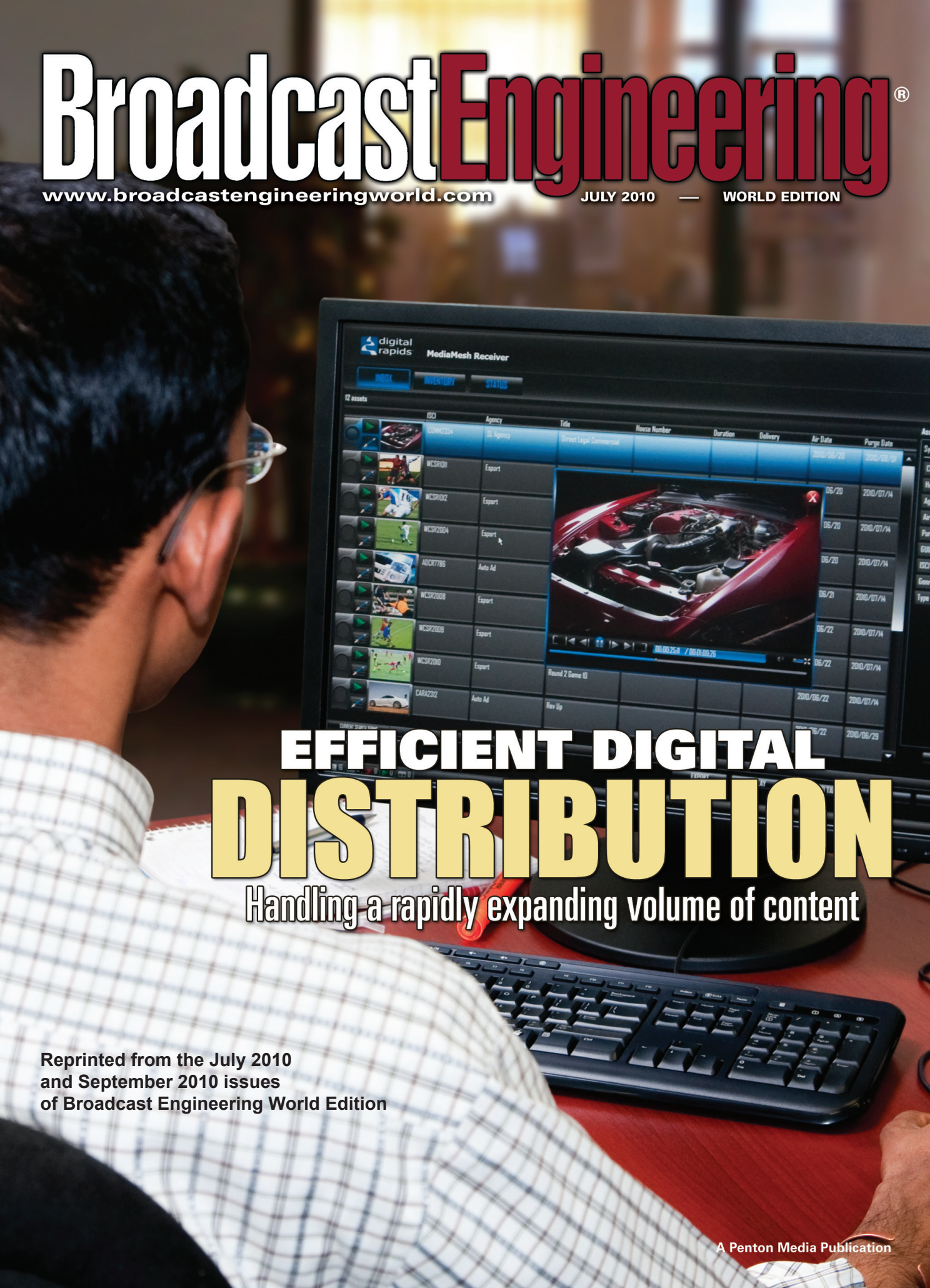
Broadcast Engineering®
www.broadcastengineeringworld.com JULY 2010 — WORLD EDITION



EFFICIENT DIGITAL DISTRIBUTION

Handling a rapidly expanding volume of content

Reprinted from the July 2010
and September 2010 issues
of Broadcast Engineering World Edition

A Penton Media Publication

Efficient digital distribution of media

The basic bottleneck on public networks is the transfer protocol.

