OpenSession: SDN-based Cross-layer Multi-stream Management Protocol for 3D Teleimmersion

Ahsan Arefin Raoul Rivas Rehana Tabassum Klara Nahrstedt
University of Illinois at Urbana-Champaign, Urbana, IL
{marefin2, trivas, tabassu2, klara}@illinois.edu

***Abstract*—Video conferencing applications pose fundamentally different service requirements than traditional data traffic on the Internet. Strong real-time *interactivity* is very important among participants unlike video streaming in VoD applications. Requirements are even more stringent in multi-stream and multi-site teleimmersive applications due to strong dependencies across geographically distributed streams. In this paper, we propose *OpenSession*, a cross-layer session-network control protocol for multi-stream multi-site 3D teleimmersion (3DTI) that improves interactivity, resource utilization and scalability. OpenSession decouples application layer data and control plane functionalities, and partially offloads the data plane functionalities to network layer switches. To control network layer switches during the session run-time, OpenSession leverages support from Software Defined Networking (e.g., OpenFlow). Through extensive evaluation with multi-stream 3D teleimmersion among four distributed sites and PlanetLab-based larger 3DTI setup, we show that OpenSession improves 3DTI interactivity and resource usage at the data plane. Furthermore, OpenSession keeps data plane robust against host failures and frequent route updates.**

I. Introduction

We have seen a surge of interest in telepresence video collaborative technologies over the past few years. Though the initial focus was on video conferencing application [9][27], emerging is the 3D teleimmersion (3DTI) technology that expands the horizon by supporting full-body interaction of physical activities in virtual environments. Applications have been found in rehabilitation, collaborative dancing, and online gaming (e.g., [17], [39]) in addition to video conferencing [19].

3DTI consists of a real-time correlated multi-stream environment, where distributed participants interact with each other in a shared virtual space by fusing streams from multiple 3D cameras, microphones and other sensors. Despite great potential, today's multi-stream 3DTI still faces significant challenges due to the high interactivity requirement and large demand on processing and network resources. Furthermore, it introduces a new notion of user interaction, where participants can frequently change their view orientations in the virtual space. Frequent view changes require frequent updates of multi-stream content distribution. Due to this dynamic user behavior, even with notable improvements in application layer [6][34], today's 3DTI struggles with heavy temporal and spatial overheads and multi-stream management complexities.

With the advancement of Software Defined Networking (SDN) research over the past few years, network components have become accessible and controllable from application layer [32]. Applications can use this flexibility to offload certain data plane functionalities to the network layer for the purpose of improving data plane complexity and efficiency. We have seen the leverage of SDN (e.g., OpenFlow [22]) in different applications such as selecting servers for load balancing [13], managing consistency in network migration [16], and finding peers for video streaming [10]. The natural question for us is "can we use OpenFlow network components to reduce data plane complexities and improve multi-stream flow management in 3DTI". To address this question, we develop OpenSession, a session-network cross-layer management control protocol for managing multi-stream flows in 3DTI.

978-1-4799-1270-4/13/$31.00 ©2013 IEEE

OpenSession introduces a split forwarding architecture at the application data plane. It *partially* offloads stream multicast functionality of the application to the SDN switches. If multiple participants request the same streams, the source participant sends only single copies of the local streams to the SDN network switch, which handles multi-stream forwarding to all remote destinations on-behalf-of the application. The splitting of data plane reduces processing load at the application, and network bandwidth usage at the network. Offloading multi-stream forwarding to the SDN network component also improves end-to-end delay in multicast-based streaming because forwarding of streams is done from the network layer of the SDN switches instead of from the application layer of the end-hosts at each multicast hop.

However, splitting the data plane introduces several challenges. First, the application layer multi-stream semantics are lost in the network layer data plane. Without keeping this semantics, the SDN switches cannot forward streams according to the overlay topology. To solve the semantic mapping problem, OpenSession develops a *packet differentiation* mechanism that assists the switches to differentiate data packets across multiple streams. Second, SDN provides a fixed interface (OpenFlow) to configure the network layer data plane. To work with this standard interface, OpenSession develops a *mapping algorithm* to map the multi-stream overlay topology from the application layer to the network layer. Third, 3DTI requires frequent updates on the data plane configuration due to frequent view changes. To allow fast and seamless view changes, the updates on the data plane should be performed consistently across all participants so that the view changes do not introduce any interference on streaming. To maintain a global consistency while updating the distributed data planes, OpenSession develops a *coordinated route update protocol*.

We implement OpenSession using OpenFlow switches [22] and Floodlight controller [12]. To prove the benefits of OpenSession, we run experiments with real 3DTI application setup among four distributed participants (within small geographical

region). To show the impact of larger scale on Internet, we also run distributed 3DTI sessions using PlanetLab [26] nodes. Our results show that OpenSession improves multi-stream 3D streaming performance. It reduces end-to-end delay in proportion to the round-trip time (RTT) of local networks (e.g., a campus network where the application host resides) at each multicast hop. It also improves bandwidth load within the local network and processing load inside the application host in proportion to the number of streams in a 3DTI session.

