# Advances in reliable file-stream multicasting over multi-domain Software Defined Networks (SDN)

Yuanlong Tan*, Shuoshuo Chen†, Steve Emmerson‡, Yizhe Zhang*, Malathi Veeraraghavan*
*University of Virginia, Charlottesville, VA, USA, {yt4xb,yz6me,mv5g}@virginia.edu
†Google, Inc., Mountain View, CA, USA, shuoshuo@google.com
‡University Corporation for Atmospheric Research, Boulder, CO, emmerson@ucar.edu

***Abstract*—**

**In prior work, we proposed a cross-layer architecture called Multicast-Push Unicast-Pull (MPUP) for Software Defined Networks (SDN) to support a reliable file-stream multicast application. In this work, we improved the algorithms used to set parameters: transport-layer sender retransmission timer, VLAN rate (which is also the sending rate) and sender-buffer size. Experimental evaluation using feeds with metadata collected from real meteorology file streams was conducted. A significant finding is that the throughput achieved is smaller than the VLAN/sending rate even though file blocks are multicast continuously in UDP datagrams. Sender-buffer waiting times and propagation delays are the main reasons for the degraded throughput. For example, increasing the VLAN rate from 20 Mbps to 500 Mbps, reduced the degradation from 90% to 45%. However, the degradation increased from 45% to 58% when the VLAN rate was increased from 500 Mbps to 1 Gbps. We found an increase in the number of block retransmissions at the higher rates, which explains this increased degradation. Increasing RTT from 0.1 ms to 100 ms caused throughput to drop from 274.8 Mbps to 27.6 Mbps on a 500 Mbps VLAN. If transmission delay was a significant component in total latency, then throughput degradation relative to VLAN rate would be small; however, the meteorology file-streams used in our study have small-sized data products. Due to bandwidth borrowing between VLAN and IP-routed services, VLAN utilization is not important, and hence we recommend using the smallest rate at which sender-buffer waiting times are insignificant.**

***Index Terms*—File distribution; Layer-2 networks; Reliable multicast; SDN**

## I. Introduction

There are interesting use cases for reliable multicasting of file streams. For example, scientists engaged in atmospheric research subscribe to real-time meteorology data streams that are sent almost continuously by University Corporation for Atmospheric Research (UCAR). UCAR runs the Unidata Internet Data Distribution system (IDD) [1] project to collect and distribute large amounts of meteorology data on a near real-time basis. Examples of meteorology data include radar data, satellite imagery wind profiler data, lightning data, and high-resolution model data. Currently, there are 576 sites in 230 domains disseminating near real-time earth observations in this IDD project. Over 30 types of file-streams (called feedtypes) are distributed in this Unidata IDD project, transmitting data products (small files) at an average rate of 45 GB per hour, with roughly 439,000 products per hour.

Currently the Unidata IDD project uses Application Layer Multicasting (ALM) in which unicast TCP connections are used from the senders at UCAR to each of the receivers. Some receivers serve as relay points in an ALM tree. This solution does not scale well with increasing number of subscribers and data size. Therefore, network multicast solutions are of interest to this IDD project team.

Unlike in ALM where a sender creates multiple copies of the data products at the application layer, in network multicast, switches create multiple copies of incoming packets and forward each of them to different ports, each headed to a different set of destinations. There are two aspects to network multicasting: data-plane packet forwarding, and control-plane routing protocols (to add entries to switch/router forwarding tables to create the multicast trees). IP multicast is a Layer-3 packet forwarding solution in which the destination IP address in the IP header is used for the table lookup to determine outgoing ports. Routing protocols such as Multicast Source Discovery Protocol (MSDP) [2] are used to create forwarding table entries for multicast destination IP addresses. MSDP and related protocols were designed for distributed implementation on IP routers. Creating inter-domain multicast trees using these protocols was challenging, and therefore, even though routers included features in hardware and software to support IP multicast, most service providers do no enable this capability.

With the advent of Software Defined Networks (SDN), it became easier to configure multicast trees in the forwarding tables of switches from centralized SDN controllers. Inter-domain multicast trees can similarly be configured through coordinated operations in SDN controllers, one in each domain. These multicast trees could be configured for Layer-3 or Layer-2 (L2) packet forwarding. In our study, we used L2 Virtual LANs (VLANs) along with MultiProtocol Label Switched (MPLS) paths to configure inter-domain multicast trees for the specific advantage of in-sequence packet delivery.

In prior work [3], we presented a cross-layer Multicast-Push Unicast-Pull (MPUP) architecture for supporting reliable file-stream multicasting over SDN networks. Specifically, the solution includes a transport layer protocol called File Transfer Multicast Protocol (FMTP) [4], which runs on top of UDP and Circuit TCP (CTCP) [5], and leverages link-layer traffic control to pace out packets at the rate used by the SDN controller to configure VLAN/MPLS paths in switches. FMTP

creates blocks from the application data and sends each block in a UDP datagram over the multicast tree. Since in-sequence delivery is guaranteed on VLAN/MPLS paths, the sender does not need to track acknowledgments from receivers as with TCP since this approach is not scalable in a multicast setting. Instead receivers in the multicast tree can determine if a block is missing (due to errors or packet drops) from the block sequence number, and send a negative acknowledgment to the sender. The latter then uses CTCP to transmit the lost block to the particular receiver on a unicast connection.

The *problem statement* of this paper is to design algorithms to determine the best values to use for the parameters of this cross-layer solution taking into account multiple input variables. The *parameters* include the FMTP sender retransmission period (while 100% reliability is desired, if the path to one sender has high packet loss rates or if a sender is slow at processing incoming packets, serving block retransmissions to individual receivers without a time limit could impact FMTP throughput in serving new products to all receivers), VLAN and corresponding multicast sending rate, and transport-layer and link-layer buffer sizes at the sender to hold products as incoming file streams can have bursty product arrivals. The *input variables* include file-stream characteristics (size of data products, inter-arrival time between data products within a file stream), number of receivers in the multicast tree, and packet loss rates and round-trip times (RTTs) on paths from the sender to each of the receivers.

The *solution approach* is to develop algorithms/heuristics for finding ideal values for parameters and evaluate these heuristics by measuring output metrics such as throughput, FMTP File Delivery Ratio (FFDR, a measure of how successfully FMTP delivered data products), and latency. For evaluation of these algorithms, we used an application-layer software package called Local Data Manager (LDM) [6]. This package is used by the Unidata IDD (for North-American data distribution) and other IDD projects, such as IDD-Brasil (South-American peer), IDD-Caribe (Central-American peer), and Antarctic-IDD, US government agencies such as NOAA, FAA, NASA and USGS, commercial entities, and foreign Met offices such as the UK office[1]. We integrated our MPUP solution into this LDM application. We then executed this new version of LDM, LDM7, on the NSF Chameleon testbed [7] and emulated different network paths (by varying packet loss rate and RTT). The collected measurements were used to compute throughput, FFDR and latency for different configurations of the input parameters. Statistical analysis was then used to test hypotheses, which led us to formulate methods for selecting parameter values.

The *key contributions* of this work are as follows. First, a trial deployment was tested, and we found it feasible to use network multicast trees across multiple domains with L2 SDN services. Second, a measurements-based approach to compute the FMTP sender retransmission timer was proposed and evaluated. Third, a new algorithm was designed to choose an appropriate value for the VLAN/sending rate and sender buffer size. Fourth, an analytical study of the measured values was used to characterize relationships between output metrics and input parameters. Finally, our prior work used only one feedtype, while in this work, we ran one experiment with multiple feedtypes. Since the traffic characteristics of input file streams affect performance, this extension was important for the general applicability of our solution.