Today's media landscape is characterized by a rapidly expanding volume of content; new opportunities and accompanying demand for wider distribution and syndication; and broadening, increasingly diverse distribution outlets. These trends bring with them the need for easier and more efficient mechanisms to deliver media between content providers, contributors, aggregators, affiliates and distribution partners.

Traditionally, the delivery of content from studios and post facilities to broadcast networks and affiliates relied on physical media and transportation, shipping videotapes or hard drives by courier. Expensive, time-consuming and labor-intensive, physical distribution is no longer efficient and has gradually been giving way to electronic distribution, although physical delivery has not yet been completely replaced. Fully digital distribution via terrestrial IP-based networks or satellite offers increased automation, greater immediacy, higher security and tremendous cost savings over tape-and-truck transport. For the benefits to be fully realized, however, there are performance impediments that must be overcome, and technical and workflow inefficiencies that can be further improved.

The breadth of distribution far exceeds the past scenario of broadcasters and affiliates. Distribution partners may also span multiple platforms and channels, including VOD providers, electronic sell-through, mobile providers and digital cinema — each of which may have specific formatting requirements. Many of these recipients do not have the dedicated network lines or satellite downlink access that traditional broadcast affiliates have had and may only be reachable over public networks. Overcoming inherent performance limitations in these networks is an essential aspect of increasing distribution efficiency, but just one of multiple factors.

In any individual transfer, speed is often limited by the receiver's bandwidth, leaving much of the sender's bandwidth unused. Optimizing unicast distribution concurrency can maximize efficient bandwidth usage, while multicasting where possible and intermediate points shared by multiple recipients can minimize the amount of data being sent across costly network links. Meanwhile, recipients' varying requirements for file conformance, compression, container and metadata formats must be addressed in an efficient and extensible manner, ideally minimizing the delivery of multiple variants while avoiding the need for multiple tools at the receiving edge. Upon receipt, workflow processes including discretionary QC, review and approval must be addressed homogeneously and efficiently as part of the distribution system.

In this two-part article, we will take a layered approach to exploring these challenges and solutions. We will first briefly look at how low-level network transport considerations affect individual transfers. In the second part we will consider optimization of overall distribution architectures to leverage concurrency in a number of scenarios. Finally, we will look at the media being moved over these architectures and conforming it to the receivers' requirements.

The basic bottleneck — the transport protocol

No distribution system can operate at optimal efficiency without maximizing the performance of its underlying transport mechanisms. Much like even the fastest courier vehicles are limited in speed when forced to drive on dirt roads in heavy traffic, the distribution of media files over IP-based

networks is limited by inherent performance impediments in the underlying communications protocols. These impediments can limit transfers to a fraction of their potential speed and reliability even on private networks, and they can all but cripple them over the public Internet, which is necessary for reaching many recipients in today's converged workflows. The inefficiencies increase exponentially with high bandwidths and long distances — significant when delivering media across the country or between countries and continents. Multicast-based transmission will be touched on in the next article when we discuss concurrency, but to understand the fundamental challenges of Transmission Control Protocol (TCP), we will first look at unicast TCP transfers between two points.

The root of these performance limitations is the nature of TCP, the protocol for IP networks upon which transfer mechanisms such as FTP are based. With TCP, transmitted data packets must be received in the correct order. To achieve this, data is sent sequentially up to the size of the recipient's receive window (buffer). (See Figure 1.) When the receive window is full, the receiver sends an acknowledgement back to the sender. (See Figure 2.) This acknowledgement must arrive at the sender before more data can be transmitted.

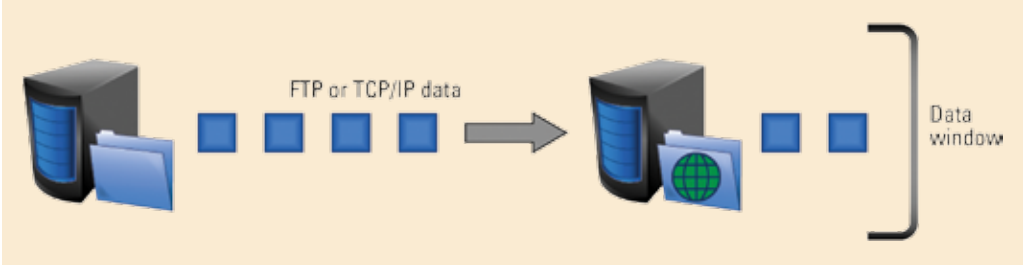


Figure 1. TCP data is transmitted until the receive buffer is full.