Though we focus on interactive 3DTI throughout this paper, OpenSession does not provide any design limitation that bounds other multi-party tele-conferencing applications to take advantage of it. Our key contributions are: 1) we design and implement OpenSession, a session-network cross-layer management control plane for multi-stream 3DTI using SDN support (§IV, 2) we demonstrate (§VI) that OpenSession improves 3DTI interactivity in proportion to RTT of the local network, and reduces bandwidth load within the local network and processing load inside the application host in proportion to the number of streams, and 3) we show (§VI and §VII) that OpenSession makes the data plane resilient against frequent view changes, route updates and host failures.

II. Background

A. 3D Teleimmersion

3DTI application model: 3DTI system is a distributed platform connecting multiple remote sites containing a large number of input and output devices (as opposed to traditional video conferencing) and creating a virtual shared space for remote interactions as shown in Figure 1. The system contains three architectural tiers. In **capturing tier**, multiple capturing devices such as multiple 3D cameras, microphones, and other haptic sensors capture cyber-physical multi-modal information of each participant at his/her physical site. The captured streams are sent to a local *rendezvous point*, called gateway (A, B or C in Figure 1), which is responsible for data dissemination. The **dissemination tier** consists of overlay network of gateways multiplexing streams to and from each site. In **rendering tier**, streams are synchronously played.

3DTI data model: Each 3DTI site (or participant) hosts multiple 3D stereo cameras; each captures a local scene from various orientations and constructs a color-plus-depth 3D stream (S). The selection of local streams to transmit to a remote participant is calculated based on local streams' orientations and remote participant's view (v) demand. For example, if the participant is currently looking at the front of a 3D scene, streams generated by the back cameras are less *important* and can be dropped when bandwidth is limited. Views are represented by a set of remote streams in order of their importance in the view. A view v by the participant at Site-C (in Figure 1) can be represented as v=[S_1^A,S_1^B, $\cdots$, S_n^A, S_n^B] for n streams per remote participant. The higher number of 3D streams a site receives (within the available bandwidth), the better quality of 3D virtual environment it creates.

Note that individual streams are not merged at the source, rather transmitted separately, because it is hard to merge depth maps for all possible view points and merging texture map is computationally expensive for real-time applications in terms of latency. Moreover, the size of the merged stream becomes very large, which invalidates the notion of dropping less important streams in case of bandwidth limitation [37]. In addition, 3DTI introduces a unique demand, called *view change*, where participants can change their views during a running session. A view v_i is different from view v_j, i.e., $v_i \neq v_j$ if $\exists S_1 \in v_i \wedge S_2 \in v_j$ such that $S_1 \neq S_2$.

3DTI properties and assumptions: 3DTI data streams are transmitted using UDP protocol, whereas the management control traffic uses TCP. Mesh-based overlay multicast routing is used for efficient resource-aware construction of multi-stream distribution topology. However, the control traffic uses point-to-point connectivity due to its very low bandwidth requirement. Bandwidth adaptation (to and from 3DTI sites) is done by dropping less *important* (low priority) streams. The number of users in 3DTI is limited because 1) human attention space is intrinsically bounded [15], and 2) the number of participants that can be displayed in the virtual space is limited due to the pixel space limitation of the physical display [38].

B. Software Defined Networking

In Software Defined Networking, a logically centralized controller manages how the network switches forward packets in layer-2 and layer-3. The controller exchanges control messages with the switches using a standard protocol, such as OpenFlow [20]. We use the terms "SDN" and "OpenFlow" interchangeably in this paper. The control plane installs *forwarding rules* to forward packets in the switch data plane using a *flow table* with *match entry* and *action field*. Incoming data packets are matched against the match entry, and corresponding actions (e.g., forwarding, header modification) in the action field are performed on the matched packets.

III. Motivation and Challenges

A. Benefits

Consider the scenario shown in Figure 2(a). Site-A generates three streams (for simplicity, we only show streams generating from a single site) S_1^A, S_2^A and S_3^A. S_1^A and S_2^A are distributed to Site-B, Site-C, and Site-D during a running 3DTI session. S_3^A is dropped due to inbound bandwidth limitation. The local network shown in the figure can be a campus network, a department network, or a home network. In OpenSession, streams are forwarded from the network switches, connected within the local network.

Local bandwidth: OpenSession introduces a significant reduction of 3DTI bandwidth within the local network of participating sites. For example, in Figure 2(a), forwarding of streams S_1^A and S_2^A to Site-C and Site-D is done from the network switch of Site-B rather than from its application gateway. Therefore, the gateway of Site-B does not forward streams S_1^A and S_2^A from the application host, which saves the forwarding bandwidth of S_1^A and S_2^A within the local network of Site-B. It eventually saves the last mile bandwidth to and from the application host. Bandwidth saving also happens at the streaming source. Though Site-A needs to send stream S_1^A to both Site-B and Site-D, only one copy traverses the local network to the local OpenFlow switch, which performs stream replication and forwarding.

