FORTRA

GUIDE *(FileCatalyst)*

Accelerating File Transfers

Table of Contents

As file sizes continue to grow, managing and delivering large files is becoming an important consideration for organizations of all sizes. Companies have abandoned the commonly used FTP/TCP protocol as a delivery method in favor of alternative file transfer solutions which provide acceleration, reliability, management, and security.

Companies seeking file transfer acceleration are not limited to the high-tech sector. Organizations leveraging the benefits of acceleration are found in sectors such as media and entertainment, natural resources, supercomputing, legal, health, government, financial, manufacturing, and more. Companies using TCP-based file transfer protocols to transfer large data sets may experience slow file transfers, or even failed and/or corrupt file transfers. This failure rate can be detrimental to organizations moving large data sets on a regular basis. This wastes valuable time, especially if these transfers take hours across an otherwise healthy network.

This white paper will address some of the issues organizations encounter when using TCP-based protocols. It will also outline some other common file sharing methods, and the issues inherent with each. It will then outline how the FileCatalyst solutions, and how they overcome the issues surrounding slow file transfers.

Finally, this paper will present a number of scenarios that showcase the advantages of switching to an accelerated file delivery system, such as FileCatalyst, that includes reliability, security, automation, and tracking.

Introduction to FileCatalyst

FileCatalyst is a software platform designed to accelerate and manage file transfers securely and reliably. FileCatalyst is immune to the effects of latency and packet loss impacting traditional file transfer methods like FTP, HTTP, or CIFS. Global organizations are now using FileCatalyst to address file transfer needs, including content distribution, file sharing, and offsite backups.

Understanding TCP

The Transmission Control Protocol (TCP), in conjunction with the Internet Protocol (IP), is the basic framework and set of rules that define the internet and how data is sent and received. TCP is also the framework used by all the common internet protocols, including FTP, SFTP, HTTP, SCP, CIFS, and SMTP.

TCP is a connection-oriented protocol; meaning that it establishes a connection between applications at each end. TCP sends and receives packets across a network between each endpoint.

TCP can break application data into packets that are easier to manage and send across a network. The packets are then numbered and sent in groups. The biggest advantages of TCP are stateful connections, guaranteed packet arrival, and built-in network congestion control.

Although TCP benefits from these advantages (and all the commonly-used internet protocols associated with it), there are some glaring disadvantages—especially with bulk data transfers over IP links where latency and/or packet loss are present.

The Pitfalls of TCP When Transferring Large Data

Latency and How It Affects TCP

To reliably transfer data across a network via TCP, the receiving party must send an acknowledgment (ACK) to the sending party confirming the packet was received. These ACKs must be sent in sequential order, and the sender cannot send another packet of data until it receives an acknowledgment that the previous packet was received. The time spent sending a packet and receiving the ACK is measured as Round-Trip Time (RTT). This is one of the reasons TCP can be slow: time is spent waiting for ACKs instead of transmitting data.

On local networks with computers sending and receiving data in close proximity to each other, ACKs spend less time in flight and do not slow down the data transmission. However, as the geographic distance increases, so does the RTT. The slower ACK reception causes an exponential throughput degradation for bulk data transfers.

TCP responds to this by adjusting the acceptable amount of unacknowledged data allowed on the link. If the acceptable amount is surpassed, the transfer will stop and wait for an ACK. The optimal amount of unacknowledged data en route should equal the end-to-end bandwidth, multiplied by the RTT. This sum is known as the bandwidth-delay product.

TCP perpetually estimates this value and sets a "TCP window." When the bandwidth-delay product exceeds the TCP window, the result is "dead air," which creates even more wait time. Some satellite connections must deal with hundreds, or even thousands, of milliseconds of RTT.