[1]http://www.metoffice.gov.uk/public/weather/forecast/world

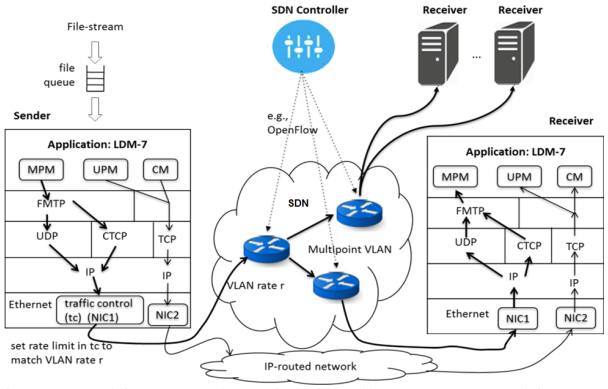



Fig. 1: Multicast-Push Unicast-Pull (MPUP) Architecture

Section II provides an overview of the MPUP architecture. Our trial deployment of the LDM software, modified for the MPUP solution, is described in Section III. Section IV describes our methods for choosing values for the FMTP sender retransmission timer, and the VLAN rate (and sender-buffer size). Section V describes our experiments and discusses the results. Section VI reviews other related research. Section VII concludes the paper.

## II. Cross-layer Architecture: MPUP

Fig. 1 illustrates a modified version of the cross-layer Multicast-Push Unicast-Pull (MPUP) architecture proposed in our prior work [3]. The specific changes made are highlighted in the description below. Network services, transport-layer, application-layer components, and cross-layer aspects are covered below.

### A. Network services

Two networks are assumed in this system model (as shown Fig. 1): IP-routed network and an SDN. For WAN deployments, the IP-routed network is the Internet, and an example of SDN that offers L2 services with which multicast trees can be created is the Internet2 Advanced Layer 2 Services (AL2S) network [8]. As the Internet offers the ubiquitous IP-routed best-effort service, this service is leveraged for reliability reasons, while the basic file-stream multicast and unicast block retransmissions are carried out on the SDN. Internet2 offers both AL2S and IP-routed services on the same physical network, as do other providers.

Fig. 1 shows the details of the protocol stack inside the sender and a receiver. Each sender and receiver has two NICs (these could physical or virtual). Applications in the end hosts choose NIC1 for SDN-based services and NIC2 for IP-routed services.

There is a Control-plane Module (CM) at the application layer, which communicates with the SDN controller to establish an L2 multicast tree[2] across the SDN. Individual receivers can be added to, or dropped from, a existing multicast tree as users at these receiver nodes subscribe to, or unsubscriber from, a particular file-stream. All control-plane communication occurs over the IP-routed network using standard TCP as shown in Fig. 1.

There are two reasons for using the SDN for the multicast tree: (i) in-sequence delivery, and (ii) rate guarantees. By establishing an end-to-end multicast tree, the path from a sender to any receiver is fixed, and all user-data carrying packets will follow the same path through the SDN. This feature of in-sequence delivery allows for the transport protocol to use negative acknowledgements, and thus avoid the ACK-implosion problem in multicast settings. Rate-guaranteed service does not necessarily require configuring data-plane actions such as policing or scheduling on every link of an end-to-end path. In networks with light loads, the SDN can offer a high probability of delivering packets at a given rate. In MPUP, the link-layer traffic-control functionality (e.g., Linux `tc`) is used to ensure that the sender paces packets at a fixed rate, equal to the rate used in the multicast-tree setup request sent in the control plane to the SDN controller. This ensures that the flow itself will not have bursty traffic, which can lead to packet drops in small-buffer, inexpensive enterprise switches on the path. Having a fixed rate of packet delivery on the multicast tree makes it easier to select parameters such as the FMTP sender retransmission timer, as will be illustrated in Section IV.

In addition to the CM, the Unicast-Pull Module (UPM) also uses the IP-routed network, as shown in Fig. 1. The purpose of this module is described in Section II-C.

### *B. Transport-layer protocols*

Fig. 1 shows the MPUP solution uses two non-standard transport-layer protocols: FTMP and CTCP. These protocols are reviewed below.

**FMTP**: For each file (data product), FMTP sends a Begin-of-Product (BOP) message via L2 multicast to all receivers. Next, the FMTP sender divides the file into blocks large enough to fit in UDP datagrams, and multicasts these file blocks. Finally, FMTP multicasts an End-of-Product (EOP) message to all receivers. The FMTP receiver checks each FMTP packet header and detects missing blocks. If all blocks are received correctly, the FTMP receiver sends an End-of-Retx-Reqs message to the sender. If one or more data blocks are missing, the FMTP receiver sends a Retx–Request back to the FMTP sender via the reliable unicast service over CTCP. The FMTP sender retransmits the requested data blocks via the reliable unicast service to just the requesting receiver. A sender retransmission timer is set for each file after the BOP is sent. When this timer expires, the FMTP sender stops serving all pending and new retransmission requests and sends back rejections. On the FMTP receiver side, a receive timer is set for each file when its BOP is received. If the timer expires before the reception of the corresponding EOP, the FMTP receiver immediately requests retransmissions for all missing blocks and the EOP for that file. This timer is new relative to the FMTP description in our prior work, and is required to handle the loss of one or more blocks at the end of the file and loss of EOP [9].

**CTCP**: In prior work [5], [10], we developed a simpler version of TCP called Circuit-TCP (CTCP) for dedicated circuits. Specifically, CTCP drops the congestion control part of TCP. Since bandwidth resources are reserved for rate-guaranteed paths, there is no possibility of packet drops in switches/routers, and therefore, no requirement for the CTCP sender to adjust its sending rate. As the link-layer traffic control sends packets at a fixed rate, equal to the provisioned multipoint VLAN rate, the CTCP should not increase and decrease the rate at which it sends segments to the lower layers. TCP flow control (to prevent receive-buffer overflows) is required since multi-tasking receivers could occasionally be handling some other task and hence not deplete the receive buffer in a timely fashion. TCP error control is also required for reliable transfers since even without congestion related losses, bit errors can occur. FMTP uses CTCP only for retransmission of missed blocks.

### *C. Application layer*

Fig. 1 shows three components in the MPUP application layer: Multicast-Push Module (MPM), Unicast-Pull Module (UPM) and Control Module (CM). The MPM is used by applications to interface with the FMTP layer. Details of this API are provided in an MS thesis [4]. Among other parameters, the sender MPM provides the FMTP layer the value to use for the sender retransmission timer.

The purpose of the UPM is to enable a multicast receiver that did not receive all blocks of a data product (file) to request retransmission of the file. As explained in Section I, to prevent one slow receiver from adversely impacting the overall system performance, the FMTP sender starts a retransmission timer, and all receivers are required to request retransmissions of missing blocks before the timer expires. However, given our goal to achieve 100% reliability, a back-stop mechanism is required to allow receivers to obtain files for which their retransmission requests exceeded this timer value. The UPM offers this capability via unicast standard TCP connections between the sender and each receiver that are established across the IP-routed network.

The CM handles receiver subscription requests for application file streams, and supports the functionality required to establish, modify and release multicast trees.

### *D. Cross-layer operation*

A detailed study of the interaction at the sending host between TCP, UDP and CTCP with the lower layers was

[2]With VLANs, the tree is more commonly referred to as a multipoint virtual topology since a sender can be located on any leg of the VLAN.