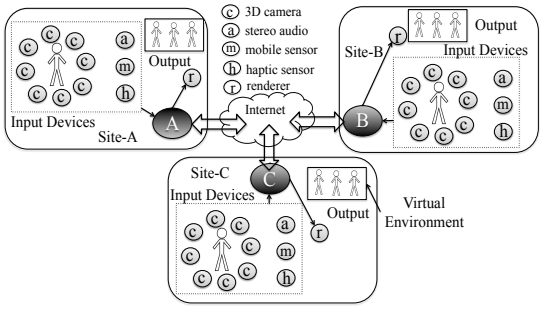


Fig. 1. 3DTI application model with three sites.

End-to-end delay: Since forwarding of stream from Site-B is done from the network layer switch (in Figure 2(a)), the end-to-end delays (EEDs) of streams S_2^A from Site-A to Site-C and Site-D are reduced by the round-trip time (RTT) of 3D frames in the local network (d_{net}) and through the protocol stack of the gateway machine (d_{app}) at Site-B. To measure the order of improvement in EED, we monitored our campus network for 3 days from January 15, 2013 to January 17, 2013. We used four gateway hosts (over wired and wireless links) at different locations within the campus, and measured point-to-point RTT among them for exchanging 3D frames. The average EED is about 4ms to 5ms, which is a significant latency in real-time communication considering a single multicast hop.

Processing load: Though Figure 2(a) considers only forwarding of two streams, multi-stream 3DTI constructs a much more complex distribution graph. We have experienced that with the increase of number of sites and the number of streams per site, the forwarding load (CPU load) of 3DTI application layer gateway increases linearly. Since, in OpenSession, the network layer is responsible for stream replication and forwarding, the application layer is not impacted by the increase of number of subscribers (i.e., participants) of a stream.

System resiliency: Application hosts fail and crash more frequently than underlying network components. As OpenSession separates multicast forwarding from the application layer, the failure of an application gateway or any crash of application due to a software bug at one site does not interfere the real-time collaborative experience among other participating sites. Failure of B does not affect S_1^A to Site-C in Figure 2(a). However, we skip this benefit from the current scope.

B. Challenges

Per-stream packet differentiation: To forward streams from the local OpenFlow switches according to multi-stream overlay distribution graphs, OpenSession must ensure application-aware differentiation of network packets at the switches. For example, stream S_2^A (in Figure 2(a)) from Site-B is forwarded to both Site-C and Site-D, whereas stream S_1^A is only forwarded to Site-C. Therefore, the OpenFlow switch at Site-B should be able to differentiate network packets of S_1^A and S_2^A. OpenFlow standard provides a fixed set of fields to match against incoming packets [22]. However, organizations usually allow limited open UDP ports for secured network. In our setup, we use the same port for multiple streams. Traditional source and destination address, and source and destination port based packet differentiation fails since multiple streams among the sites use the same network layer flow semantics. Therefore, to use the benefit of OpenFlow-based split data plane architecture, we need to ensure two properties in *packet differentiation*: 1) overridden fields are not modified anywhere in the transmission path, and 2) the design does not require any changes in network components or OpenFlow protocol.

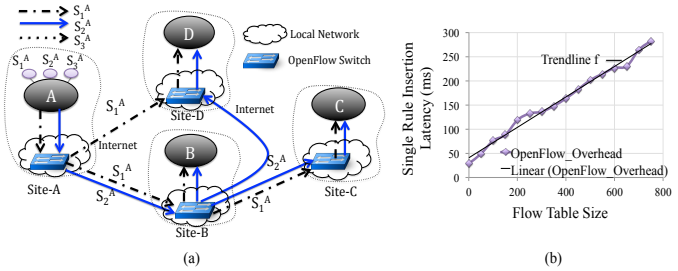


Fig. 2. (a) Example of a 3DTI session among 4 sites, (b) OpenFlow layer-3 rule insertion latency for different sizes of flow table.

Overlay to flow table mapping: Next, we need to ensure that flow table rules are consistently installed at all the participating OpenFlow switches to follow the stream *forwarding actions* according to the overlay. However, blind mapping of overlay to flow tables fails since it may interrupt other running flows in current or concurrent 3DTI sessions (e.g., via overriding or superseding available flow table entries). We need to ensure that the flow table size is optimized in the mapping process.

To understand the impact of switch flow table size on 3DTI session, we have conducted an isolated experiment with IBM G8264 OpenFlow switch.We measure a single rule insertion latency for different number of pre-installed rules into the switch. The result is shown in Figure 2(b). Wildcarded layer-3 rules are stored in a 'linear' table in the TCAM space of the switch [3]. As TCAM is expensive and small, every wildcarded layer-3 rule insertion performs an internal optimization of flow tables [14], which represents significant part of the rule insertion latency. This optimization latency increases with the increase of flow table size, which impacts session initiation and session update latency. Therefore, we need to ensure two properties while mapping the multi-stream overlay topology to switch flow tables: 1) modification of flow table does not interrupt on-going forwarding actions, and 2) size of flow table is as small as possible.