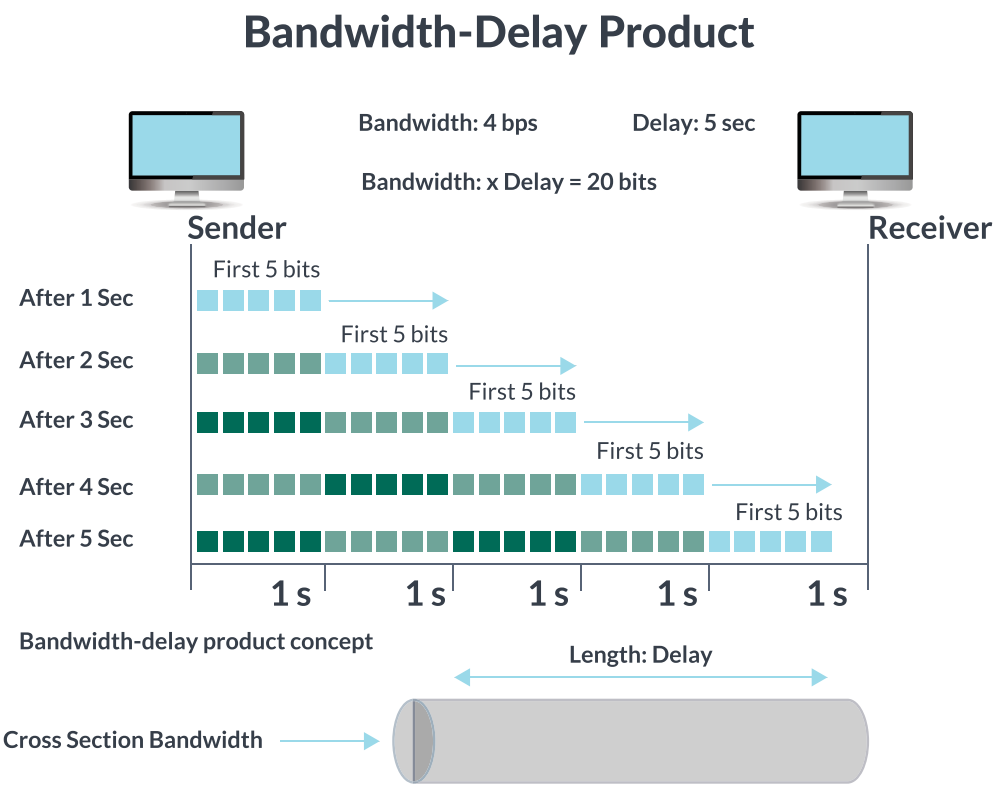

Image 1: Bandwidth Delay Product

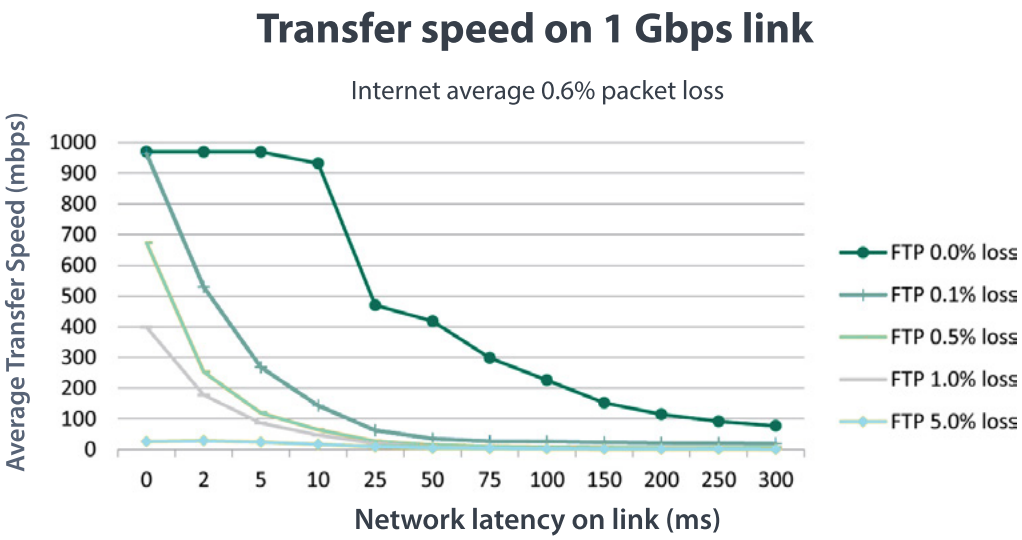

Image 2: The Effects of Latency on TCP

Packet Loss and Its Effect on TCP

Network congestion typically causes buffer overflow on routers that are unable to handle a large amount of congestion placed on them. A router experiences buffer overflow when it does not have the capacity to accept all the incoming packets. This causes packet loss.

TCP cannot distinguish between packet loss caused by network congestion, and congestion caused by interference in wireless or satellite networks. Physical structures in the "route" of a wireless or satellite connection cause interference, and ultimately packet loss. TCP will cut the TCP window in half when packet loss is detected, which is too aggressive when inherent interference is present. The ideal solution should be able to react to congestion in a less aggressive manner.

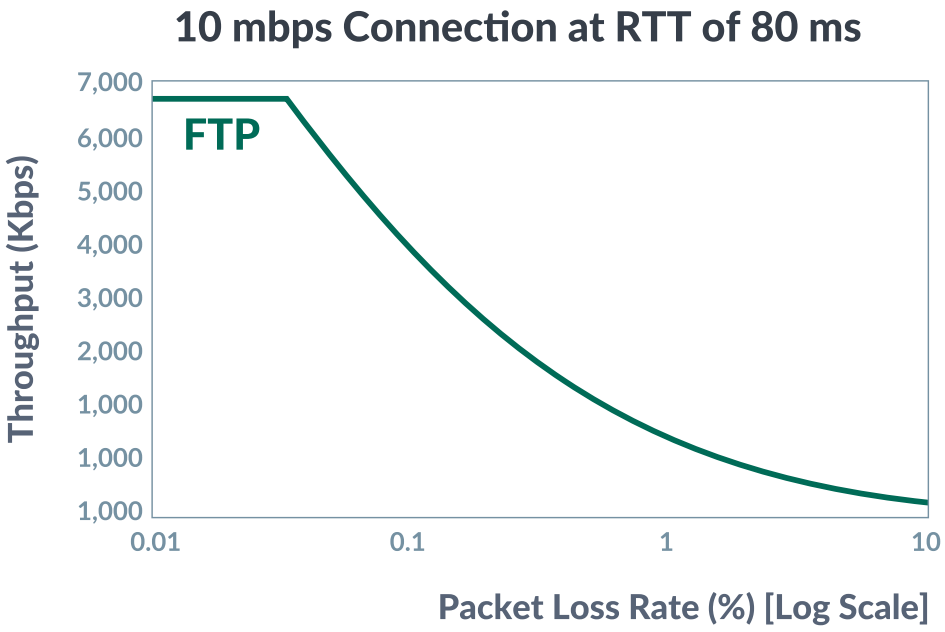

Image 3: The Effects of Packet Loss on TCP

The Pitfalls of Common TCP-Based Protocols

Speed constraints are not the only issues associated with TCP-based protocols. Some of the limitations and disadvantages include:

- **HTTP:** HTTP transfers have a size limit of approximately 2 GB. The file is placed on the computer's memory during a transfer. The larger the file, the more resource intensive the file transfer becomes.
- **FTP:** FTP does not use encryption by default when transferring files. Alternatively, FTP can be secured by using the SSL (Secure Sockets Layer) or SFTP protocols. Both SSL and SFTP are inherently secure.
- **Bandwidth Prioritization:** FTP (or any other TCP-based file transfer protocol) does not give users the ability to adjust bandwidth in order to speed up or slow down occurring file transfers.
- **Integrity Checking:** Many TCP-based protocols do not check the integrity of a file after it is transferred.
- **SMTP:** Size limits are commonly placed on SMTP transfers. This is not practical for sharing large files. If you are using your own individual mail server, however, the limits can be adjusted.
- **Blind Resuming:** When a file is paused and resumes, most TCP-based protocols will blindly append the file with no checks, sometimes resulting in a corrupt file. incomplete transfers.

Optimizing TCP

TCP makes use of two buffers referred to as "windows" to perform transfers: the congestion window on the sender machine and the receive window on the receiver machine. The congestion window is scaled up and down by the sender in reaction to packet loss. On clean links with little to no loss, the window can quickly scale to its maximum value. However, on lossy links, the window will quickly lower itself to reduce the re-transmission of redundant data. This is referred to as "congestion control." The size of the receive window determines how much data the receiving machine can accept at one time before sending an acknowledging receipt back to the sender. When a TCP connection is established, the window sizes are negotiated based on the settings on each machine. The lower value between the two machines will determine the size of the congestion window.

To optimize TCP performance, you must increase the value of the TCP windows[1]. The congestion window must be configured on the sender side, and the receive window must be configured on the receiver side. The congestion window should be tuned to maximize the inflight data and reduce the "dead air" on your link. The amount of in-flight data needed to maximize the link is called the bandwidth-delay product. The receive window should be increased to match the size of the congestion window on the sending machine.