This round-trip time, the time between sending the data and the sender receiving acknowledgement of its receipt, is referred to as latency, and it can dramatically limit transfer speeds irrespective of available bandwidth. While local area networks and intra-area links (such as those within a major city) typically have latencies of less than 10ms, latency is particularly problematic over long distances. Transmission over the public Internet between the West and East Coasts of the United States may have latency between 80ms and 100ms. Links between continents typically have latency in excess of 120ms; a tested connection between sites in Toronto and China during the writing of this paper reported more than 280ms. There are many factors influencing this latency — the number of network "hops" between the sender and receiver, the characteristics of each hop, configuration of the routers along the transfer path, the sender and receiver's effective proximity to high-speed backbones, the sender and receiver's particular connectivity, etc. The practical result is the same: The latency between any two transfer points is a significant factor in the overall throughput of TCP-based transfers such as FTP.

Another key factor constraining network throughput is packet loss, when a transmitted packet does not reach its destination. Loss can be caused by many factors, but it's typically network congestion — essentially the overloading of a point anywhere in the network between the sender and receiver. The receiver may request retransmission of a lost packet if it recognizes this condition, or the sender may automatically retransmit if no acknowledgement has been received within a defined time period, which clearly must be at least as long as the network latency or every packet would be retransmitted. With basic TCP/IP, the entire receive window will be retransmitted even for a single failed packet within it or lost successful acknowledgement, which further reduces efficiency and causes large amounts of bandwidth to be wasted resending data that had already been successfully received. In addition, these retransmissions must occur before subsequent packets can be transferred, further slowing throughput.

TCP responds to packet loss (or perceived packet loss, which may be the result of increased latency or other factors) by significantly lowering its transmission rate on the assumption that the loss was caused by congestion, although that is not always the cause. The "slow start" nature of the TCP protocol, necessary to avoid congestion with other sessions competing for bandwidth, means that TCP does not ramp its performance back up nearly as aggressively as it throttles it. This creates further

inefficiencies in performance, particularly for short sessions. In scenarios where multiple TCP sessions are contending for resources, over-subscription — exceeding the sustainable reliable capacity — is also extremely common.

High amounts of loss can cause TCP-based applications such as FTP to fail completely. Combining the effects of latency and loss, FTP transfer throughput may max out at 1.5Mb/s or less on typical nonlocal networks even with 45Mb/s of available bandwidth. Most notably, increasing the bandwidth available will not help because it will not reduce the loss or latency that is responsible for slowing down the transmission.

There are many other considerations that affect the throughput and efficiency of network transfers, but the above is likely sufficient to convey the overall inefficient nature of TCP for high-speed transfer of large amounts of data such as media files. The net result is poor bandwidth use — in general, less than 30 percent and often less than 10 percent for long-distance delivery.

One approach to overcoming these limitations and maximizing the throughput and efficiency of file transfers over IP networks is to use the User Datagram Protocol (UDP) network protocol rather than TCP. UDP transmission does not require acknowledgements from the receiver at the network level; any reliability management and error checking must be performed at the application level. As packets can be continually sent without waiting for an acknowledgement from the receiver, the problem of network latency is overcome. The sender (transmitter) is able to use a much greater percentage — in fact, almost all — of the available network bandwidth.

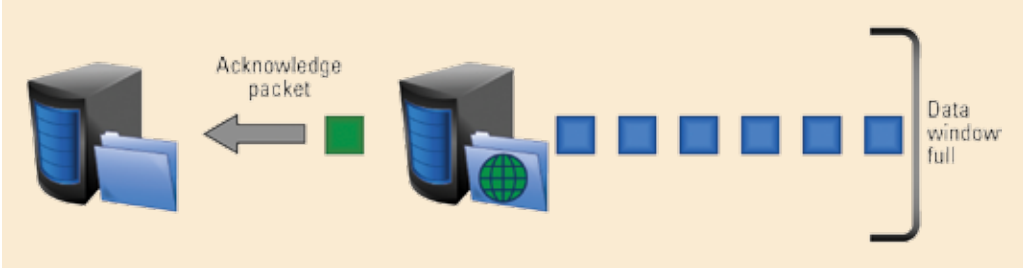


Figure 2. An acknowledgment is sent to the transmitter when the receive buffer is full.