Seamless view change: View change introduces a new human behavior in the multi-stream 3DTI environment. If the participant at Site-B updates a view, which requests streams S_2^A and S_3^A instead of S_1^A and S_2^A, then stream S_1^A is no longer required by Site-B. However, if Site-A stops transmission of S_1^A to Site-B, the streaming path of S_1^A to Site-C is impacted and therefore, should be reconfigured. Sites who are impacted by remote view changes are called *victim sites*. Here, Site-C is the victim site due to the view change by Site-B. To ensure seamless streaming to Site-C due to this view change, the update of streaming path for S_1^A to Site-C should not interfere the immersive experience at Site-C in terms of packet loss.

Therefore, to ensure seamless view changes, the creation of new topology and the removal of old topology for streams should be performed concurrently. For example, if we update the path $A \rightsquigarrow D \rightsquigarrow C$ in Figure 2(a) before removing the path $A \rightsquigarrow B \rightsquigarrow C$ for S_1^A, then Site-C may get transiently overloaded with redundant S_1^A. Likewise, if we remove the path $A \rightsquigarrow B \rightsquigarrow C$ first and then update rules to create $A \rightsquigarrow D \rightsquigarrow C$, it introduces temporal disconnectivity of S_1^A to Site-C. What is required is a concurrent update of distributed flow tables that ensures two properties: no transient 1) bandwidth overloading, and 2) network disconnectivity.

IV. OpenSession Framework

OpenSession control plane contains three-layer session management hierarchy shown in Figure 3 and described below.

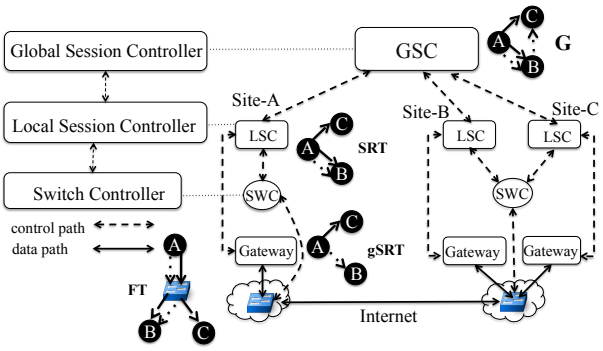


Fig. 3. OpenSession framework.

Global session controller: GSC represents a global control plane, which is an important design component for optimized content distribution [18]. GSC periodically (every few seconds) monitors participating sites (such as streaming rate, 3D construction time) and their underlying infrastructure (such as available inbound and outbound bandwidth and end-to-end delay). During session initiation and session update (such as view change), GSC constructs an optimized multi-stream overlay content distribution topology using *global view* of the sites (usually 5 to 10 [5]) and their underlying resources. Once the multi-stream topology is computed, the overlay routing related to each participating site is sent to the corresponding LSC. The information is represented by a session routing table or SRT (§V-A1). In Figure 3, we show an example graph corresponding to the SRT of site-A for a multi-stream overlay G computed at the GSC (streams from Site-A are only shown).

Local session controller: LSC is a lightweight (in terms of processing overhead) session layer. The main responsibility of LSC is to update session routing table (called gSRT) at the local gateway (i.e., at the application data plane) using the SRT received from GSC (§V-A2). Gateways use gSRT for the data plane forwarding of local streams. gSRT ensures that single copies of local streams are delivered out of the gateway. Figure 3 shows an example of gSRT at the gateway of Site-A. LSC forwards SRT to local SWC to map it to the switch flow table (FT). LSC is also responsible for monitoring local devices and resources, and sends them to GSC.

Switch controller: SWC directly communicates with the OpenFlow switches connected in the local network to install forwarding actions defined by SRT (§V-A3). An example of forwarding actions in FT related to the SRT and gSRT of the previous example is shown in Figure 3. SWC is also responsible for maintaining consistency and size optimization in FT. We run SWC as a module of the OpenFlow controller.

Data plane: The data plane is separated from the management control plane and divided between the application gateway and the OpenFlow switch. The gateways schedule local streams to multicast remotely based on the connectivity graph defined by gSRT. The OpenFlow switches multicast local and remote streams based on the forwarding actions defined in FT.

V. OpenSession Protocol and Functionality

A 3DTI Session is initiated by sending a session initiation request to the GSC, which contains a list of sites, a list of local devices (streams) at each site, and a list of desired views for all participants. Designing an efficient content distribution overlay that can combine different notions of quality (e.g., startup delay, frame rate) and user experience is an open challenge that is outside the current scope. Here we focus on OpenSession protocols for: (1) mappings of multi-stream content distribution topology to application and network layer data planes, and (2) handling of seamless changes in the distribution topology during session run-time.

A. Multi-stream Overlay Mapping

In this section, we will discuss, how the multi-stream overlay is mapped into SRT (§V-A1), gSRT (§V-A2) and finally to OpenFlow FT (§V-A3).