Although tuning TCP can yield increased transfer rates, it can also create problems on networks containing packet loss. A single dropped packet will invalidate the TCP windows. When this occurs, the entire block is re-transmitted, substantially lowering the throughput of TCP transfers with large window sizes. Another possible issue is that tuning TCP for high speed transfers may reduce speeds for everyday usage such as email and web browsing.

Changing the window sizes can be a complicated process. Increasing the TCP window size involves changing configurations on both the receiving and sending parties of the transfer. This is less than ideal for environments containing multiple endpoints. On Linux-based systems, administrators must manually edit the system config files. On Windows-based systems, the registry settings must be updated. In both cases, administrative privileges will be required. Once the window sizes are increased, the optimization may be marginal. At high speeds (greater than 1 Gbps) several concurrent flows will typically be required to achieve full link capacity. With high RTT (> 50ms), 10 or more streams may be required to achieve 1 Gbps. Using this method to reach 10 Gbps or higher may require 100 or more streams, creating a strain on CPU resources.

Available Solutions

Since TCP is the backbone of many existing transfer processes, most file transfer solutions are based on TCP. These options are suited for transfer scenarios that occur spontaneously and infrequently. Some TCP-based solutions include:

FTP/SFTP/FTPS Server

Some organizations host their own FTP servers to provide their own file transfer service. This can be beneficial, but there several considerations when deploying your own server.

When sending files across the public internet, an organization's FTP server should have special security measures in place, including SSL protection for FTPS.

Even after the FTP server is deployed and configured (which takes skilled and experienced IT staff), the transfer is still performed via TCP. All the bottlenecks inherent to TCP still apply to the FTP server.

The FTP server may be able to perform file transfers, but it may lack many features included with other commercial MFT (Managed File Transfer) solutions. The drawbacks of an in-house FTP server include:

- Lack of tracking and reporting
- No email notifications for completed file transfers
- No MD5 checksum
- Unreliable resume and restart features
- No file delta capabilities
- Inability to transfer directly from the web browser

Email

In home and enterprise scenarios, email attachments can serve as an easy way to deliver files. Even though email is a common method, there are some limitations to consider, especially for enterprise applications:

- **File Size Limitations:** Email servers are configured to handle attachments of a certain size. Files that exceed the maximum size limit will be "bounced."

- **Archival Storage Costs:** Emails sent within an organization are archived on a mail server. This means that every sent attachment is archived, thus taking up space on the server. These archived attachments may greatly increase IT storage costs.
- **Poor Network Utilization:** Emails rely on the Simple Mail Transfer Protocol (SMTP). This protocol is built on TCP, so all the inherent issues of TCP are present.

Cloud Services/Digital Delivery

Cloud services such as Dropbox, Hightail and others have recently become popular methods for transmitting large amounts of data. These services are relatively easy to use and scalable, but they also have drawbacks. With cloud-based transfers, the sender must upload the file(s) to the cloud, and then the receiver must download the file(s) from the cloud as a separate step after the initial upload is complete. Since cloud-based solutions use the internet, they also suffer from the limitations inherent to TCP.

Sending structured folders containing a large number of files can be difficult via cloud services because cloud services use basic HTTP upload tools to move files. It is difficult to move complex directory structures without first zipping the payload into a single archive. Most web-based HTTP upload tools used by cloud services have a limit of 2-5 GB per file.
Even with all the limitations of digital delivery methods, a digital delivery solution is still the ideal solution when file sizes are under 2 GB. However, for a cloud solution to be truly complete, it should overcome the issues of speed caused by packet loss and latency as well as work with files of any size.

The ideal solution should accelerate file transfers and maximize the already existinginfrastructure. It should also be easy to implement and—most importantly—easy to use.

Shipping Physical Media

Rather than deploying a transfer acceleration technology, some companies use physical storage as a means of delivering data. Physical mediums include tape storage, hard drives, flash drives and DVD/Blu Ray disks.

Amazon Web Services (AWS) offers a service that physically ships storage called a "Snowball." A large hard drive is shipped to the user's location where they copy their data to the drive. Amazon then picks up the Snowball and ships it to the user's desired location.

Shipping and archiving large data sets via physical storage can work on an occasional basis, but this is not an easily scalable method. The time it takes to copy and ship the data, along with the expense of shipping, can generate high costs and inefficiency.

Common Drawbacks of Physical Media:

- **Preparation Time:** Storing and shipping data requires physical human interaction; from copying the data to printing out mailing labels and shipping. Where there is human interaction, there is also the potential for human error.
- **Cost:** Depending on the frequency and urgency, costs for using a courier service can easily add up.
- **Delivery Time:** Shipping physical media may take up to 5 business days or longer, depending on the destination.

FileCatalyst Acceleration Features

The User Datagram Protocol (UDP) can draw more performance from an IP network than TCP by omitting some of the features included with TCP. UDP is a "connectionless" protocol, meaning it does not depend on sequenced acknowledgments. Without acknowledgements, transfers have the potential to become unreliable where there is any form of packet loss.

FileCatalyst's core transport technology is based on the UDP protocol, which provides a mechanism by which data can be transmitted at precise rates. Files can be transferred via the UDP protocol without being impeded by network impairments such as latency and packet loss. UDP alone, however, doesn't have a way of recovering lost packets. In the past, there was no way to take advantage of the UDP protocol for reliable transfers over a network with impairments. FileCatalyst adds the reliability and rate control features missing from UDP, without sacrificing the desirable properties of UDP.

Like TCP, FileCatalyst also breaks data into blocks. The major difference between FileCatalyst and TCP is that with FileCatalyst, there is no delay while waiting to receive a block of data before commencing subsequent blocks of data. Transmission is initiated immediately, even if previous blocks have not yet been acknowledged. Regardless of network latency, data transmission remains constant with FileCatalyst, enabling transfers to occur at full line speed.