As the UDP protocol does not ensure that packets will be delivered in their original transmission order, it is up to the receiving application to ensure that packets are reassembled in the correct order to reconstruct the transferred file. The transfer application is also responsible for handling packet loss, as the UDP protocol does not have inherent retransmission. When the receiving application determines that it is likely that a packet has been lost, retransmission can be requested as part of periodic status updates that are sent back to the sender. The amount of data resent is equal to the amount originally lost, minimizing overhead.

Because the UDP protocol lacks the acknowledgement-based flow control of TCP, UDP transmission can saturate available network capacity, creating congestion and thus causing packet loss. Repeatedly doing so creates one of the key performance inhibitors that we're attempting to overcome, degrading performance. The impact of high loss can be greater than the performance gained by overcoming latency, so the result may actually be slower than TCP-based FTP. Many UDP-based transfer solutions use measurements of loss to throttle transmission, but in the effort to continually maximize throughput, loss is effectively intentionally created as a measurement technique. Beyond the impact on the transfer itself, this congestion and the resulting loss is also unfriendly to other traffic on shared networks. To avoid this, it is best not to rely upon packet loss as the sole rate determination measurement. Instead, monitor additional network conditions, and adjust the transfer rate accordingly.

A deeper look at techniques for increasing transfer efficiency beyond standard TCP is beyond the scope of this paper. The key takeaway is that overcoming the inherent limitations of TCP-based transmission is a necessary component of improving digital content distribution over public networks, but it is just one aspect of maximizing overall distribution efficiency. In the next article, we'll look at concurrency optimization and receipt-based conforming.

BE

Concurrency & conforming

Mesh topologies and rewrapping media essence at the receiver can increase efficiency.

In part one of this look at distributing digital media files, we discussed overcoming impediments inherent in the TCP/IP protocol to improve the performance of individual file transfers. (See "Efficient digital distribution of media" in the July issue.) With improved performance for any given transfer between two points achieved, we must now extend this into real-world applications, where content is distributed to many — even hundreds — of destinations. We will look next at optimizing distribution architectures for delivery concurrency and addressing recipients' differing media format requirements.

The second stage — concurrency

Ideally, multicast could be used to significantly reduce the total amount of data transferred. While multicast is viable on private terrestrial IP or satellite networks, it is not practical over the public Internet. As a result, delivery to multiple recipients reachable only over the public Internet generally involves multiple IP unicast transfers.

For a content owner or provider delivering content to large numbers of distribution partners, the next best thing to the ideal of multicast is a hybrid model — multicast over satellite to those who can receive it, multicast over IP networks where possible (such as private networks) and multiple IP unicast transfers to the remaining recipients. The transfer application should dynamically acquire information about the receiving capabilities of each destination point or user and concurrently manage a hybrid mix of transfer types to reach all destinations optimally.

With multiple IP unicast transfers almost inevitable in broad distribution schemes, optimizing transfer concurrency is a key to efficiency. It is also significant that a location that typically functions as a receiver may also sometimes function as a sender. For example, a local broadcast television affiliate receiving national network content may also be a contributor of local news coverage to other affiliated stations in the region. In this case, while multiple stations receive content from a central point, there are also transfers directly between some of these stations.

To demonstrate ways in which transfer concurrency can be made more efficient, consider a number of different scenarios for multiple unicast transfers. All of the below scenarios involve a source location, A, delivering a file to three destinations: B, C and D. Assume that the outgoing available bandwidth at A is at least as great as the incoming bandwidth of each receiver, as this is typically the case for senders that frequently distribute large amounts of content. For simplicity, assume that connections between any of these points share the same characteristics for packet loss and that an optimized transfer system is being used to overcome latency and achieve high utilization on the network.

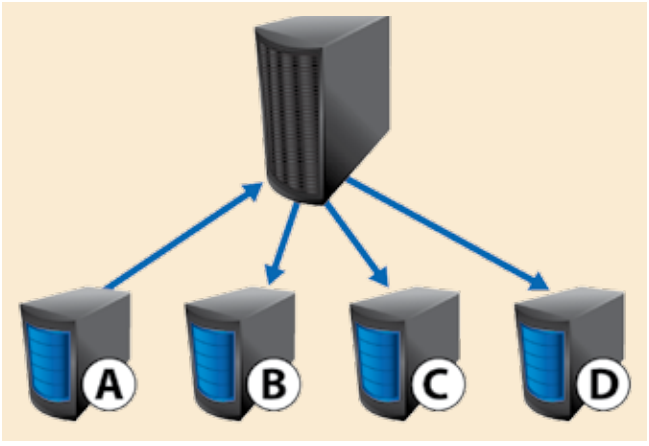


Figure 3. Scenario 2 uses a central server in a star configuration.