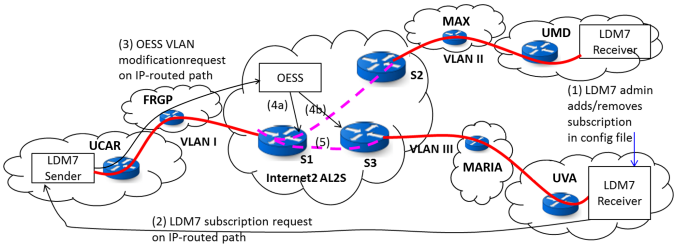



Fig. 2: LDM7 control-plane solution; black arrows: control-plane messages; red lines: provisioned VLAN segments; magenta dashed lines: dynamic MPLS paths

conducted [4]. The function that calls `tc`-layer enqueue is an IP-layer function, but this IP-layer function merely passes through the return status from the enqueue function to the calling transport-layer function. The differences in behavior of the three transport-layer protocols lie in the manner in which the calling transport-layer function reacts to a failed enqueue attempt.

Irrespective of the calling transport-layer function, if there is no space in the `tc`-layer buffer, the function responsible for enqueueing the packet will simply drop the packet, and send back a failed response to the calling function. A TCP sender will react to a failed-enqueue response by entering the CWR state, and halving the congestion window (cwnd). Applications will hold data in user-space memory if the TCP buffer is full. A UDP sender will not react to the failed-response status because UDP does not offer reliable transport service. Since data-product arrival from the application layer could be bursty, and the link-layer is moving packets out at a fixed rate, a large UDP/`tc`-layer buffer is required to hold packets. With CTCP, the sender also ignores the failed-response status as CTCP does not adjust sending rate dynamically. In MPUP, CTCP is used just for block retransmissions, and therefore a moderate-sized buffer should suffice. Experimental findings on the required UDP/`tc`-layer buffer size are presented in Section V-D2.

## III. Multi-domain trial deployment

As noted in Section I, we modified LDM ver. 6 to create a new version LDM7, which uses FMTP, CTCP and TCP, as illustrated in the MPUP architecture of Fig. 1. As explained in Section II-A, MPUP uses an SDN that supports multicast services. The availability of such an SDN is hence important for a multi-domain WAN deployment.

Internet2 and ESnet, the two major US backbone Research-and-Education Network (REN) providers, have deployed SDNs, and offer dynamic L2 (VLAN/MPLS) path services. We chose Internet2 to test LDM7 deployed at multiple university campuses. The Internet2 SDN controller is called Open Exchange Software Suite (OESS) [11]. In our current deployment, only the multipoint L2 virtual network (VLAN/MPLS) across Internet2 AL2S is dynamically established, reconfigured as needed, and released.

Fig. 2 illustrates our trial deployment at the University of Virginia (UVA), University of Maryland (UMD) and UCAR. The corresponding regional REN providers are MARIA, MAX and FRGP, respectively. VLAN segments are provisioned from the LDM7 servers in the three campuses to their corresponding Internet2 AL2S switch ports, e.g., VLAN III from UVA, manually. Our vision is that as SDN deployment increases, and campus and regional RENs follow Internet2's lead and offer dynamic L2 services, these segments can also be established dynamically. Fig. 2 illustrates how Internet2's dynamic L2 service capability is leveraged for our LDM7 trial deployment. Assume that VLAN I had been previously connected to VLAN II via an Internet2 AL2S MPLS path represented by the magenta dashed line between switches S1 and S2. Assume the LDM7 sender at UCAR is sending a particular feed to the LDM7 receiver at UMD. Steps (1) through (5) required for the UVA LDM7 receiver to join this feed are shown in Fig. 2.

We have currently deployed LDM7 on 8 campuses and have expansion plans. This trial deployment demonstrates the feasibility of a WAN deployment of MPUP.

## IV. Methods for choosing parameter values

This section describes our methods for determining how to set parameters: FMTP sender retransmission timer, VLAN/sending rate, and sender-buffer size, in a general MPUP architecture.

### A. FMTP sender retransmission timer

This parameter, $\tau_{snd}(i)$, limits the duration for which an FMTP sender serves retransmission requests for file $i$. We use FFDR to study the impact of this parameter [3]. FFDR evaluates the success of file delivery by FMTP from a single sender to multiple receivers. A file $i$ is said to have been delivered successfully to receiver $j$ by FMTP if all blocks of file $i$ were received by receiver $j$ either via multicast or via missed-block retransmissions on the unicast CTCP connections from the sender to each receiver. When FFDR is less than 100%, the application-layer UPM described in Section II-C ensures successful delivery of all files to all receivers as long as receivers request files within the specified duration for which files are served by the backstop reliable mechanism. Hence, FFDR captures the extent to which FMTP is successful in delivering files without the backstop.

In our prior work [3], this timer was defined to be dependent on file size as shown below:

$$\tau_{snd}(i) = \max(f_{snd} * s_i/r, \max_{1 \le j \le m} RTT_j) \qquad (1)$$

where the symbols are explained in Table I. While this definition considered only file transmission delay and propagation delays, we found that sender-buffering delay was a dominant factor when the VLAN/sending rate was low (e.g., 20 Mbps).

Even with a large $f_{snd}$ value of 5000, FFDR was not 100% when packet loss rate is non-zero. For example, when packet loss rate was 1%, FFDR was 89% with 8 receivers, and fell further to 77% with 16 receivers. LDM7 log files showed that a majority of unsuccessful files, i.e., files that were not fully delivered by FMTP because of sender retransmission timeout,

TABLE I: Notation

| | |
|---|---|
| **Input parameters** | |
| $i$ | file index |
| $j$ and $m$ | receiver index, and number of receivers, respectively |
| $(t, t + k\tau)$ | $k^{th}$ holding interval (during which VLAN rate and buffer size are held unchanged) |
| $a_i$ | time instant when file $i$ was delivered to sender's FMTP layer |
| $s_i$ | size of file $i$ |
| **System parameters** | |
| $\mathbb{W}$ | maximum queueing delay (waiting time) in sender buffer |
| $\mathbb{E}$ | maximum total delay for processing, switch/router buffer queueing and FMTP block retransmission delays |
| $\alpha$ | threshold for per-receiver fraction of products that exceeded latency threshold |
| $\beta$ | threshold for fraction of receivers that exceeded $\alpha$ threshold |
| $f_{snd}$ | FMTP sender retransmission timeout factor |
| $\tau_{snd}(i)$ | FMTP sender retransmission timer for file $i$ |
| $RTT_j$ and $p_j$ | round-trip time (propagation delay) and artificially injected packet loss rate from sender to receiver $j$ |
| $r_k$ | configured VLAN rate used in the $k^{th}$ holding interval; when $r$ is used without a subscript, $k$ is implicit |
| **Intermediate values** | |
| $R_k$ | ideal computed VLAN rate for the $k^{th}$ holding interval |
| $B_k$ | ideal computed sender-buffer size for the $k^{th}$ holding interval |
| $q(t)$ | sending-host buffer (queue) occupancy at time $t$ in ideal case |
| $\epsilon_{ij}$ and $\epsilon'_{ij}$ | sum of sender processing delay for packets of file $i$, queueing delays at switch/router buffers experienced by packets of file $i$ en route to receiver $j$, and FMTP block retransmission delays, actual and ideal, respectively |
| $w_i$ and $w'_i$ | sender-buffer waiting time experienced by file $i$, actual and ideal, respectively |
| $d_{ij}$ and $d'_{ij}$ | latency incurred in delivering file $i$ at receiver $j$, actual and ideal, respectively |
| $\mathbf{A}_k$ | set of files that arrived in the $k^{th}$ holding interval |
| $\mathbf{D}'_k$ | set of files that departed the sender in the $k^{th}$ holding interval in ideal setting |
| $N_k$ and $N'_k$ | number of files that arrived and departed the sender in the $k^{th}$ holding interval, actual and ideal, respectively |
| $\mathbf{V}_{kj}$ | set of files whose actual latency $d_{ij}$ exceeded threshold in the $k^{th}$ holding interval at receiver $j$ |

were small-sized. For example, if $f_{snd} = 5000$ and $r = 20$ Mbps, the retransmission timer for the smallest observed file, which had a size of 0.06 KB, would have been set to 0.12 second. This retransmission timer is so small that it could have expired even when the file was still queued in the sender buffer. If this happens, none of the receivers would have received this file, which would result in a significant number of retransmissions.

