it can face connectivity issues with devices behind Network Address Translation (NAT) without external solutions.

VMDT Drastically Reduces Transmission Time Under Packet Loss

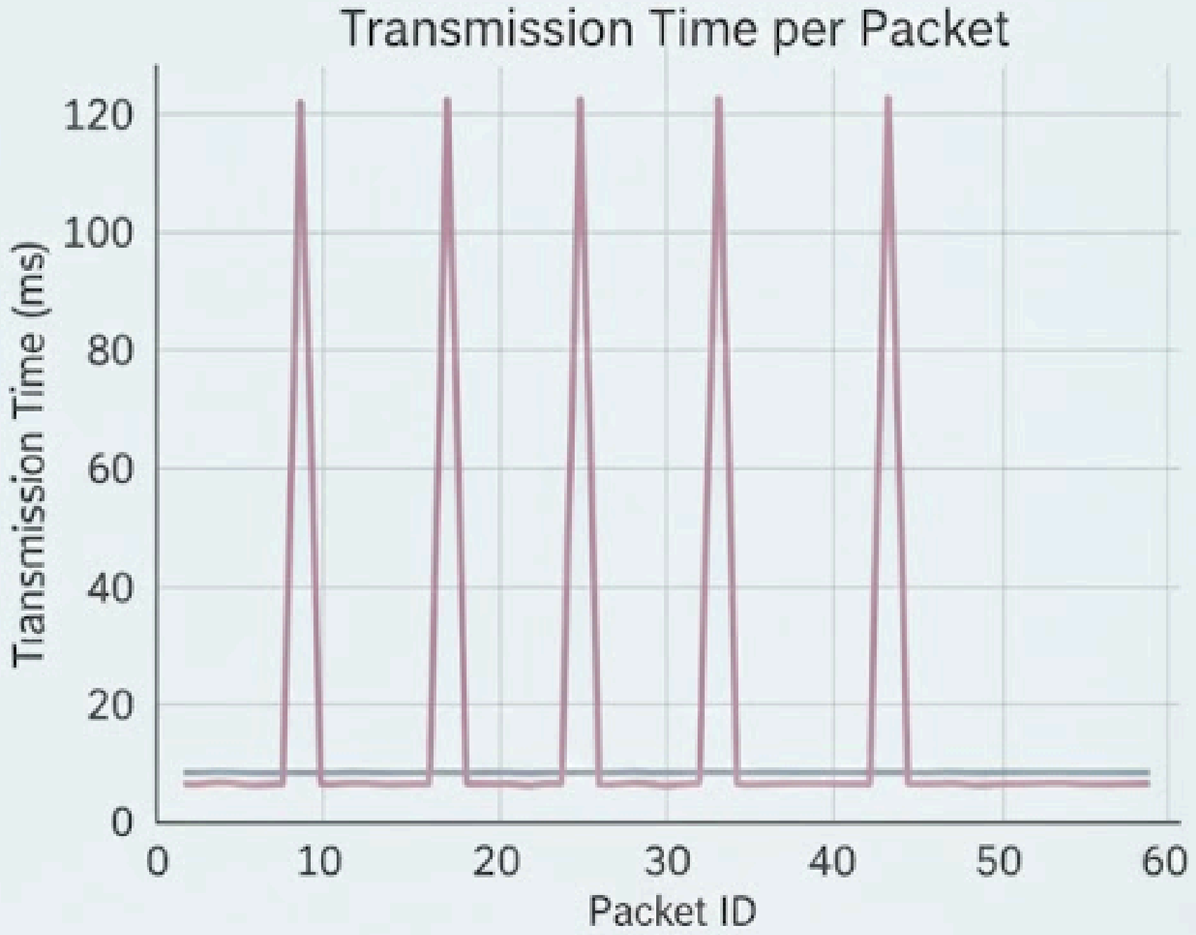
Comparison based on a transfer of 59 packets with 6.0% packet loss.