Scenario 1: Consecutive (nonconcurrent) transfers

In this scenario, content is first transferred from A to B, then A to C and finally A to D. This is clearly the least efficient, as there is no concurrency. Such an approach could be acceptable if all points have equivalent bandwidth or if the sender's outgoing bandwidth is lower than the incoming bandwidth of each recipient. In such cases, the sender's outgoing bandwidth is the constraint on overall performance. This is unlikely when content is regularly distributed from a centralized source, however, as the sender will typically have considerable bandwidth dedicated to distribution.

As such, transfer rates are typically limited by receiver bandwidths; if the sender has 45Mb/s outgoing bandwidth and a receiver only 10Mb/s, utilization of the sender's outgoing bandwidth can be less than 25 percent during that transfer. Another problem is that the total time before all destinations have received the file increases with the number of recipients, irrespective of the available bandwidth at the sender. Requirements for timeliness of content may make this impractical.

Scenario 2: Star (central server)

In this architecture, similar to a star network topology, all senders and receivers access a central server. A content file is uploaded to the server and is distributed to all destinations. (See Figure 3.)

In this example, the file would be first transmitted from A to the server; the server then delivers the file to the three recipients. This may be advantageous from the persp-

tive of the sender, as the total amount of "good" data (ignoring overhead, resends of lost packets, etc.) transmitted from A is the size of the file, similar to a multicast transmission. From the overall system perspective, however, the transfer to the server increases (by the size of the file) the total amount of good data transferred. This could be negligible overall if there are a large number of unicast recipients (with more than 100 recipients, the extra transmission adds less than 1 percent overall), but it will be significant with a small number (33 percent additional overhead in this example).

This still could be beneficial if the bandwidth of sender A is the overall bottleneck, as it only delivers the file to one recipient directly. Again, though, where A is a common distribution source, it will likely have considerable dedicated bandwidth. Another scenario in which this topology may be beneficial is if the link from A has high transit costs (such as a private network link) or unreliable performance. In such cases, minimizing the transfer from A can result in cost savings or increased overall throughput. Also note that as the central server is linked to all senders and recipients, it can work to optimize the overall throughput of the system.

However, all that is achieved in this scenario is to defer the concurrency problem, moving it from A to the central server. This topology does not offer a direct connection between the transferring locations. An additional file transfer intended to move from B only to C would need to go through the server, doubling the data transfer required.

Scenario 3: Network mesh

An improved scenario employs a network mesh topology in which all points can connect directly to each other. (See Figure 4.) File transfers take place directly between sending and receiving locations. By separating system-level management from the transfers themselves, a central server can still be used to monitor the network links between each point and control each point to optimize individual transfers for overall throughput but without the server actually performing any file transfers.

This topology can use all available bandwidth at A and keeps the total overall amount of good data transferred to the unicast minimum while reducing the overall time before all recipients have received the file. This architecture also allows the formation of a star-like topology within the mesh, where a receiver also functions in a fashion similar to the server in Scenario 2. (See Figure 5.) The transfer to that receiver isn't additional overhead, however, because it needs to receive the file anyway. Furthermore, the potential advantages of Scenario 2 can be maintained, such as when A has limited bandwidth or high outbound transit costs. Finally, additional transfers directly between points can be performed easily.

