JETSTREAM

Cumul8:Jetstream Technical Overview

Introduction

Cumul8:Jetstream (Jetstream) is a datagram-based file transfer system. Its network protocol, the Jetstream Transfer Protocol, is specifically designed to operate over unreliable networks such as the commodity internet and is robust to packet loss, packet reordering, and packet delays. The Jetstream Transfer Protocol supports unencrypted transfers, or transfers that are additionally secured and encrypted by the Jetstream Security Layer (JSL).

UDP-based Protocol

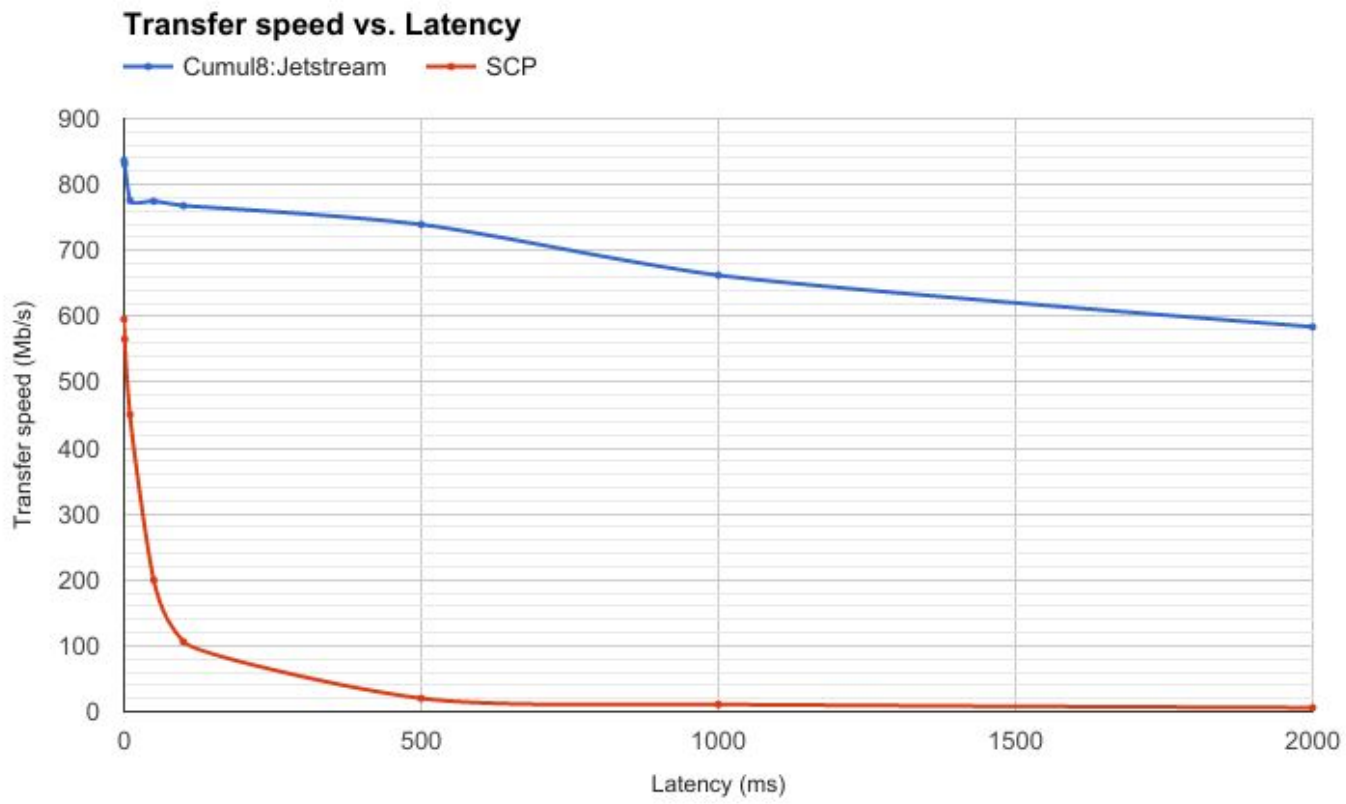
By relaxing the need for strict, reliable, in-order delivery at the network packet level, Jetstream is tolerant of high-latency connections that would severely limit TCP-based methods. Jetstream achieves this by using a *deferred, aggregate acknowledgement algorithm*. The source files to be transferred are broken up into fixed-size *blocks*. The sender Jetstream node reads blocks from the source storage, encrypts, and transmits them to the receiver node. The receiver collects blocks and writes to the target storage. Periodically, the receiver sends an acknowledgement report back to the sender containing the list of blocks that it successfully received. Jetstream coalesces consecutive block indices into *block ranges* in the acknowledgement.

The sender periodically saves the list of acknowledged blocks to storage. This enables Jetstream to quickly restart an interrupted transfer from the last save point.