On the other hand, $\tau_{snd}(i)$ for the largest file, which had a size of 23.7 MB [12], would have been set to 13.17 hours. However, LDM7 guarantees to hold each file in its product queue for only one hour. This could cause a problem because FMTP does not hold a copy of the file in its memory space; rather it serves retransmission requests by reading blocks of files directly from the LDM7 product queue. Therefore, application constraints should be considered when deciding the FMTP sender retransmission timer.

**File-independent solution**: A simple solution is to have the application provide a single value for the FMTP sender retransmission timer to use for all files of a particular feedtype, i.e., $\tau_{snd}(i) = c$, where $c$ depends on feedtype, and the maximum time the application holds each file in its memory space.

While the application dictates a maximum value for this FMTP timer, layers below FMTP and path characteristics dictate a minimum timer value. Therefore, propagation delays, queueing delays, sender-buffering delays, and packet loss rates (which impact the time for retransmissions) should be considered when determining the minimum value for the timer. As the lower-layers-dictated timer value could be larger than the application-dictated timer value, a holistic approach is required to setting this timer value.

Our solution is to monitor the maximum latency of products in fixed intervals for each feedtype, and then adjust the FMTP sender retransmission timer if needed. In our prior work, maximum latency was measured for just one hour of one feedtype NGRID. In this work, we measured maximum latency for multiple feedtypes over multiple hours to determine whether this value changes significantly from one hour to the next for a given feedtype. Section V-D1 presents these results.

### *B. Multipoint VLAN/sending rate and sender-buffer size*

In prior work [12], we proposed an algorithm for selecting these parameters. However, there were a few drawbacks in this algorithm: (i) no method was offered to set the threshold $\mathbb{W}$ (see Table I); (ii) only a single receiver (instead of multiple receivers) was considered in the model; (iii) per-file throughput metric was used instead of an aggregated average throughput metric (the former gives equal weight to all files irrespective of size); (iv) idealistic assumptions, which are challenging to implement, were made, such as needing to determine the number of files that exceeded a waiting-time threshold at the sender, and that multipoint VLAN rate should be modified with an Exponential Weighted Moving Average (EWMA) method, and (v) VLAN utilization was considered. Our practical implementation showed that with bandwidth-borrowing (also called work-conserving) between VLAN queues and IP-routed queues, it is not important to maintain a high VLAN utilization. The implication of this finding is that the VLAN rate does not need to be decreased often, and therefore instead of an EWMA method, excessive latency violation can be monitored and VLAN rate increased only when significant thresholds are

crossed. To modify VLAN rate, signaling is required to SDN controllers, which is a high-overhead operation, especially in multi-domain paths. These drawbacks are fixed in the model presented below.

*1) Ideal buffer size computation:* Using an ideal path rate $R_k$, buffer occupancy at the time of arrival of each file is as follows [12]:

$$\begin{aligned} q(a_1) &= q(t+(k-1)\tau) \\ q(a_2) &= \max\{0, q(a_1)+s_1-R_k\times(a_2-a_1)\} \\ q(a_i) &= \max\{0, q(a_{i-1})+s_{i-1}-R_k\times(a_i-a_{i-1})\} \end{aligned} \quad (2)$$

To ensure 0 loss in the $k^{th}$ holding interval, the sender-buffer size $B_k$ should ideally be

$$B_k = \max_{1\le i\le|\mathbf{A}_k|)} q(a_i) \quad (3)$$

*2) Ideal path rate computation:* Waiting time at the sender buffer for file $i$ is:

$$w'_i = \frac{q(a_i)}{R_k} \quad (4)$$

Ideal path rate $R_k$ for the $k^{th}$ holding interval is the smallest value at which the following holds:

$$w'_i \le \mathbb{W}, 1 \le i \le N'_k \quad (5)$$

where $N'_k = \left|\mathbf{A}_k \cap \mathbf{D}'_k\right|$.

The ideal latency to deliver file $i$ to receiver $j$ would be

$$d'_{ij} = (w'_i + s_i/R_k + RTT_j/2) + \epsilon'_{ij} \quad (6)$$

accounting for sender-buffer waiting time, transmission delay, one-way propagation delay, and a small delay $\epsilon'_{ij} \le \mathbb{E}$.

*3) VLAN/sending rate adjustment:* Actual file latency $d_{ij}$ is measured from the time instant when the sender-side application provides a file $i$ to the FMTP layer to the time instant when a receiver $j$ receives all blocks of the file via FMTP. Applications can log these timestamps, allowing for a computation of actual file latency. In our application LDM7, the sender sends the arrival timestamp for each data product to the receiver, which logs this timestamp along with the reception time for the product, and Network Time Protocol (NTP) is used to synchronize clocks.

File $i$ belongs to set $\mathbf{V}_{kj}$ if $d_{ij} - (\epsilon_{ij} + s_i/r_k + RTT_j/2) > \mathbb{W}$, where an estimate of $\epsilon_{ij}$ is $RTT_j$ multiplied by the number of block retransmissions plus a small number to estimate processing delays and switch/router buffer queueing delays. A receiver $j$ is in violation of the latency requirement in the $k^{th}$ holding interval if $|\mathbf{V}_{kj}|/N_k > \alpha$. If the fraction of such receivers relative to the total number of receivers in the multicast tree exceeds $\beta$, then the sender should initiate an increase in the VLAN rate for the next holding interval. If $\epsilon_{ij}$ also exceeds $\mathbb{E}$, packet loss rate could be high. If the $\beta$ threshold is not exceeded but the $\mathbb{E}$ is exceeded, it may not be useful to increase VLAN rate since this rate may not influence the components of $\epsilon_{ij}$.

If in the SDN controller, VLAN call blocking rate becomes high, delay analysis should be conducted, and individual VLAN rates should be decreased if possible.

*4) Setting of thresholds used in the algorithm:* Methods for setting thresholds $\mathbb{W}$ and $\mathbb{E}$ will depend upon the application. We demonstrate our method for the IDD project. We propose using 2 ms (less than 10% of the propagation delay across the US) for the waiting-time threshold $\mathbb{W}$. On the other hand, if all products are small and all receivers are close to the sender, 2 ms may be a much higher percentage of the total latency. This is still acceptable for the near-real-time requirement of the IDD receiver analysis programs that consume the products as they arrive. Finally, this 2 ms number does not make the required VLAN rate too high relative to available link rates in the IDD project.

For $\mathbb{E}$, we propose using a small value, e.g., 2 ms plus $\max_{1\le j\le m} RTT_j$. The small value accounts for processing and switch/router buffer queueing delays, which are typically in microseconds in the IDD environment. The second term accounts for retransmission delay. We estimate only one block retransmission per product, given low packet-loss rates and small product sizes in the IDD project.

We recommend $\alpha$ to be 0.1 and $\beta$ to be 0.01, i.e., 10% and 1% respectively for the IDD application, but these numbers depend upon the IDD administrator's determination of user-analysis-program requirements with respect to near real-time product delivery.

## V. Experimental evaluation

### A. Setup

Chameleon, a deeply reconfigurable, NSF-supported network testbed, was used to run the experiments. For our experiments, we created a VLAN with hosts distributed in two different racks, both of which are geographically located at Texas Advanced Computing Center (TACC) at the University of Texas, Austin. The number of bare-metal hosts used in each rack was varied as we modified the number of receivers in our multicast experiments.