Scenario 4: Network mesh and shared reception points

As a further extension to Scenario 3, consider A, B, C and D as master transfer points ("engines") in separate local areas. As previously discussed, transfers on a LAN or within a metropolitan area can be more efficient than those over significant distances because of factors such as lower packet loss. If private networks are being used (as opposed to the public Internet) for long-distance transit, the cost of transfer on local network links can also be much less than that over the long-distance links (or free in the case of a LAN). In this scenario, multiple

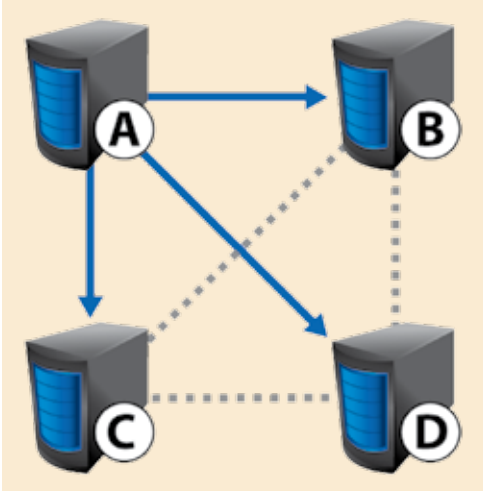


Figure 4. Scenario 3 uses a mesh topology.

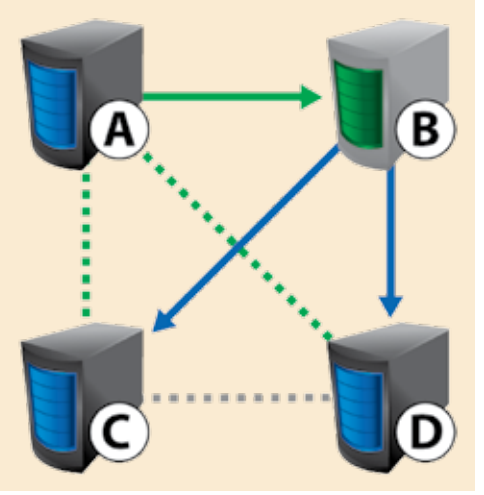


Figure 5. Scenario 3 can use a star topology within the mesh.

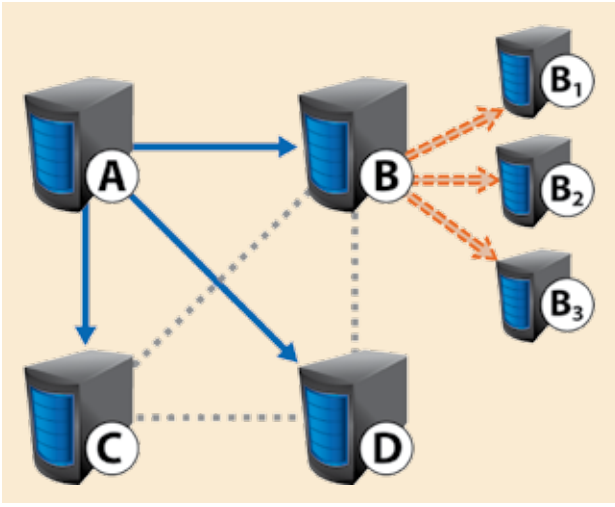


Figure 6. Multiple clients can share a node on the mesh.

recipients (clients) within the same local area can share a particular engine using a star-like topology. (See Figure 6.) The engine acts as the server for that local area, with the file transferred once to the engine and then from the engine to the local clients. Unless the engine is one of the final recipients, this does add to the total overall data transferred but minimizes the amount transferred on the lower-performance or more costly external network (as opposed to sending directly to each client from the origin).

Conforming to recipient requirements

The preceding sections can be applied to transferring any type of data. With digital media, additional considerations affect the overall distribution beyond raw data delivery. Recipients may have varying conformance requirements for compression format, media container format and metadata. Even just looking at broadcast affiliates, there will be a variety of brands and models of playout servers in use, each with its own requirements.

Delivering variants of the content directly from the sender in each required output format can hinder distribution efficiency and increase the amount of data transferred, particularly where multicast could have been used. (See Figure 7.) As rewrapping media essence in a different container format can be nonlossy, the ability to conform a transmitted media package at the receiving end (such as rewrapping it from MXF to LXF) allows the sending of a common master format without any visual degradation, reducing the number of variants that must be sent. Content delivery solutions now exist that integrate functions such as rewrapping and transcoding, as well as output to tape, directly within the receiving appliance.