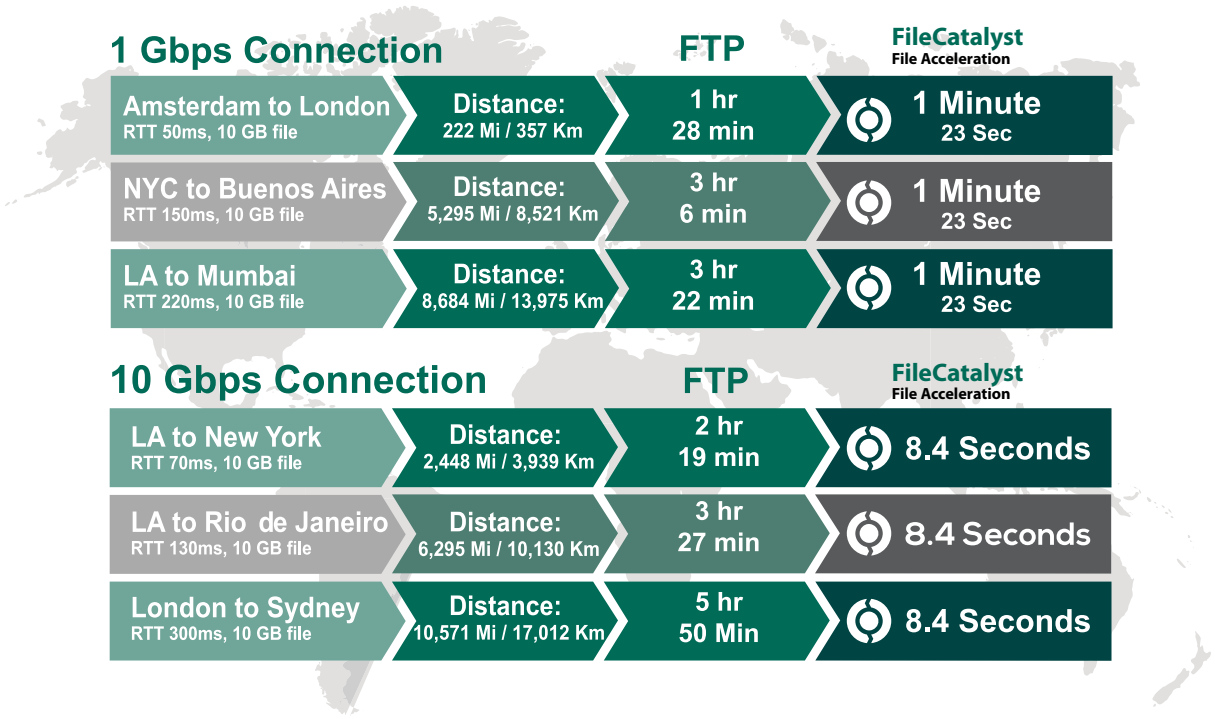

Image 5: The Benefits of Transferring a 10 GB File on a 10 Gbps Connection Using FileCatalyst

Congestion Control

Congestion control allows FileCatalyst to adapt to changing network conditions, ensuring that the transmission remains optimal and avoids congestion collapse. FileCatalyst provides three modes of congestion control: no congestion control, RTT-based, and loss-based, which is the default setting.

No Congestion Control

This setting allows users to send data as fast as possible with only minimal background traffic checks. This option is excellent for dedicated links or links configured via Software-Defined Networking (SDN) which is specific for file delivery tasks.

RTT-Based

"RTT-based" congestion control establishes a baseline average RTT before the data starts to flow. Once the transmission begins, RTT is continuously monitored. While the RTT stays within a certain range of the baseline RTT, the speed of the transfer will increase. Once the RTT begins to go above a certain threshold, the speed is decreased. How much the RTT can spike above the baseline average is controlled by the congestion control aggression setting provided by FileCatalyst.

This type of congestion control is suited for wireless or satellite links where there is packet loss from sources other than congestion. TCP will slow down, for example, when a packet is lost due to interference. When FileCatalyst is using the RTT-based congestion control, it ignores individual packet losses and focuses only on RTT.

Loss-Based

There are circumstances in which RTT-based congestion control may not work properly. For example, when a router's queue is very small the RTT may never spike when congestion is present. If the RTT remains low, FileCatalyst will continue to increase the transmission rate, even when congestion is present. For these scenarios, the only way to detect congestion is by monitoring packet loss. As outlined previously, packet loss may come from sources other than congestion, so this is best used on land-based networks where packet loss is due to real congestion.

Loss-based congestion control reacts to packet loss by slowing down, similar to TCP, but far less aggressively. TCP can be quite aggressive in its congestion avoidance, which may underutilize your link; FileCatalyst's loss-based congestion control algorithm

was designed to maximize link utilization while still avoiding congestion. Like RTT-based congestion control, the loss-based congestion control mode aggression can be tuned.

Multi-Client File Transfers

The FileCatalyst protocol can transfer multiple files from a single data source concurrently, letting an organization better optimize their computing and network resources during a file transfer. FileCatalyst's Multi-Client feature can transfer multiple growing files at once, as well as auto-archive smaller files into a single file, thereby greatly increasing throughput on data sets containing many small files.

Compression

Data compression is a general term referring to a technology that can encode large files in order to reduce their size. Not all data can be compressed, but compression can significantly shrink the size of data that can be compressed—resulting in less information to transfer. This method naturally results in a faster transfer.

Data compression applies an algorithm to the data which stores repetitive bits of information as a "shorthand." Once the shorthand is sent, the receiving end uses a decoder tool that restores these pieces back to their original state. After the decoding process is complete, an identical copy of the original file is formed on the receiving end.

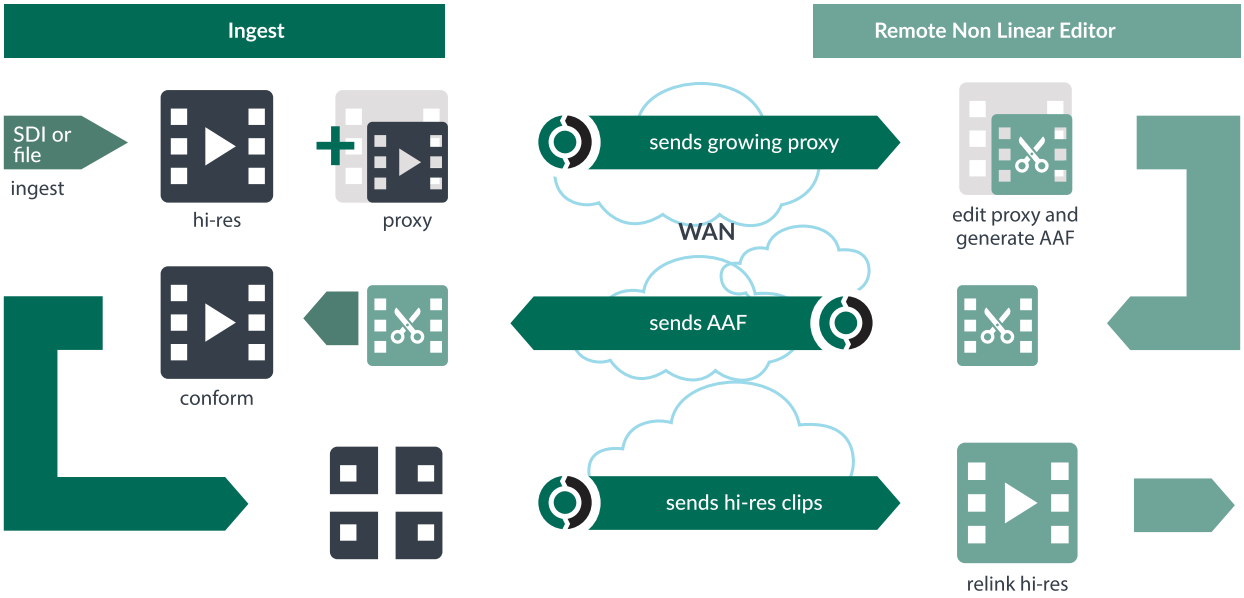

Image 6: A Live Video Production Workflow Employing FileCatalyst's Progressive File Transfer Features.