Each bare-metal node had 2 cores (Intel® Xeon® CPU E5-2670 v3 @ 2.30GHz), 128 GiB RAM, and 250 GB disk space. Two 10GE NICs were used in this experiment. A VLAN was stitched between the top-of-rack switches in the two racks, and the rate of the VLAN was set to 10 Gbps.

The software used in our experiments consists of: (i) LDM7, (ii) Linux traffic-control (`tc`) utility to adjust sending rate, (iii) Linux network emulation utility, `netem`, to increase RTT between sender and receivers, (iv) Linux utility, `iptables`, to inject/remove artificial packet losses by inserting/deleting DROP rules, (v) Linux utility `tcpdump`, to capture network traffic on CTCP connections, and determine the number of missing-block retransmissions, and (vi) Python scripts to parse LDM log files for file latency, throughput, and FFDR.

### B. Execution

Experiments were run to measure performance of LDM7, an MPUP implementation. Table II shows values used for four input parameters (see Table I for interpretations of the

symbols), and a fifth parameter is the specific feed (file-stream) used in our experiments.

TABLE II: Values for input parameters

| Symbol | Value |
|---|---|
| $m$ | $\{1, 8, 16\}$ |
| $RTT_j$ | $\{0.1, 10, 20, 30, 40, 50, 100\}$ ms |
| $p_j$ | $\{0, 5\}$ % |
| $r_k$ | $\{20, ..., 60\}, \{500, ..., 1000\}$ Mbps |

**Feeds (file streams)**: A data analysis of five IDD feedtypes showed that both file inter-arrival times and file sizes have long-tailed right-skewed distributions [12]. Our prior work [3] used only the NGRID feedtype to compare LDM7 and LDM6 (application-layer multicast version). In this work, for a comprehensive investigation of the performance of LDM7, we set up a receiving LDM6 server at UVA and configured it to collect metadata (size and creation-time) for six real feed types from a UCAR sending server. These feedtypes are listed in Table III [13]. Specifically, the LDM utility, `notifyme`, which allows a downstream LDM server to receive just the metadata about files in a feedtype instead of the actual files, was executed to receive metadata for the week of November 15-21, 2018. The real metadata collected for these feedtypes was used as input to a program called `pq_insert` to create dummy data products with the corresponding creation times and sizes. LDM7 was used to multicast these dummy products from a sender to multiple receivers.

TABLE III: LDM feedtypes used

| Feedtypes | Average traffic proportion | Description |
|---|---|---|
| NGRID | 22.8% | NOAA port high-resolution model output |
| CONDUIT | 17.9% | NCEP high-resolution model output |
| NEXRAD2 | 9.6% | Next-generation radar Level-II radar data |
| NEXRAD3 | 3.1% | Next-generation radar Level-III products |
| HDS | 2.5% | High-resolution data service |
| IDS\|DDPLUS | 0.2% | International data service |

The Linux `tc` utility was used for the traffic control module shown in Fig. 1. A combination of Hierarchical Token Bucket (HTB) and Bytes First In First Out (BFIFO) queueing disciplines of `tc` were used to make two queues, one for multicast and the second for retransmissions. As shown in Fig. 1, UDP and CTCP feed packets into `tc`. The UDP datagrams were directed to one queue, while packets from all $m$ CTCP connections (from the sender to each receiver) were directed to the second queue.

Our prior-work comparison of LDM7 and LDM6 with the NGRID feedtype showed that with a VLAN/sending rate of 20 Mbps LDM7 and LDM6 achieved the same throughput value. Therefore, we used 20 Mbps as a starting value for the VLAN rate $r$ with a buffer size $B$ of 600 MB to ensure that no packets were dropped by the `tc` buffer at the sender. A dropped packet at the sender will require retransmissions for all the receivers, and is hence avoided. The rate $r$ was set as the HTB `rate` and `ceil` parameters for both queues, while the 600 MB buffer size was set in the BFIFO parameter. The same values of VLAN/sending rate and sender-buffer size were used for the CTCP queue.

The LDM application Product Queue (PQ) has two parameters: $q_{size}$ and $q_{slot}$, which represent the maximum size of the PQ in bytes, and the maximum number of files that can be stored in the PQ, respectively. If a newly arriving file causes either of these limits to be exceeded, one or more of the oldest files will be deleted to make space for the new file. We selected the values 5 GB for $q_{size}$ and 35000 for $q_{slot}$ to ensure no file drops in the PQ for any 1-hour file streams.

The *experimental workflow* consists of four steps: (i) upload LDM7 software, configuration files (for FMTP and LDM), metadata of 6 feedtypes and network emulation scripts (to adjust sending rate, packet loss rate and RTT) to the Chameleon bare-metal nodes from a local host, (ii) run the software and monitoring tools on the Chameleon nodes, (iii) download collected logs from the Chameleon nodes to a local host, and (iv) run the log parsers to extract performance measures for analysis.

### C. Output measure

A file-set throughput is computed at each receiver by summing file sizes over a set of file indices and summing corresponding latencies, and dividing these two sums [3]. A set is defined to include a subset of files whose total size is close to $G$, and all files in the set would have arrived and departed within a holding interval $k$. A holding interval $k$ could have multiple sets. An averaging operation is performed to compute the average file-set throughput values across all receivers. The average file-set throughput for the $l^{th}$ subset of files that arrived in holding interval $k$ is defined as follows:

$$\Gamma_{kl} = \frac{1}{m} \sum_{j=1}^{j=m} \frac{\sum_{i \in \mathbf{Z}_{kl}} s_i}{\sum_{i \in \mathbf{Z}_{kl}} d_{ij}} \tag{7}$$

where $\mathbf{Z}_{kl}$ is a subset of files that arrived in holding interval $k$. The subset is defined by file indices $L_1$ and $L_2$, which are chosen such that $\sum_{i=L_1}^{i=(L_2-1)} s_i < G$ and $\sum_{i=L_1}^{i=L_2} s_i \geq G$, where $G$ is the aggregate file-set (group) size. A $G$ value of 200 MB was used.

In the $k^{th}$ holding interval, there will be multiple $\mathbf{Z}_{kl}$ subsets, each of which has a cumulative size close to $G$. The number of such sets in the $k^{th}$ holding interval is $\lambda_k$. The throughput for a holding interval $k$ is defined as:

$$T_k = \frac{1}{\lambda_k} \sum_{l=1}^{l=\lambda_k} \Gamma_{kl} \tag{8}$$

### D. Results

*1) FMTP sender retransmission timer:* The solution to use a constant value for the FMTP sender retransmission timer, described in Section IV-A, was evaluated on the Chameleon setup using multiple 1-hour traffic traces from the six LDM feedtypes. The purpose of these experiments was to determine whether or not the maximum latency experienced by data products within an hour for each feedtype varies significantly from one hour to the next. If measures of variation were small, and if the maximum latency was below the default 1-hour

value used by the LDM7 application to hold files in its product queue to serve retransmission requests from receivers, then our hypothesis that a per-feedtype constant value could be used for the FMTP sender retransmission timer would be validated. If either of these conditions were not met, then an alternative approach would be required to determine how to set the FMTP sender retransmission timer.

Our findings are shown in Tables IV and V, corresponding to VLAN rate settings of 20 Mbps and 500 Mbps, respectively.