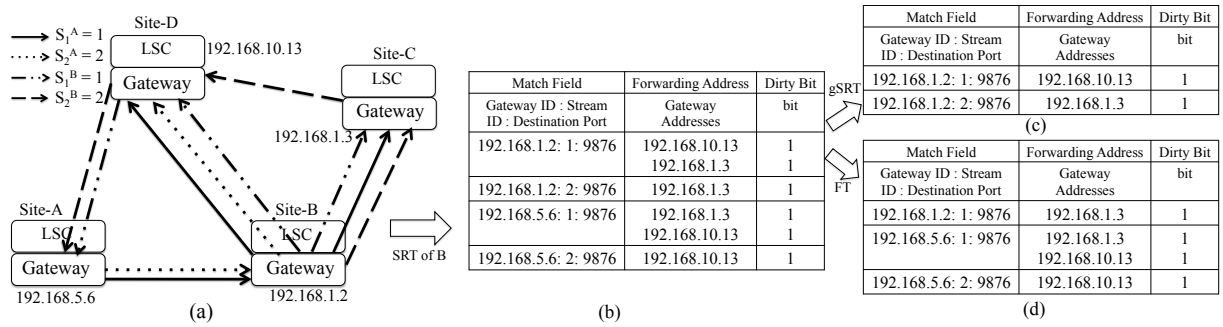
1) Overlay Topology to SRT Mapping: Once the multi-stream overlay topology is computed, it is mapped to a session routing table (SRT) for each participating site. The mapping of overlay to SRT aims to associate forwarding actions with each stream. SRT contains three fields: 1) **Match field** (SRT_{match}) that contains a stream ID, a UDP destination port and the gateway IP address of the stream originating site, 2) **Forwarding action** (SRT_{action}) that contains a list of gateway IP addresses to forward matching streams to, and 3) **Dirty bit** (SRT_{dirty}) that indicates whether an entry has been modified or not. An example of SRT computation from a multi-stream overlay is shown in Figure 4. For clarity, we only show two streams originating from Site-A and Site-B. The overlay topology computed by GSC is shown in Figure 4(a). Figure 4(b) shows the corresponding SRT of Site-B.

2) SRT to gSRT Mapping: The purpose of mapping SRT to gSRT is to define the forwarding responsibility to the application layer data plane at the gateway. Each gateway sends only single copies of the local streams towards the local OpenFlow switch, though multiple sites may request the same streams. Streaming towards multiple sites are then done from the OpenFlow switch via address modification (destination IP and MAC addresses) in the packet header. Therefore, the mapping from SRT to gSRT is done only for local streams to ensure that the gateway sends only one copy. In Figure 4(b), Site-B sends stream S_1^B to both Site-C and Site-D. However, the LSC only maps the streaming path to Site-D in gSRT at Site-B (application data plane). The streaming to Site-C is done from the network layer data plane. Figure 4(c) shows the entries of SRT that are mapped to gSRT.

3) SRT to FT Mapping: Finally, the mapping of SRT to FT is performed by SWC in three steps below:

Prune SRT entry: We only map dirty SRT entries. Moreover, we prune the SRT entries (associated to the local streams), which are already assigned to the application layer data plane (i.e., to gSRT). For example, in Figure 4(b), the SRT entry to forward S_1^B (with Match Field $192.168.1.2:1:9876$) to Site-D is pruned and not mapped to FT at Site-B. However, the SRT entry to forward S_1^B to Site-C is not pruned.

Update match field: After pruning SRT, we map SRT match field (SRT_{match}) to FT match field (FT_{match}). The OpenFlow



Match Field	Forwarding Address	Dirty Bit
Gateway ID : Stream ID : Destination Port	Gateway Addresses	bit
192.168.1.2: 1: 9876	192.168.10.13 192.168.1.3	1 1
192.168.1.2: 2: 9876	192.168.1.3	1
192.168.5.6: 1: 9876	192.168.1.3 192.168.10.13	1 1
192.168.5.6: 2: 9876	192.168.10.13	1

(b)

Match Field	Forwarding Address	Dirty Bit
Gateway ID : Stream ID : Destination Port	Gateway Addresses	bit
192.168.1.2: 1: 9876	192.168.10.13	1
192.168.1.2: 2: 9876	192.168.1.3	1

(c)

Match Field	Forwarding Address	Dirty Bit
Gateway ID : Stream ID : Destination Port	Gateway Addresses	bit
192.168.1.2: 1: 9876	192.168.1.3	1
192.168.5.6: 1: 9876	192.168.1.3 192.168.10.13	1 1
192.168.5.6: 2: 9876	192.168.10.13	1

(d)

Fig. 4. (a) Multi-stream overlay topology for S_1^A, S_2^A, S_1^B, and S_2^B, (b) the corresponding SRT of Site-B, (c) SRT entries that map to gSRT at Site-B (application data plane), and (d) SRT entries that map to OpenFlow FT serving at Site-B (network data plane). 9876 is UDP data communication port.

protocol allows direct mapping for source IP address (gw) and destination port (P_{dst}). However, the notion of application layer stream ID (S_{id}) is lost in the network layer data plane. To keep this semantic information in the network layer, we override the higher 5 bits of *IP ToS (Type of Service)* field. Overriding of IP ToS bits is very common in Internet applications as they are less frequently used in Internet [7]. When stream data is sent by the gateway, the 8bit IP ToS field of the UDP packets is set based on 5 bits of stream ID with zeros in lower 3 bits (i.e., we do not modify *ECN* bits used for congestion control). For example, if $S_1^A = 1$, the corresponding ToS field in the IP data packet is 00001000. Therefore, $\text{SRT}_{match} = [192.168.5.6.1 : 1 : 9876]$ maps to $\text{FT}_{match} = [192.168.5.6.1 : 00001000 : 9876]$.