Incremental Transfers

In some scenarios, similar files may already exist on both sides of the transfer, but changes may have been made to the file at the source location. When a difference is detected, an algorithm calculates the differences between the source and destination and stores them as discrete files. These small delta files containing the changes are then transferred to the destination.

Once the delta files are sent to the destination, the changes are applied to the destination file as a "patch." The patch updates the destination file, resulting in an identical copy of the revised file. The benefit is quite clear: sending a 4 MB delta instead of a 2 TB file is an incredible difference.

Progressive Transfers

FileCatalyst can send files as they are being written on the disk. This is very beneficial when the process of creating the final file takes significant time.

Progressive transfers allow FileCatalyst to transfer a file as it is being built by another application. This is especially beneficial for live video

production workflows, when the encoding process for a large video file may take several hours to complete. Without FileCatalyst, the transfer could not start until the encoding process is complete.

The progressive transfer feature, combined with concurrent and/or multi-client transfers, allows FileCatalyst to transfer several growing files at once. This feature also allows for auto-discovery of new growing files in a predefined directory.

FileCatalyst and the Competition

FileCatalyst uses a patented proprietary User Datagram Protocol (UDP) file transfer protocol which includes congestion control, bandwidth throttling, incremental transfers, security features, and real-time management.

FileCatalyst has been used for some of the largest sporting events in the world to support television broadcast workflows. FileCatalyst has been awarded two technology Emmy awards (2015 and 2016) for their work in the industry. Under many circumstances, FileCatalyst's underlying technology is faster, more flexible, and more reliable than competing solutions.

Independent research from Anhalt University in Germany shows that FileCatalyst is the fastest and most reliable protocol for networks containing latency and packet loss[2]. The study tested multiple commercial vendors including ExpeDat (Data Expedition), Tixel (TIXStream), Catapult, Velocity, and FileCatalyst. During the tests, FileCatalyst Direct was the only solution able to deliver full line speed.

Open Source

There are several open source projects that provide an accelerated file transfer solution via UDP. Some solutions are more mature than others, and they all use different technologies to solve the same problem.

Some commercial solutions that claim to use UDP acceleration have simply integrated an open source project into their core file transfer technology. These solutions inherit the strengths, but also the weaknesses, of the open source project they leverage. FileCatalyst has developed and patented a UDP-based protocol that does not use any code from any open source UDP technology.

One common problem with these open source solutions is their lack of a Graphical User Interface (GUI). Some provide only a bare-bones sender/receiver Application Programming Interface (API), meaning that the end user must compile from the source. Other solutions mayonly come with a Command Line Interface (CLI).

Another common problem with open source solutions is the lack of checkpoint restarting and automatic MD5 checksum verification. This doesn't allow file transfers to resume automatically if the link temporarily fails, which can potentially lead to corrupt file transfers. A lack of firewall traversal support through a reverse/forward proxy is also common with open source solutions. While this is not an issue for internal transfers, most organizations send files over the WAN, which will almost certainly have at least one firewall somewhere on the route.

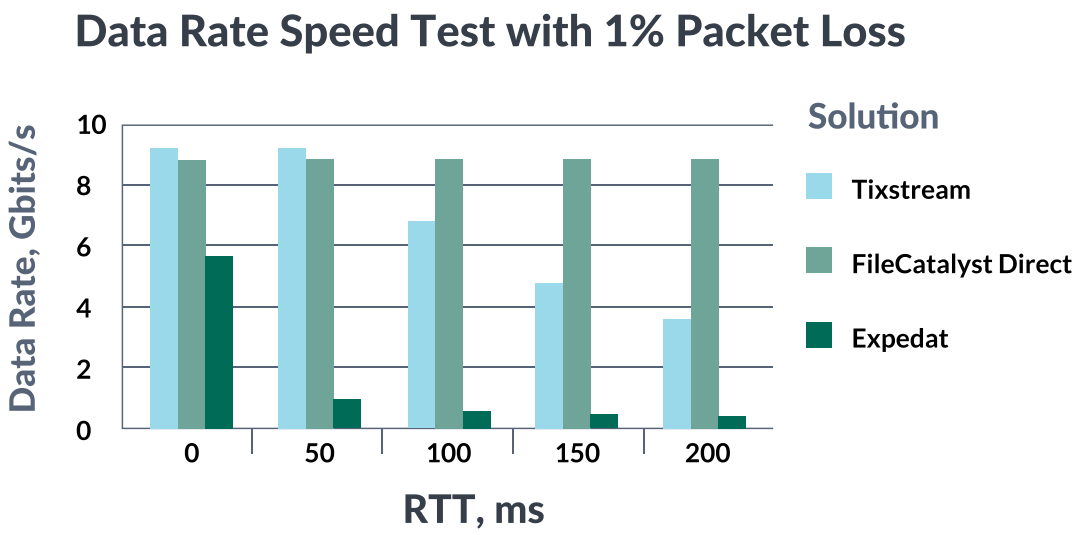

Image 7: Data Rate Speed Test Between TIXStream, FileCatalyst Direct, and Expedat

Most open source solutions do not fare well in network conditions where packet loss or high latency is present. Finally, the congestion control options included in open source UDP-based solutions do not adapt to ever-changing network conditions.

FileCatalyst Security

FileCatalyst Server requires one TCP port (default 21) to be open to inbound traffic in order to establish a control connection. The client, or connecting system, uses a short-lived port that's allocated automatically from a range predefined by the IP stack software. This is usually called an "ephemeral port," and is used as the source port to establish the control connection. Ephemeral ports typically range from port 1024 to 4999. The control connection is used to perform authentications via a username and password, share information regarding available files/directories on the server, and to negotiate protocols/ports for data transfers.