TABLE IV: Maximum latency across products in three 1-hour intervals (16:00-17:00 UTC on Nov. 15, Nov. 16 and Nov. 17); $r = 20$ Mbps

| **Feed Type** | **Per-hour maximum latencies (s)** |
|---|---|
| NGRID | 64.70; 60.77; 55.73 |
| CONDUIT | 154.93; 186.90; 162.27 |
| NEXRAD2 | 0.31; 0.41; 5.31 |
| NEXRAD3 | 0.30; 3.81; 0.93 |
| HDS | 1.75; 9.47; 7.52 |
| IDS\|DDPLUS | 0.14; 0.10; 0.10 |

TABLE V: Statistics for maximum latency across NGRID products in 24 one-hour intervals (Nov. 16, 2018); $r = 500$ Mbps

| **NGRID** | **Time (ms)** |
|---|---|
| Min. | 102 |
| 1st Q | 262 |
| Median | 315 |
| 3rd Q | 458 |
| Max | 3352 |
| Mean | 509 |

Waiting times in the sender buffer can be large when $r = 20$ Mbps, and therefore maximum latency is higher in Table IV than in Table V. First, we observe that there can be significant differences among the one-hour values, e.g., for NEXRAD3, two 1-hour segments had maximum latency less than 1 sec, while a third one had a maximum latency of 3.81 s. However, all the numbers observed are smaller than the 1-hour LDM7 timer limit. Therefore, it is feasible to set a constant value, which could be different for each feedtype, for the FMTP sender retransmission timer.

*2) Sender-buffer size requirement:* In a preliminary set of sequential experiments, LDM7/FMTP/UDP was executed with the multicast HTB class rate set to 20 Mbps, and the BFIFO buffer size set to 7.2 MB, 300 MB, and 200 MB. The number of packets dropped by `tc` in each of these experiments was 623019, 0, and 48354, respectively. We then conducted a systematic experimental study by varying the VLAN/sending rate, $r$, and increasing the `tc`-layer buffer size for each setting of $r$ until the dropped-packet rate reported in `tc` statistics reached 0. Fig. 3 shows that the required buffer size dropped from 300 MB to 20 MB as the sender multicast rate, $r$, was increased from 20 Mbps to 500 Mbps. The waiting time in the sender buffer is correspondingly smaller at higher values of $r$.

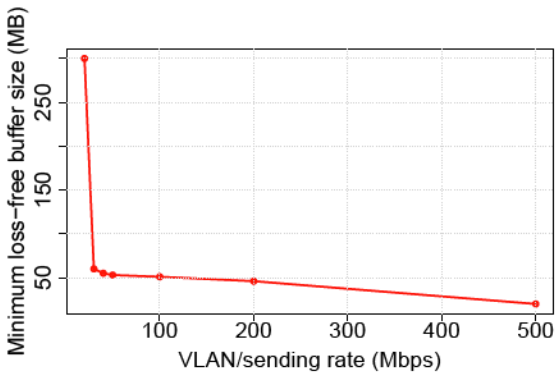



Fig. 3: Minimum loss-free buffer size required for various VLAN rates [4]

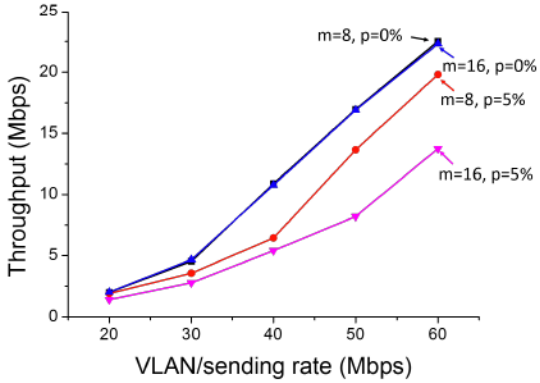



Fig. 4: Feed: NGRID 2018-11-16 22:10-23:10; $RTT = 0.1$ms

*3) Impact of number of receivers and packet loss rate:* Of the five input parameters listed in Section V-B, the feed and RTT are kept unchanged. The feed used in this experiment is NGRID, collected on 2018-11-16, from 22:10-23:10, and the RTT is 0.1 ms. Multiple values were used for the other three input parameters, VLAN/sending rate $r$, number of receivers $m$, and the artificially injected packet loss rate $p_j$.

Fig. 4 show that when packet loss rate was 0%, there was not much difference in throughput when the number of receivers was increased from 8 to 16. In these logs, we found that no packets were lost, and therefore receivers did not require FMTP block retransmissions. Therefore, the throughput was almost the same for both settings of $m$.

Fig. 4 shows that when the artificially injected packet loss rate was 5%, the gap in throughput widened at higher values of VLAN/sending rate $r$. This is because at low rates, the main determinant of throughput is waiting time in the sender buffer, while as rates increase, the FMTP block retransmission delays matter more.

*4) Impact of VLAN/sending rate:* To gain a better understanding of the impact of VLAN/sending rate on latency, Fig. 5 shows throughput and degradation for the NGRID feedtype, where degradation is defined as the ratio of the VLAN rate

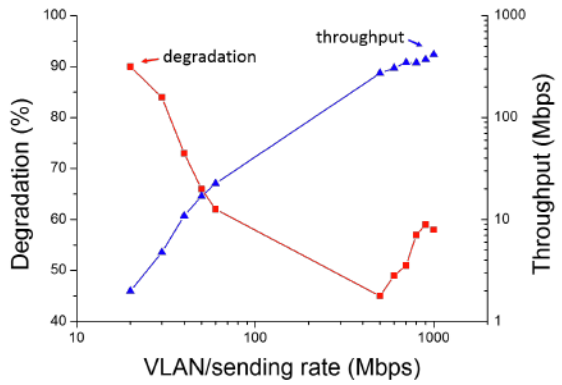



Fig. 5: Impact of VLAN/sending rate; $m = 8$, $RTT = 0.1$ms, $p = 0$

minus throughput to the VLAN rate. Degradation decreases from 90% to 45% with increasing VLAN rate until 500 Mbps, and then increases to around 58%. When the VLAN rate is low, files are queued in the sender buffer waiting for transmission. This waiting time increases total latency, which reduces throughput.

We illustrate the impact of sender buffer waiting delays, and then explain the higher degradation at 1000 Mbps.

TABLE VI: Evidence of sender buffer buildup, $RTT = 0.1$ms, $p = 0$, $r = 20$ Mbps

| File (No.) | Arrival time at sender (HHMMSS.ms) | Reception time at a receiver (HHMMSS.ms) | Size (B) | Latency (ms) | Throughput (Mbps) |
|---|---|---|---|---|---|
| 0 | 004800.721 | 004800.935 | 516824 | 213.77 | 19.34 |
| 1 | 004800.861 | 004801.099 | 394075 | 238.05 | 13.24 |

Table VI shows the arrival time of two files at the sender, which is the time instant at which the file was inserted into the product queue at the sending LDM7 server, reception-time at a receiver, which is the time instant at which the product was received by FMTP and passed up to the receiving LDM7 server, and size. Latency is the difference between the two timestamps. Throughput values were computed by dividing product size by latency (both values are less than the VLAN/sending rate of 20 Mbps). No retransmissions were required for either file.

File 0 experienced no waiting time in the sender buffer. But the slightly lower throughput is due to other components, such as processing delays, one-way propagation delay, and switch/router queueing delays. The transmission delay for file 0, including FMTP, UDP and IP headers, is 212 ms. If we assume that file 0 processing delays in the sender are in $\mu$s and hence negligible, the whole file should have been transmitted on to the wire by 004800.933. File 1 arrival time is 004800.861, which means file 1 had to wait in the sender buffer for 72 ms. Transmission delay of file 1 is 162 ms. Together these two delays lower throughput to 13.8 Mbps. The remaining part of the drop to 13.24 Mbps is explained by the other delay components. This example illustrates that waiting time in the sender buffer plays an important role in degrading throughput when the VLAN/sending rate is small.