Update action list: In addition to forwarding the incoming packets to the intended destination (based on packet header), the OpenFlow switches need to forward the same packets to other sites (based on $\text{SRT}_{actions}$) by replacing the destination IP and destination MAC addresses. If the destination IP is in the remote subnet, OpenSession uses MAC address of the gateway-router. Also, the physical port for each forwarding action is assigned based on the destination IP. Details are skipped due to brevity. In summary, in addition to forwarding the packets based on their original headers, we add three actions in FT_{action} field for each forwarding IP x in SRT_{action}: 1) set destination IP to x, 2) set destination MAC address to MAC_x, and 3) forward the packets to port Port_x. Here, Port_x is the physical port number at the switch that forwards data packets to destination x, and MAC_x is the MAC address required to forward data packets to destination x.

B. Seamless Session Adaptation

Session adaption (adding or dropping streams) is triggered when participants change views, or bandwidth resources at the participating sites change. In this section, we only consider view changes; other causes will naturally follow. The result of the view change is route update for one or more streams.

1) Seamless Route Update Problem: As we show in §III-B, a route update may cause a) **overloading** when redundant streams are transmitted to any participating site), and b) **disconnectivity** (when streaming of one or more streams becomes temporarily unavailable). To understand the phenomena clearly, we show a route update example with six participating sites in Figure 5(a)-(d) for both overloading and disconnectivity. With frequent view changes, interactive streaming performance at the victim sites due to transient overloading and disconnectivity becomes very poor.

2) Route Update Problem Formulation: The route update problem can be mapped to a *constraint-based graph transformation* problem. Initially, we will consider only single stream. Later (in §V-B5) we will extend our solution for multiple streams. Let us consider an existing overlay graph (G_{old}^S) and a new overlay graph (G_{new}^S) computed at GSC after a view change request. We represent each graph by a connectivity matrix M^S, where $M^S[i][j]$=1 if Site-i forwards stream S to Site-j, otherwise $M^S[i][j]$=0 (note that for the rest of this section we consider A as 0, B as 1 and so on while indexing the connectivity matrix). The transformation of graphs can be formulated by their connectivity matrices as follows:

Graph Transformation Problem: Transform M_{old}^S (connectivity matrix of G_{old}^S) to M_{new}^S (connectivity matrix of G_{new}^S) for each stream S, where$M_{old}^S \neq M_{new}^S$, so that the intermediate states of the connectivity matrix (M_{int}^S) in the transformation process do not violate following two constraints (N= number of sites): 1) $\sum_i M_{int}^S[i][j] \leq 1$, $\forall_{0 \leq j \leq N-1}$, and 2) $M_{int}^S[i][j] \neq 0$, if $M_{old}^S[i][j] = M_{new}^S[i][j] = 1$ $\forall_{0 \leq i,j \leq N-1}$. Constraint (1) ensures that there is no transient overloading of S, and (2) ensures that there is no transient disconnectivity.

3) Route Update Solution Overview: To solve seamless route update in OpenSession, we consider an additional packet header field (η) in the FT match entry. When we add new rules (for the purpose of transforming M_{old}^S to M_{new}^S) with a new value in η, the new rules are not matched against the incoming streams right away, because incoming streams do not contain the new value of η in their packet header. In Figure 6, we provide an example of how consideration of an additional match entry (η) in FT can solve the route update problem for S_1^A as shown in Figure 5. Below we describe.

Figure 6(a) shows the initial FT rules in Site-A before the route update. Since Site-D does not forward S_1^A, there is no FT entry at Site-D for stream S_1^A. While updating the route from Figure 5(a) to Figure 5(d), we first add a new FT rule at Site-D (in Figure 6(b)) with match field entry [A, S_1^A, P_{dst}, η'] (where η' is the new value of η) and an action to forward the matched stream to Site-C. Even though Site-D receives S_1^A from Site-A, the data packets are not forwarded to Site-C as incoming packets do not contain η' and are not matched with the new rule (i.e., no violation of constraints (1)).