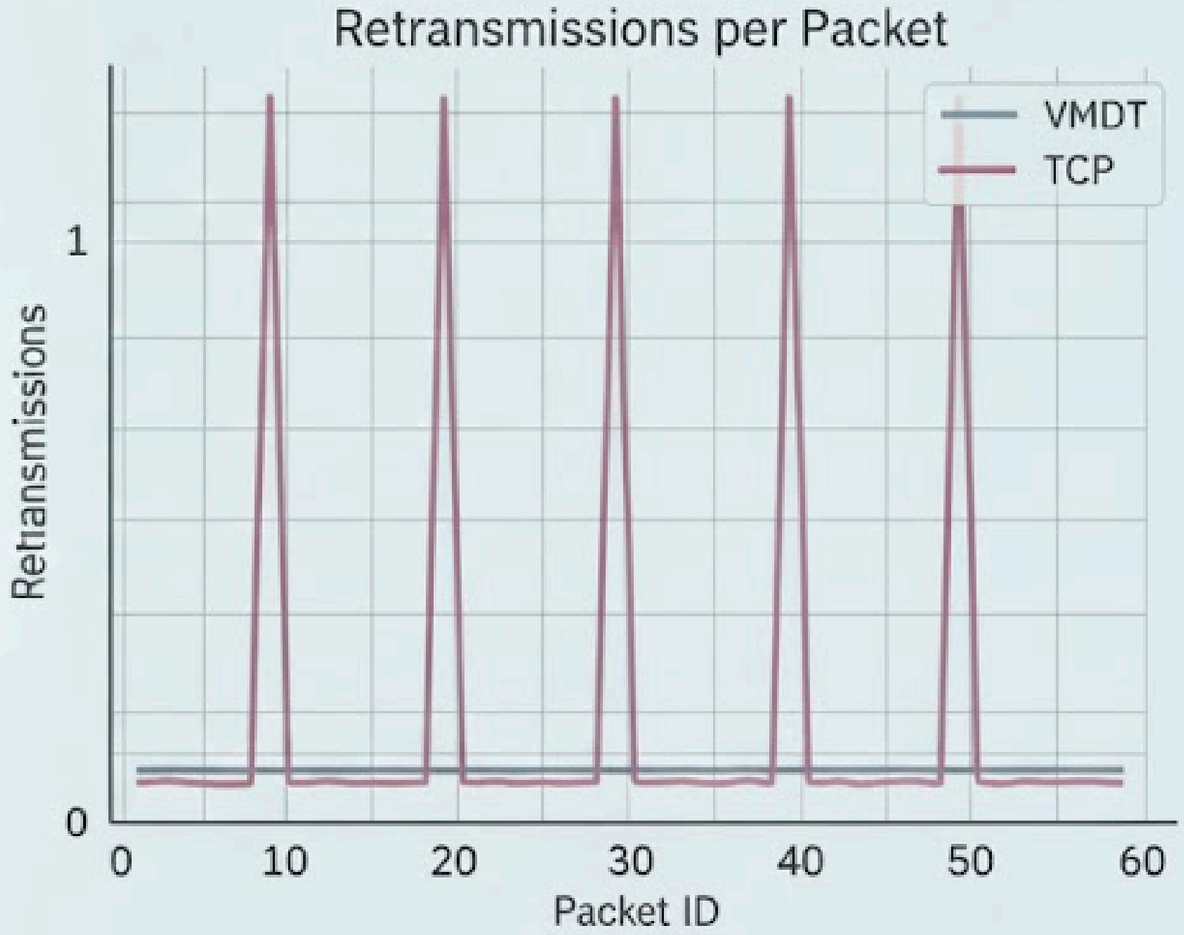


VMDT is over 3x faster than TCP in this lossy environment.

Packet-Level Analysis: How VMDT Maintains Stability

(n=6, k=3, 59 packets, 6.0% loss)





TCP stalls during packet loss, leading to high-latency retransmissions. VMDT's k-out-of-n design **simply ignores** lost packets, ensuring a smooth and predictable data flow without interruption.

The Road Ahead for VMDT

Implementing end-to-end encryption keys for enhanced security.

Adding formal client authentication and authorization layers.

Integrating standard NAT traversal solutions like STUN/TURN.

Developing a performance and monitoring dashboard for easier management.

Adding support for mobile clients.

VMDT: Secure, Resilient, and Decentralized Communication

VMDT provides a robust solution for P2P communication by uniquely combining **Cryptographic Secret Sharing** and **True Multipath Routing**.

Its decentralized architecture delivers exceptional **Security** against eavesdropping and **Fault Tolerance** against network failures by design.

It is a powerful alternative to traditional protocols for modern applications where **Resilience** and **Privacy** are non-negotiable requirements.