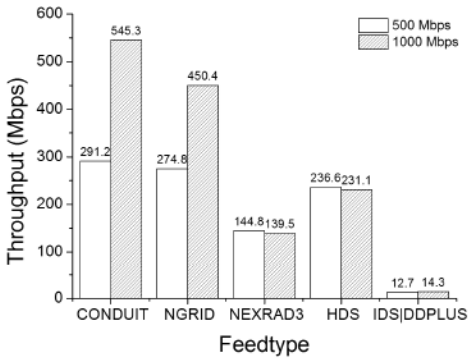
TABLE VII: Statistics for total variable delay for one-hour NGRID products as a function of $r$; $m = 8$, $RTT = 0.1$ms, $p = 0$

| **Time** (ms) | 20 Mbps | 60 Mbps | 500 Mbps | 1000 Mbps |
|---|---|---|---|---|
| Min | -0.58 | 0.11 | -0.07 | 0.05 |
| 1Q | 5.01s | 2.84 | 0.16 | 0.39 |
| Median | 17.69s | 16.12 | 0.31 | 0.55 |
| 3Q | 35.47s | 79.38 | 0.85 | 0.84 |
| Max. | 80.52s | 2286.21 | 265.98 | 610.54 |
| Mean | 23.33s | 61.04 | 2.41 | 1.60 |

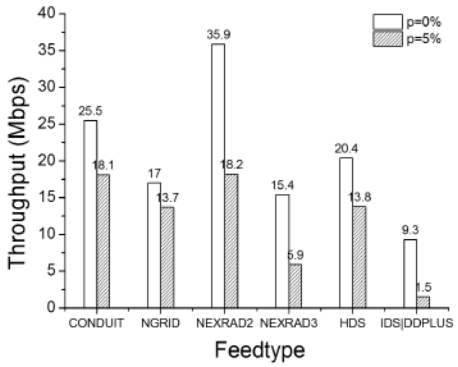
While we cannot provide measurements for just the sender-buffer waiting times, we can compute latency minus the sum of transmission delay and one-way propagation delay to determine the sum of the following components: sender-buffer waiting delays, processing delays, switch/router buffer queueing delays and retransmission delays. We refer to this sum as total variable delay. Table VII shows statistics for this total variable delay for different VLAN/sending rate settings. The negative minimum values occurs because NTP is not precise enough in synchronizing sender and receiver clocks. Delays are significant, on the order of seconds, when $r$ is 20 Mbps. These delays decrease with increasing $r$. Waiting time in sender buffer is the key delay determined by rate. With $r$ set to 500 Mbps, waiting times drop considerably.

Differences in values of the total variable delay between the 500 Mbps and 1 Gbps are likely due to retransmission delays. We ran the Linux utility `tcpdump` at one receiver to capture block retransmissions. We found that several blocks required retransmissions when the VLAN/sending rate was 1 Gbps, while no block retransmissions were required when the rate setting was 20 Mbps.

*5) Impact of feedtype:* This experiment considered the impact of the feed type on throughput. Fig. 6a shows that CONDUIT and NGRID feedtypes achieve higher throughput than the other three feedtypes. A comparison of the size distribution of products for these feedtypes provides an explanation. For example, we considered measurements obtained from one-hour feeds of the six feedtypes. The median, mean, and maximum product size was 52.6, 127.8 and 1357.2 KiB for CONDUIT, 31.8 KiB, 149.1 KiB and 15.6 MiB for NGRID, 7.8 KiB, 29.2 KiB and 17.8 MiB for HDS, 36.2, 50.4 and 778.3 KiB for NEXRAD2, 5.8, 12.2 and 152.2 KiB for NEXRAD3, and 0.2, 1.3 and 215.1 KiB for IDS|DDLUS. The total number of products delivered in that hour were 23610 for CONDUIT, 15409 for NGRID, 8661 for HDS, 10498 for NEXRAD2, 13591 for NEXRAD3 and 7334 for IDS|DDLUS. These numbers suggest that products are generally larger for CONDUIT and NGRID. The difference between throughput and VLAN/sending rate is primarily influenced by the contribution of transmission delay to total latency: the higher the contribution, the smaller the difference. At both 500 Mbps and 1 Gbps, sender-buffering waiting delay is negligible. Also, with no artificially injected packet losses, there will be few retransmissions, if any. With NEXRAD3 products, transmission delay, switch/router buffer queueing




(a) At high rates (500 and 1000 Mbps)




(b) At lower rate (50 Mbps)

Fig. 6: Input parameters: $m = 8$, $p = 0$, $RTT = 0.1$ms

delays, processing delays, and propagation delay will have equal weights. For example, when the VLAN rate $r$ is 500 Mbps using the median file size, the transmission delay is 0.096 ms, which is 96 $\mu$s, one-way propagation delay is 50 $\mu$s, and processing and switch/router buffer queueing delays are likely to also be in $\mu$s. This explains why NEXRAD3 throughput is much lower than the VLAN/sending rate $r$. We also observe that the NGRID feedtype has fewer and larger files. In this case, sender buffer waiting delays could occur with NGRID, leading to higher latency and hence lower throughput relative to CONDUIT.

Fig. 6b offers insights into the effects of sender-buffer waiting delays since the VLAN/sending rate is only 50 Mbps. In this graph, we observe that NEXRAD2 performs the best, with its throughput reaching 35.9 Mbps. NEXRAD2 has similar size characteristics as CONDUIT. However, a study of the product inter-arrival time characteristics showed that in the one-hour CONDUIT feed, more than 50% of the inter-arrival times were less than 1 ms, while the first quartile of inter-arrival times with NEXRAD2 was 16 ms. This observation explains that there is a smaller probability of sender buffer buildup for the NEXRAD2 feed type, and hence waiting times are less with NEXRAD2 than with CONDUIT. Correspondingly, throughput is higher for NEXRAD2. With an artificially injected 5% packet loss rate, the impact of feedtype on throughput decrease depends on the relative weight of retransmission delay to overall latency. The smaller the files, the more significant the impact of retransmission delay since more files are likely to require block retransmissions. Conversely, NGRID has the largest files, and correspondingly the smallest drop.

*6) Impact of RTT:* We ran a single-receiver experiment in which we emulated paths with multiple RTT values: 0.1 ms, 10 ms, 20 ms, 30ms, 40ms, 50 ms, and 100 ms. Two sets of experiments were executed with rate $r$ set to 20 Mbps and 500 Mbps. The sender-buffer size was set to 600 MB.

Fig. 7 shows the results. When rate $r$ is low, sender-buffer waiting delays are the dominant component of latency and hence determine throughput, not RTT. But when the base rate is higher, RTT plays a significant role in determining

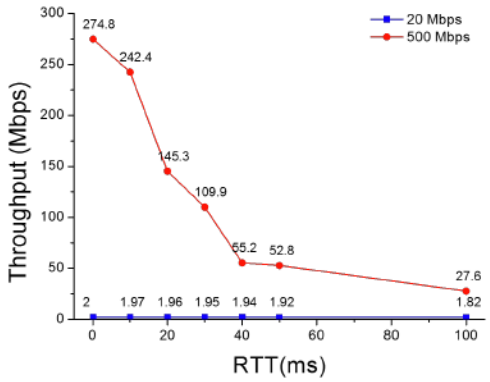



Fig. 7: Impact of RTT on throughput

throughput. As noted earlier, transmission delays are small for the IDD feeds, since file sizes are small. The remaining components, processing delays, switch/router buffer queueing delays, and retransmission delays are small in the IDD high-performance, low-loss environment.