To initiate the matching with new entries in FT (at Site-D), the OpenFlow switch at the source site (at Site-A) needs to update η with the new value η' in the outgoing packet header while streaming (S_1^A). We replace the FT rule at Site-A that forwards S_1^A to Site-B by a new FT rule with two actions: 1) forward S_1^A to Site-D, and 2) set η with η' in the outgoing packet header. When this rule is inserted at Site-A, the data

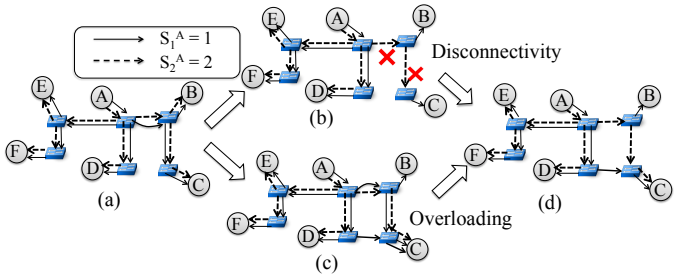


Fig. 5. Problem of route update: (a) old overlay graph, (b) disconnectivity at Site-C, (c) overloading to Site-C, and (d) new overlay graph.

packets of S_1^A towards Site-D contain η' in the packet header and are redirected to Site-C from Site-D. It also terminates the streaming to Site-B at the same time (Figure 6(c)). We can control the instant when we want to switch streaming paths concurrently for a stream by modifying FT rules at the streaming source (i.e., concurrently start streaming from Site-D to Site-C and stop streaming to Site-B and Site-C). Thus, OpenSession ensures that the victim sites are not impacted due to the view change (or any session adaptation) triggered by remote sites unless the updated path creates a large jitter, which depends on route computation algorithms.

Among other layer-3 and layer-4 fields, we consider source port (P_{src}) in the UDP header as η, because changing UDP source port for the outgoing packets does not violate any routing constraints. P_{src} is monotonically increased with every route update and reset to 1024 (0 to 1023 are specific purpose ports) when all running 3DTI sessions terminate. Note that gateways do not modify P_{src} in the packet header, rather the OpenFlow switch performs the replacement on the outgoing packets of the local streams.

If we update P_{src} in the FTs only at the sites impacted by a route update, soon there will be different values of P_{src} in the FTs across all sites with the same source IP, UDP destination port and IP ToS bits, which may create inconsistency in future FT update. Therefore, when a route update happens for stream S, all distributed FT rules that forward S are updated and start matching η' in the P_{src} match field.

Delete obsolete rules: When incoming streams are matched against the new rules, old rules becomes obsolete and are deleted automatically using the OpenFlow-defined timeout primitive. For a FT entry, if there are no matching packets for a continuous interval defined by `idle_timeout`, the rule will be deleted from the switch. We specify 30sec `idle_timeout`. OpenFlow also uses a `hard_timeout`, which removes the rule from FT after the timeout even matching flows exist. OpenSession uses infinite `hard_timeout`.

4) OpenSession Route Update Algorithm: Summarizing the above discussion, 1) we need to specify a new η each time a route update happens, and 2) FT entries in the remote sites need to be updated first before updating the FT entries at the source site (Figure 6(a)-(c)). Below we formally present the distributed route update algorithm at different controller levels.

Route update at GSC: GSC is responsible for initiating the route update. Depending on M_{new}^S and M_{old}^S, GSC classifies Site-i into four states (a site can be in multiple states):

1) **No-route** Site-i not forwarding S to any site before and after the route update ($\forall_j\ M_{old}^S[i][j]{=}0 \wedge M_{new}^S[i][j]{=}0$),
2) **Drop-route** Site-i drops forwarding S to any site after the route update ($\exists_j\ M_{old}^S[i][j]{=}1 \wedge M_{new}^S[i][j]{=}0$),

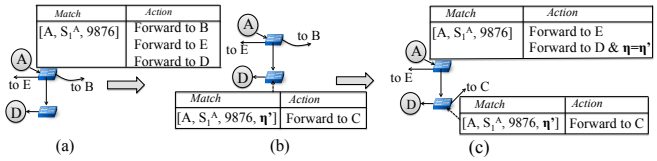


Fig. 6. (a)-(c) Sequence of FT updates for a seamless route update of S_1^A for the case shown in Figure 5. 9876 is the data communication port.

3) **Add-route** Site-i starts receiving S from any site after the route update ($\exists_j\ M_{old}^S[j][i]{=}0 \wedge M_{new}^S[j][i]{=}1$), and
4) **Keep-route** Site-i keeps receiving S from same site after the route update ($\exists_j\ M_{old}^S[j][i]{=}1 \wedge M_{new}^S[j][i]{=}1$).

If a site is in no-route state, no updates in SRT and FT are required and no messages are sent to the site's LSC. If a site is only in drop-route state, no SRT and FT updates are required (at non-origin sites). Obsolete entries (no match for η') are removed by the timeout. If a site is in keep-route state, though there is no update required in SRT, an update in FT is still required to assign $P_{src}{=}\eta'$ in the FT match entry (§V-B3). A route update request message Ω_{FT} is sent to the site's LSC. If a site is in add-route state, we need to update both FT and SRT to reflect the changes in the forwarding paths at Site-i. In this case, GSC sends a route update message Ω_{SRT_FT} to the site's LSC. If a site is in multiple states, the route update messages corresponding to the states are sent cumulatively to the site's LSC. Messages Ω_{FT} and Ω_{SRT_FT} are sent concurrently to all sites except at the source site of S. After all sites update their tables (SRT and FT), GSC sends a route update request message to the source site. For the source site, the selection of route update message is also made based on its state. However, even the source site is at drop-route state (i.e., streaming to one or more remote sites is dropped), we still need to send Ω_{SRT} message to update gSRT. Each route update message contains new SRT (SRT_{new}) with all dirty bits set to 1 (since we need to update P_{src} for all entries), and η'. To ensure the ordering of route update requests, GSC constructs a *depth first traversal (DFT)* of sites from M_{new}^S and sends route update requests to all sites concurrently except the last site in the order, which is the source site. Once route update requests are replied by all sites, then a route update message is sent to the source site to perform an atomic switch from M_{old}^S to M_{new}^S.