There may also be a require-

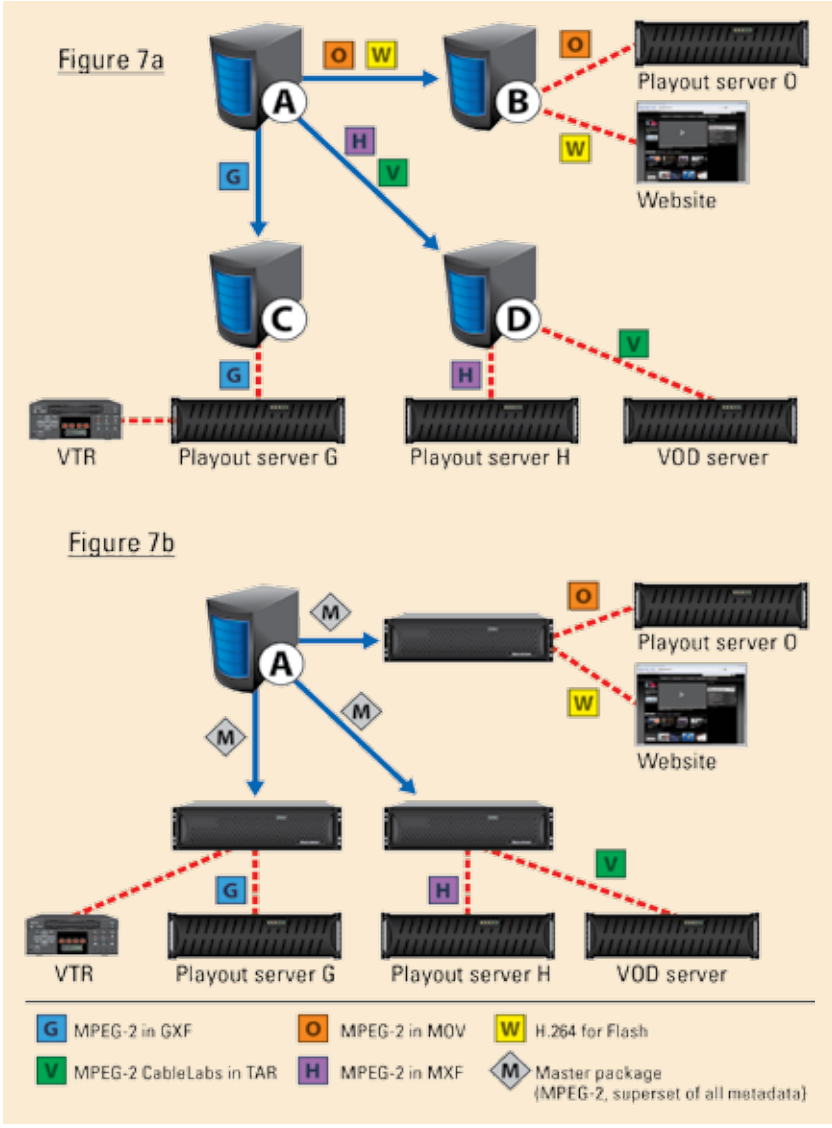


Figure 7. Transcoding and rewrapping to suit a client can be performed at the receiver.

ment to transcode into a different compression format, frame size or bit rate. For example, some distribution partners and affiliates may hold the rights to the content for both local television and their own website. While transcoding between compression formats for content intended for the same display size and type may not be ideal for quality because of re-

The metadata distributed with a master package should be a superset of all required metadata.

compression artifacts, the use of a source with a sufficiently high bit rate can mitigate these issues (such as transcoding a 50Mb/s MPEG-2 source down to 6Mb/s H.264). Similarly, content destined for lower resolutions and bit rates (such as a website) can typically be transcoded from a higher source without significant loss of visual quality. Repurposing the asset at the receiving end eliminates the need for transmitting separate versions for each platform from the source, increasing delivery efficiency. Of course, sending multiple variants from the source can still be done when maximum quality must be maintained.

As target output platforms go beyond just broadcast servers, there may also be unique metadata required by some recipients. For example, Web or mobile publishing platforms may require additional descriptive or technical metadata beyond what is normally associated with broadcast content. As such, the metadata set supported by the distribution system must be extensible with these unique additions, and the metadata distributed with the master package should be a superset of all requirements across all recipients. This may require that the metadata be sent in a format such as XML rather than within a standard media container.

Conclusion

Maximizing efficiency in digital content distribution systems is more than just accelerating transfer performance; concurrency optimization and local conforming are key contributors to achieving the benefits that digital delivery offers. **BE**

Brian Stevenson is director of product management and Mike Nann is director of marketing/communications at Digital Rapids. Originally presented at the 2009 SMPTE Technical Conference.