## VI. Related Work

Multiple solutions have been proposed to leverage SDN capabilities for network multicast. A Software Defined Network Aware Pub/Sub (SAPS) solution [14] uses a hybrid approach with both Application Layer Multicast (ALM) and OpenFlow based multicast (OFM), unlike our solution, which uses only OFM. Huang et al. [15] introduced the Branch-aware Steiner Tree (BST), a multicast tree for SDN, which is used to minimize the total number of edges and branch nodes in the tree. Iyer et al. [16] presented Avalanche, an SDN-based system, which enables multicasting in commodity switches used for data centers. Shen et al. [17] proposed the Recover-Aware Steiner Tree (RST), a reliable multicast tree for SDN, and presented an approximation algorithm to solve the problem of finding an RST with low tree cost and recovery cost. Ren et al. [18] characterized and addressed the Delay-guaranteed Minimum Cost Forest (D-MCF) problem to ensure the quality of service (QoS) of multicast applications. However, these OpenFlow-based multicast advances focused on the control-plane problem of finding the best multicast topologies, and

not on the data-plane problem of using multicast trees for data dissemination. The focus of our work is on data-plane aspects.

Other solutions have investigated methods to make multicasting reliable using techniques, such as coding or using acknowledgements. These include MCTCP [19] and ECast [20]. Multicast TCP (MCTCP) retains the main TCP methods, even positive acknowledgments (the ACK implosion problem of the sender having to deal with ACKs from multiple receivers is handled by restricting its use to small groups), but adds a control-plane component in which the SDN controller monitors link utilization and reconfigures the multicast tree to reduce congestion. In contrast, our FMTP solution uses negative acknowledgments, which is feasible because in-sequence delivery is guaranteed on L2 multipoint VLANs. ECast is an OpenFlow-enabled elastic loss recovery solution, in which information on lost packets is obtained from receivers, and new trees are created to multicast retransmissions. Receivers with common missed packets will belong to the same elastic-area multicast tree. In our WAN based multi-domain VLAN solution, the time to set up a new multipoint VLAN is much higher than the low latencies required in the data plane. Therefore, this ECast solution will not work in our context.

Solutions based on P2P [21] will incur higher latency than our network multicast solution, but hybrids are possible for a receiver to obtain missed blocks from a nearby receiver, though all receivers should then also have sender capabilities.

## VII. Conclusions

This work demonstrated that it is feasible to deploy a network multicast solution leveraging Software Defined Networks (SDNs) with dynamic multipoint VLAN service and IP-routed service. We improved our prior Multicast-Push, Unicast-Pull (MPUP) cross-layer architecture to support reliable file-stream multicasting in two ways: (i) we evaluated a simpler solution of using a constant value for the sender retransmission timer, and found this to be a feasible solution for the meteorology feedtypes (file-streams) used in our study, and (ii) we improved the rate-selection algorithm by using measurable characteristics such as file-delivery latency. An experimental evaluation revealed interesting findings: (i) sender-buffer waiting time is a dominant delay when using low VLAN rates, and with bandwidth-borrowing between VLAN and IP-routed services, VLAN utilization is not an important consideration, and therefore higher VLAN rates should be used; (ii) increasing VLAN rate beyond a certain level offers no gains for file-streams with small files; (iii) in WAN applications with small-sized products, propagation delays can dominant total file latency; (iv) high packet-loss rates increase retransmission delay of missed blocks and since this delay depends on propagation delay, losses can cause a significant degradation of throughput in WANs; further, this degradation increases rapidly with the number of receivers in the multicast tree.

## VIII. Acknowledgment

We thank NSF for grant 1659174 and Michele Co for her contributions. Shuoshuo Chen's work was done when he was affiliated with University of Virginia; Google is his current affiliation. Results presented in this paper were obtained using the Chameleon testbed supported by the NSF.

## References

[1] Internet Data Distribution. [Online]. Available: http://www.unidata.ucar.edu/software/idd/

[2] B. Fenner and D. Meyer, "Multicast Source Discovery Protocol (MSDP)," IETF, RFC 3618, Oct. 2003. [Online]. Available: http://tools.ietf.org/rfc/rfc3618.txt

[3] S. Chen, X. Ji, M. Veeraraghavan, S. Emmerson, J. Slezak, and S. G. Decker, "A cross-layer Multicast-Push Unicast-Pull (MPUP) architecture for reliable file-stream distribution," in *2016 IEEE 40th COMPSAC*, vol. 1, June 2016, pp. 535–544.

[4] S. Chen, "A cross-layer architecture and protocols for reliable file-stream distribution," Master's thesis, University of Virginia, 5 2016.

[5] M. McGinley, H. Bhuiyan, T. Li, and M. Veeraraghavan, "An in-depth cross-layer experimental study of transport protocols over circuits," in *Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on*, Aug 2010, pp. 1–6.

[6] Local Data Manager. [Online]. Available: http://www.unidata.ucar.edu/software/ldm/

[7] K. Keahey, P. Riteau, D. Stanzione, T. Cockerill, J. Mambretti, P. Rad, and P. Ruth, "Chameleon: a scalable production testbed for computer science research," in *Contemporary High Performance Computing: From Petascale toward Exascale*, 1st ed., ser. Chapman & Hall/CRC Computational Science, J. Vetter, Ed. Boca Raton, FL: CRC Press, 2018, vol. 3, ch. 5.

[8] Internet2 Advanced Layer 2 Services (AL2S). http://www.internet2.edu.

[9] J. Li, M. Veeraraghavan, S. Emmerson, and R. Russell, "File multicast transport protocol (FMTP)," in *IEEE CCGrid, SCRAMBL workshop*, May 2015, pp. 1037–1046.

[10] A. Mudambi, X. Zheng, and M. Veeraraghavan, "A transport protocol for dedicated end-to-end circuits," in *IEEE ICC*, vol. 1, June 2006, pp. 18–23.

[11] Open Exchange Software Suite (OESS). http://globalnoc.iu.edu/sdn/oess.html.

[12] X. Ji, Y. Liang, M. Veeraraghavan, and S. Emmerson, "File-stream distribution application on software-defined networks (SDN)," in *Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual*, vol. 2, July 2015, pp. 377–386.

[13] LDM-6 feed types. https://www.unidata.ucar.edu/software/ldm/ldm-current/basics/feedtypes/.

[14] T. Akiyama, Y. Kawai, Y. Teranishi, R. Banno, and K. Iida, "SAPS: Software defined network aware pub/sub – a design of the hybrid architecture utilizing distributed and centralized multicast," in *Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual*, vol. 2, July 2015, pp. 361–366.

[15] L.-H. Huang, H.-J. Hung, C.-C. Lin, and D.-N. Yang, "Scalable Steiner tree for multicast communications in software-defined networking," *arXiv preprint arXiv:1404.3454*, 2014.

[16] A. Iyer, P. Kumar, and V. Mann, "Avalanche: Data center multicast using software defined networking," in *2014 Sixth Intl. Conf. on Comm. Sys. and Nets. (COMSNETS)*. IEEE, 2014, pp. 1–8.

[17] S.-H. Shen, L.-H. Huang, D.-N. Yang, and W.-T. Chen, "Reliable multicast routing for software-defined networks," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 181–189.

[18] B. Ren, G. Cheng, and D. Guo, "Minimum-cost forest for uncertain multicast with delay constraints," *Tsinghua Science and Technology*, vol. 24, no. 2, pp. 157–159, 2019.

[19] T. Zhu, F. Wang, Y. Hua, D. Feng, Y. Wan, Q. Shi, and Y. Xie, "MCTCP: Congestion-aware and robust multicast TCP in software-defined networks," in *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*. IEEE, 2016, pp. 1–10.

[20] X. Zhang, M. Yang, L. Wang, and M. Sun, "An OpenFlow-enabled elastic loss recovery solution for reliable multicast," *IEEE Systems Journal*, vol. 12, no. 2, pp. 1945–1956, 2018.

[21] A. Sampaio and P. Sousa, "An adaptable and ISP-friendly multicast overlay network," *Peer-to-Peer Net. and Apps.*, pp. 1–21, 2018.