The sender will continue transmitting blocks at the current *target rate* for a period of time even in the absence of acknowledgement from the receiver. The decoupling of transmission and acknowledgement, together with the ability to acknowledge variable amounts of successfully transmitted data using a small amount of bandwidth allows Jetstream to maintain a high transmission speed over a high-latency, potentially unreliable network.

Comparison to TCP-based Protocols

As latency increases in a network, TCP-based transfer protocols quickly become limited by the round-trip time of the connection.



100x20 M byte (2 G) files transmitted over a 1 Gbit network between Windows Server 2012 and CentOS 7 hosts connected through a dedicated netem (https://wiki.linuxfoundation.org/networking/netem) routing node to introduce latency. Consult https://wondernetwork.com/pings for estimates of ping times between cities across the Internet.

Even under ideal conditions, Jetstream's low-overhead UDP engine outperforms SCP. As latency is introduced, TCP-based transfers experience severe performance drop off. By using a decoupled, low-bandwidth back channel for acknowledgements, the Jetstream protocol is relatively latency tolerant up to ping times of 500 ms, and performs well beyond that. Real-world latencies over the Internet range from about 100 ms for coast-to-coast in North America, 150+ ms for North America to Europe, and 200+ ms for North America to Asia.

Rate Control

Motivation

Being able to accurately manage the rate at which packets are transmitted is critical for two reasons.

One, because the sender does not wait for acknowledgement of successful transmission from the receiver, it is possible for a sender to flood a slower receiver with blocks faster than the receiver can commit them to storage. This could be caused by a bottleneck somewhere along the network, unreliability, or bandwidth limitations of the receiver's storage. Overwhelming a receiver results in dropped packets and the need to retransmit, lowering efficiency.

Two, if allowed to run uncapped, a Jetstream transfer can saturate shared storage and network resources, degrading the performance of other clients.

In Practice

The Jetstream rate controller is designed to achieve the maximum practical transmission rate with a minimal amount of dropped packets and retransmissions.

The rate control design is about *end-to-end performance*, not just network bandwidth. It adjusts the rate based on the time to do a full transaction: read from storage, encrypt, send over the network, traverse all firewalls and routers, receive from the network, decrypt, and write to storage. If any of these stages becomes a bottleneck, the rate will adjust accordingly. Other systems just measure how quickly bytes can go over the network without considering file IO or other factors that affect the actual throughput.

The Jetstream rate controller configuration specifies minimum and maximum send rates for each transfer. The transfer engine attempts to achieve, but will not exceed the maximum send rate. As acknowledgements arrive from the receiving side, the rate controller builds a statistical model of the current network conditions. This model dynamically adjusts the send rate to find the sweet spot of optimal transmission speed with minimal retransmissions.

Optimized for High-performance Storage

To achieve the goal of end-to-end performance, Jetstream tailors its IO system around the capabilities of high performance NAS, SAN, and direct-mounted SSD and RAID storage systems. The lightweight and multi-threaded reading and writing systems achieve quick transfer startup and sufficient IOPS to maintain the target transmission rate.

Jetstream's IO system supports packing multiple files into a single transfer block, with no loss of efficiency when transferring small files. It is not necessary to "tar up" groups of files to improve performance.

There are no built-in limits in terms of maximum file size or number of files in a transfer.

Server Design

Jetstream uses a persistent server for both sending and receiving data. By queuing transfers, the server optimizes the send rate across requests, avoiding “bubbles” between the end of one transfer and the start of the next one. Jetstream maintains rate control parameters for each destination, not individual transfers, allowing future transfers to start immediately at an optimal rate.

Access Control

Jetstream manages access to its servers by integrating with an organization's existing user authentication services. Jetstream can authenticate against local user accounts or an LDAP/AD infrastructure. Under Mac and Linux, Jetstream uses Pluggable Authentication Modules (PAM). Under Windows, it uses the system authentication service. Jetstream restricts IO operations to those granted to the authorized user.

Security and Data Integrity

While Jetstream supports unencrypted transfers, the server defaults to full encryption over the control (API) and UDP transfer ports. Jetstream uses modern industry-standard session negotiation and payload encryption algorithms.

By default, Jetstream establishes secure connections to its servers via TLSv1.2. For wider compatibility, the server can optionally be configured to also accept connections via the older TLSv1.0-1.1 standards. The client may optionally verify the server's API public key to check its identity. Once the connection has been established, the client sends authorization credentials to the server to gain access to the Jetstream API. Jetstream forwards these credentials to the machine hosting the server for validation.

Before starting a transfer, the client registers the receiver's server as a destination. The sender may optionally verify the public key of the receiver to verify its identity. The receiver may optionally filter which senders may connect to it by whitelisting sender public keys.

The client then initiates a transfer with credentials used to authorize access to the receiver server.

Jetstream UDP traffic is encrypted using 128-bit AES-GCM (Galois/Counter Mode) with a secret nonce (initialization vector) for each datagram and a tag length of 96 bits. GCM provides confidentiality of the payload data and includes integrity assurance over the entire datagram.