Establishing data connections, depending on your settings, requires specific TCP and/or UDP ports to be open for inbound/outbound traffic on the server side of the firewall, as well as inbound/outbound ports on the client side. FileCatalyst uses TCP for data connections when using "FTP mode" for transfers, or when the client requires a directory listing from the server. FileCatalyst uses UDP to transfer data when in "UDP mode". FileCatalyst Server defines the range of ports used for data transfers. The default port range is 8000 to 8999. However, this range may be customized to an organization's needs.

When TCP connections are established, all connections are outbound from the FileCatalyst client to the FileCatalyst Server, regardless of the direction of the transfer. As with control connections, the connecting system uses an ephemeral port as its source port and will connect to a port in the range as defined by the FileCatalyst Server.

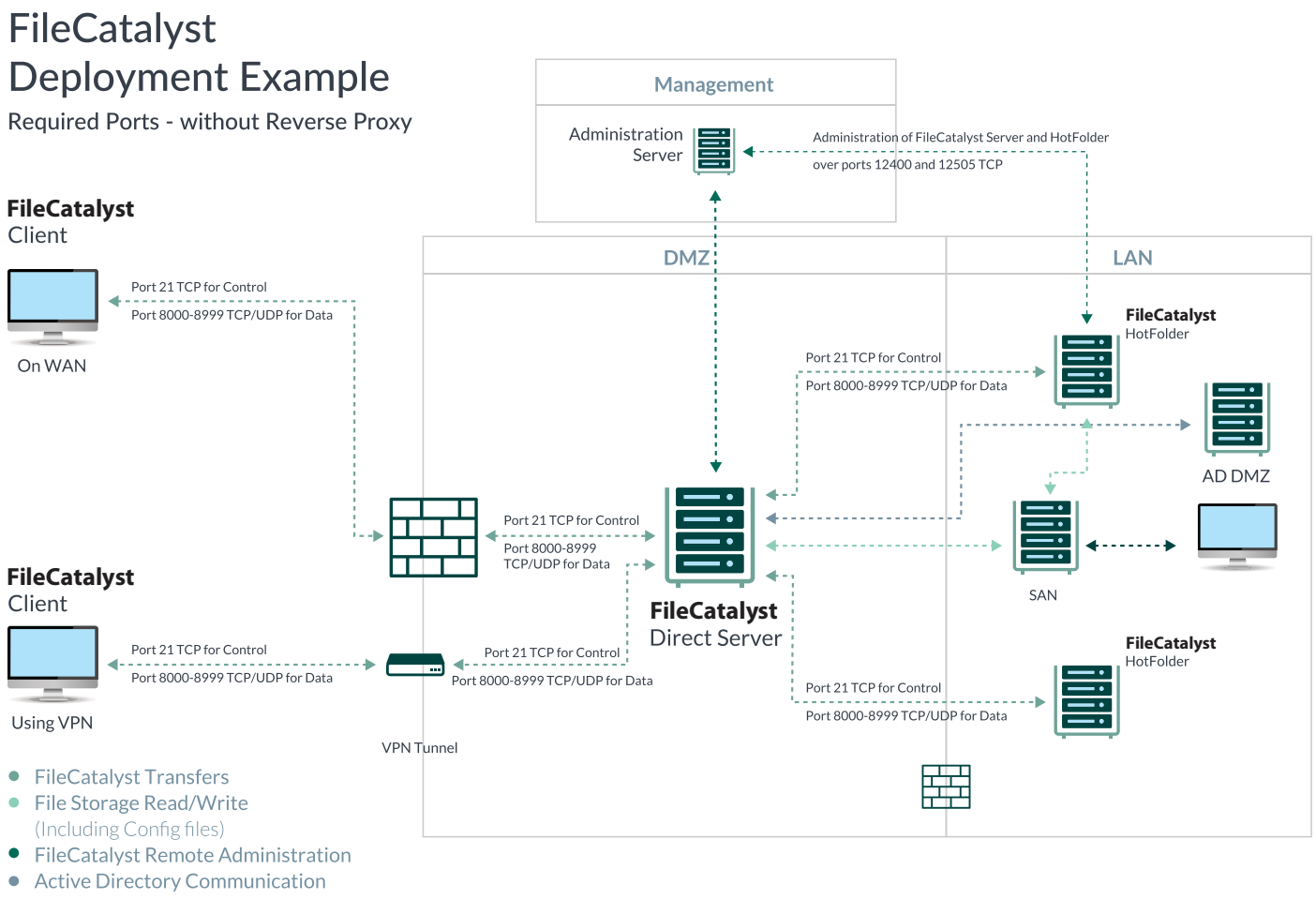

Image 10: FileCatalyst DMZ Deployment without Reverse Proxy

FileCatalyst Deployment With Reverse Proxy

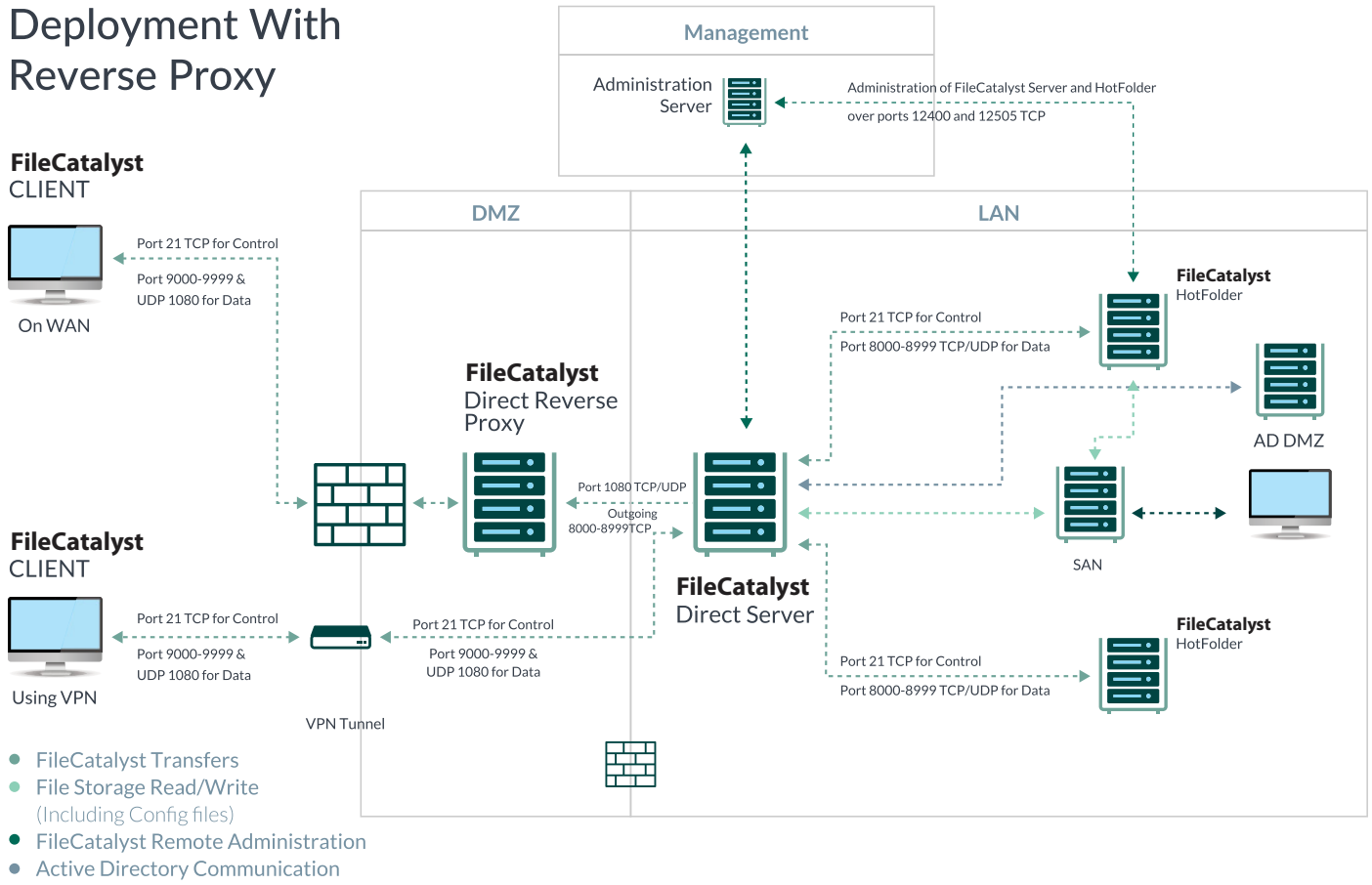

Image 11: FileCatalyst DMZ Deployment with Reverse Proxy

When UDP mode is used for uploads, the source and destination ports are in the same range (default 8000-8999). This means that the client side must allow outgoing UDP—and the server must allow incoming UDP—on all ports in the defined range.

When UDP is used to download, the source port range is defined by the server and the destination port range is defined by the client. The default client-side port is 0, meaning that the client will pick any open port. FileCatalyst uses a firewall hole punching mechanism (similar to Skype) that performs a NAT traversal. If the default values fail, or if others are required, the value may be set to a specific incoming port. Firewalls and NAT devices may be set accordingly.

Note that even when these ports are open on the firewall, FileCatalyst does not listen for connections unless it has been pre-negotiated by an existing FileCatalyst client. This mitigates the risk of third parties attempting to establish un-authorized communications with a FileCatalyst server or client.

In the scenario above, the Reverse Proxy server is deployed in the Demilitarized Zone (DMZ)and facilitates the connectivity between the FileCatalyst Direct Server, located on the secure network, and the unsecured public internet. In this scenario, the port connectivity between the internet and the DMZ remains essentially the same as when not using a Reverse Proxy. However, the connectivity between the Direct Server on a secure network and the Reverse Proxy on the DMZ requires only a single configurable outgoing port from the secure network tothe DMZ. The maximum achievable file transfer speed using a Reverse Proxy is 1 Gbps.

Minimizing Port Usage

Depending on the volume of concurrent client connections, it may be desirable to fine tune the number of ports in the data port range. One is required for every concurrent transfer taking place on the server. Thus, a range of 8000-8999 could potentially support up to 1000 concurrent transfers. If the anticipated volume of concurrent transfers is known, the port range can be adjusted.

FTP transfers on the Windows Operating System (OS) may require additional ports to support higher transfer volumes. The Windows OS does not release a closed TCP socket for up to 3 minutes but rather places it in a CLOSE_WAIT state. The consequence is that FTP transfers made up of several smaller files will quickly exhaust ports. When using FTP for transfers with a Windows-based FileCatalyst server, minimizing the data port range should be done with caution. Linux and other UNIX based operating systems do not suffer from this limitation.

Encryption

By default, the FileCatalyst TCP control connection and TCP/UDP data connections do not use encryption. All control data is text-based and sent in clear text, and should often be secured. FileCatalyst Server provides mechanisms that secure both the control and data connections. The TCP control connection and TCP data connections (FTP mode) can be secured using SSL (Secure Sockets Layer). When SSL is enabled, you must set the connection mode to "FTPS/Implicit" in order to connect with a third party FTP client.

By default, FileCatalyst generates self-signed certificates for SSL communications. Valid Certificate Authority (CA) certificates should be employed to prevent a MiM (Man in the Middle) attack. FileCatalyst's client and administration software provide options for strict validation of the domain to which it is connecting. If enabled, FileCatalyst will not connect if there is a domain mismatch.

When transferring in UDP mode, you must enable the Advanced Encryption Standard (AES)option for the data connection to encrypt data. When enabled, FileCatalyst uses AES encryption to ensure that intercepted data is useless to everyone except the intended recipient. Since AES exchanges a shared encryption key, you must also enable SSL to ensure the encryption key is not intercepted. In addition, FileCatalyst rotates the AES key for every new file in transit as an added measure of security.

By default, FileCatalyst uses 128-bit AES. However, a stronger encryption may be enabled if your country permits it.

SSL Cipher Restrictions

FileCatalyst Server allows the selection of specific SSL ciphers, which are considered appropriate for encrypted communication. The entire Java SSL/TLS set is utilized by default. The ciphers used can be modified (i.e.: enforce a minimum 128-bit encryption cipher).

IP Filters

FileCatalyst provides an IP filter feature that allows administrators to permit or deny specific IP addresses. Furthermore, administrators can permit or deny entire ranges of IP addresses. Connection attempts from IP addresses outside of the defined rules will be dropped by FileCatalyst Server instantly.

Login Security

FileCatalyst Server provides mechanisms that block brute force password attacks by automatically blocking the offending IP address and/or disabling the compromised user account. This feature may be enabled or disabled, and the administrator may set the number of failed attempts that trigger the block.