Jetstream will use hardware-accelerated AES-NI if supported by the host CPU. Jetstream automatically selects progressive fallbacks for processors that don't support AES-NI.

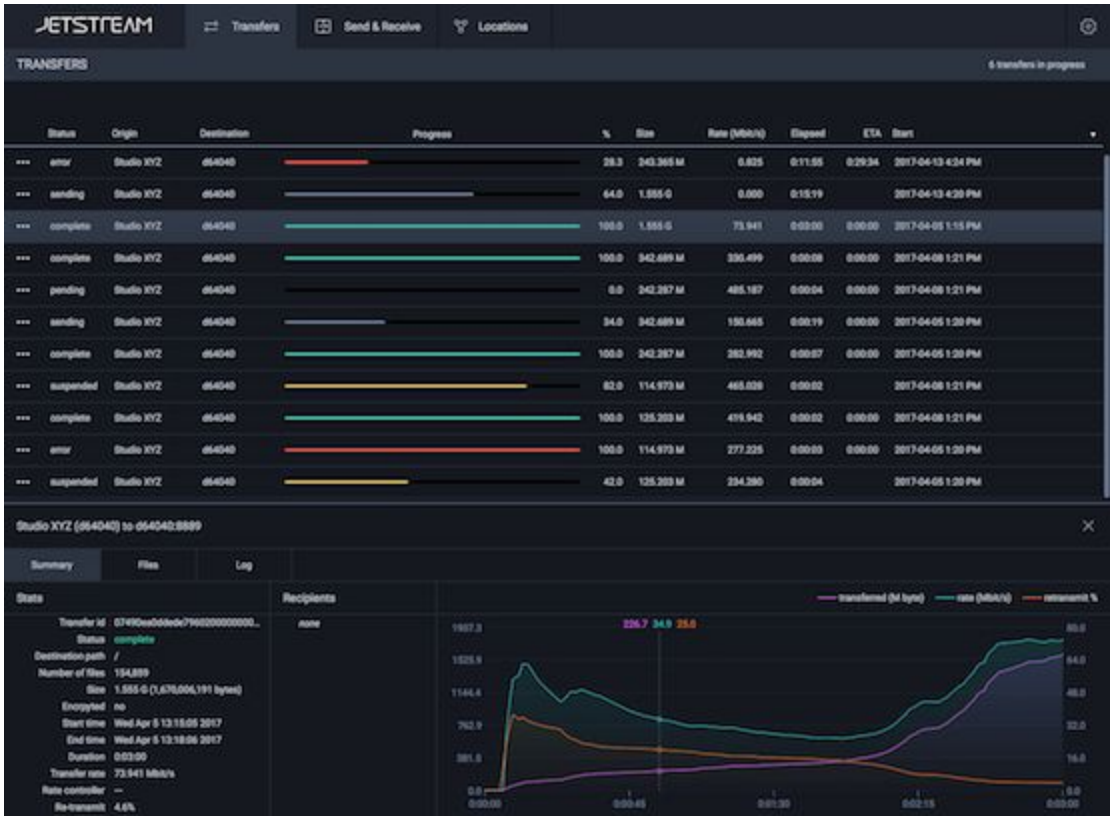
For added security, Jetstream optionally provides an encrypted, streaming proxy service to safely transmit confidential assets from a private network without direct internet access. This eliminates the need for dedicated staging areas connected to the Internet, which expose assets and introduce extra data movement between storage pools.

Comprehensive Logging and Analysis Tools

Each Jetstream node maintains an *event log* which records all activity of that node in a sequence of flat files containing key-value pairs. Jetstream records detailed statistics of ongoing transfers, including network performance, at one-second intervals. This data can be either read directly from the file system where it is stored, or retrieved over the network via the Jetstream API. For long-term data retention, exploration, analysis, and troubleshooting, the Jetstream Command and Control Server (C&C) application periodically collects event log entries from registered nodes and stores them in a relational database. The information can then be explored with the Jetstream C&C web client application. The Jetstream native GUI client also provides simple graphical tools for exploring the event log for transfers.

Designed for Integration

All Jetstream tools are built with the public *Jetstream API* that is also available for customers to integrate high-performance data transfer into their own applications and workflows.



The Jetstream graphical client is built with the Jetstream C++ API bindings.

Jetstream follows a client-server architecture. Clients typically connect to a Jetstream server over a secure TCP connection via one of the client APIs or client applications. The wire protocol consists of JSON requests, which Jetstream optionally compresses for efficiency. The Jetstream distribution contains API bindings for C++ and Python. API commands can also be issued from the command line or in a shell script.

Client applications include jscp, an SCP-like secure file copy utility, and the Jetstream graphical client, a native interactive front end application.

Summary

Cumul8:Jetstream is built from the ground up to be a performant and secure data movement solution that any organization can easily integrate into its workflow to serve its particular data transfer needs.