HIPAA Security Compliances

FileCatalyst can ensure HIPAA compliance from a technical standpoint by ensuring the following:

1. Access Control - The ability to access the system using user accounts. Restrictions based on IP and authentication against a directory can also help ensure access control.
2. Audit Controls - Every login is audited, and every event is logged in the system. Full reports on file transfers are available via FileCatalyst Central.
3. Integrity - FileCatalyst can ensure that data isn't compromised by performing MD5 Checksums on the files once complete. If the checksum does not match, the file is deleted on the destination side and re-sent.
4. Authentication - FileCatalyst can authenticate against a local database or a directory service. When authenticating against a directory service, no passwords are stored locally.
5. Transmission Security - FileCatalyst can use SSL and AES to ensure transmission security.
6. Reverse Proxy - The ability to configure the FileCatalyst Server on a secure corporate network using a Reverse Proxy,

deployed on the unsecured DMZ network, is a common deployment strategy for security-conscious organizations.

Penetration Testing

External parties have conducted penetration testing on FileCatalyst products. The results are proprietary and cannot be shared. However, all issues identified have been addressed. Some of the public domain issues that have been identified and resolved include:

1. POODLE
2. HeartBleed

FileCatalyst uses patched SSL libraries to solve these security issues. Always maintain the latest version of Java to ensure that newly discovered vulnerabilities are patched.

The use of Java Virtual Machine also eliminates several security problems common with Native C Applications such as Memory Corruption, Privilege Escalation, and Injection[4].

File Transfer Acceleration Scenarios

Scenario 1 - Small to Medium Enterprise (SME) Profile - Performing Large File Transfers

A company headquartered on the Eastern Coast of the USA regularly sends and receives large files to and from India and Australia. They currently rely on FTP as their main transfer method, but they also ship physical media. They have recently noticed that they aren't able to take full advantage of their connection. They currently use the following connection:

- 150 Mbps Bandwidth

Challenges - Bottlenecks Created By FTP

Since all their current file transfers utilize FTP, they can only realize about 25% of their 150 Mbps connection. Not only is their connection underutilized, but they also must deal with the inherent issues of FTP including "dead air" and packet loss. They even lost a contract due to a failed transfer.

Since they can only leverage 25% of their potential bandwidth, they are wasting 75% of their annual costs.

Solution - FileCatalyst Direct

After deploying FileCatalyst, the company accelerated their file transfers by 80%. Not only did they accelerate their connection speed and maximize their productivity, but they were also able to realize a return on their investment in less than a year and a half.

Scenario 2 - Multi-National Enterprise Profile Globally Delivering Content

A large enterprise organization distributes and publishes news content. They are headquartered in New York, with 4 branches: Los Angeles, Berlin, Moscow, and London. Their editors are Moscow, and most of the production and content creation happens between Los Angeles and New York.

Challenges - Underutilization of the connection

Their 200 Mbps link is quite fast theoretically. The average latency of pan-Pacific transfers is 200ms, with an average packet loss rate of 1%. Under these conditions, FTP will only transfer at a top speed of 490Kbps. If they can only get speeds of 490Kbps, they are only using a mere 1% of the connection they are paying for.

Solution - FileCatalyst Server and HotFolder

The company sought out various solution providers and decided to evaluate FileCatalyst. They installed FileCatalyst Server across all 4 of their locations, as well as FileCatalyst HotFolder at every location. Between every location and team project, a total of 42 nodes were set up to send and receive files.

Once the FileCatalyst Server and FileCatalyst HotFolder were installed and configured, they noticed that they could leverage FileCatalyst Central to manage and monitor every node from a web browser. Administrators can monitor and manage all 42 nodes, monitor transfers and manage alerts, regardless of which office they are located in.

They also realized that they could leverage FileCatalyst's ability to send large files through download links via email, allowing them to easily distribute assets to their coworkers between offices easily.

With FileCatalyst's UDP-based acceleration solution, the transfer speeds now reach near link speed. Production transfers that formerly took a hundred minutes to deliver now take one minute.

Scenario 3 - Cloud Profile Migrating Data to the Cloud

A research and development firm headquartered in San Diego assessed options for migratingtheir large archive into the cloud for easier remote collaboration.

They have Amazon Web Services (AWS) storage in both Europe and Asia, and want to migrate their data sets to these locations. Not only do they want to migrate their archived backups to the cloud, but they also want to perform backups on a weekly basis.

Challenges - Slow Upload Speeds

They initially began the cloud migration without acceleration, relying on the basic TCP/FTP method to transfer their archive. The initial command line tool provided by AWS had very slow throughput when transferring from San Diego to Europe and Asia. The process was deemed an unacceptably slow and ineffective process.

They considered using Amazon Snowball, by which a large drive is shipped to the HQ to have the data copied onto a physical drive and shipped by Amazon to their offices and uploaded to the cloud.

Solution - FileCatalyst Direct and HotFolder

The company chose to evaluate FileCatalyst. The installation process included an instance of FileCatalyst Server in Amazon's Elastic Compute Cloud (EC2) hosting in EC2 on a Virtual Machine (VM) in Europe and Asia. The servers were then connected to their S3 storage in the respective regions. This immediately yielded a speed increase of five times.

Not only was the speed increased, but they were also able to leverage FileCatalyst HotFolder to sync files to the server automatically. Now all the employees with HotFolder installed always have access to the latest data in their cloud storage.

References

1. "TCP window scale option." https://www.ietf.org/rfc/rfc1323.txt Accessed 6 Sept. 2017.

2. Kachan, Dmitry, et al. "Comparison of Contemporary Solutions for High-Speed Data Transport on WAN 10 Gbit/s Connections." ThinkMind(TM) Digital Library, 24 Mar. 2013, www.thinkmind.org/index.php?view=article&articleid=icns_2013_2_40_10167. Accessed 6 Sept. 2017.

3. IBM. "Big Data Technologies for Ultra-High-Speed Data Transfer in Life Sciences." IBM, IBM, 23 Dec. 2016, www-01.ibm. com/common/ssi/cgibin/ssialias?htmlfid=ZZW03369USEN. Accessed 6 Sept. 2017.

4. "Is Java more Secure then C" October 5, 2015, https://insights.sei.cmu.edu/sei_blog/2015/10/is-java-more-secure-than-c.html Accessed Sept 26, 2017

FORTRA

Fortra.com

About Fortra

Fortra is a cybersecurity company like no other. We're creating a simpler, stronger future for our customers. Our trusted experts and portfolio of integrated, scalable solutions bring balance and control to organizations around the world. We're the positive changemakers and your relentless ally to provide peace of mind through every step of your cybersecurity journey. Learn more at fortra.com.