Route update at LSC: When Ω_{FT} or Ω_{SRT_FT} is received at LSC for remote streams, it simply forwards the message to SWC. However, for the local streams, LSC updates gSRT using SRT_{new} using the algorithm discussed in §V-A2 and then forwards the message to SWC.

Route update at SWC: When SWC receives route update message Ω_{FT} or Ω_{SRT_FT}, it maps the rules from SRT_{new} to FT with $P_{src}{=}\eta'$. Process for handling route update messages at the source site is the same, except P_{src} is not considered in the FT match entry field for S, and an additional action is added in the FT action list to replace P_{src} with η' in the outgoing packets of S. We modify our mapping algorithm in §V-A3 to incorporate P_{src}, however we skip it for brevity. To ensure that the rule has been added to the FT entry, a *barrier message* is sent to the switch [22]. Reply of the barrier message confirms that the previous command for rule insertion is completed successfully. Once FT entries are updated successfully, a confirmation is sent to the GSC. Figure 7 shows an example of OpenSession seamless route update process for the update of stream S_1^A.

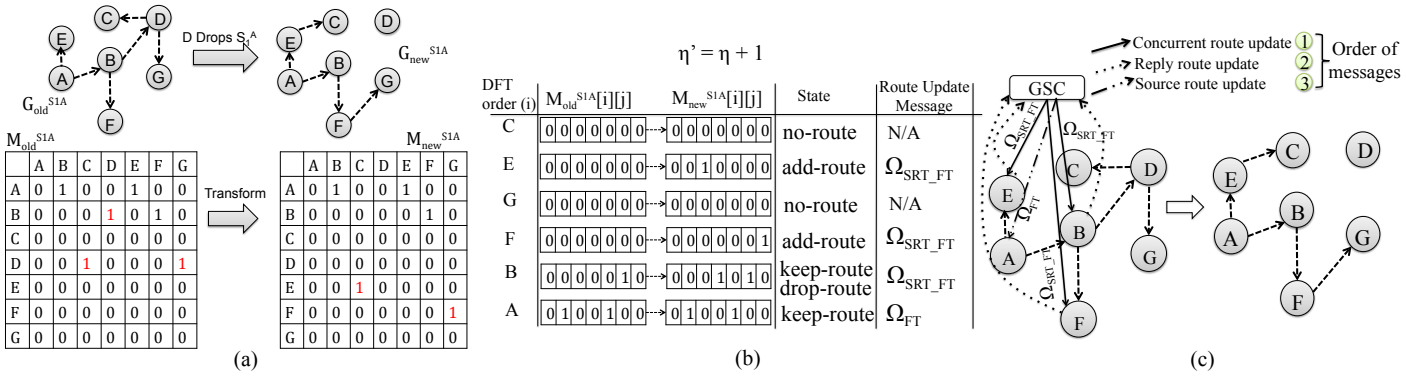


Fig. 7. Example of OpenSession route update process for a single stream: (a) route update due to Site-D drops stream S_1^A, (b) route update messages and computed state of the sites due to the route update, and (c) route update protocol showing the order of route update messages.

5) Multi-stream Route Update: During a view change, if routes of multiple streams are required to update in the new overlay, OpenSession performs source-based route update as shown in §V-B4 for all modified streaming paths in parallel. The route update messages (in Figure 7(d)) are sent cumulatively for all streams. However, the parallel route updates can be *dependent* if the updates of routes overlap at any site. For example, if Site-i drops (adds) forwarding (receiving) of S_1^A and adds (drops) forwarding (receiving) of S_1^B in the new overlay during session adaptation, the route updates for S_1^A and S_1^B become dependent as the updates overlap at Site-i. It is challenging to guarantee seamless session adaptation when route-update dependencies exist (though rare) among streams from different source sites. The source-based route update cannot guarantee atomic update of routes for S_1^A and S_1^B at Site-i as streams are originating from separate sources.

To ensure seamless route update at all sites, OpenSession considers a route update constraint at GSC for computing new multi-stream overlay: (**RUC**) if a site drops (adds) forwarding (receiving) of S_x^i, it does not add (drop) forwarding (receiving) of S_y^j in the same route update process for any x and y where $i \neq j$. Sites violating RUC may not experience seamless adaptation. Note that view changes do not violate RUC since during a view change $i=j$ (i.e., streams come from the same site). Also, when lower priority streams are dropped due to network congestion, the overlay modification does not usually violate RUC. RUC is a natural constraint, and in most of the cases, RUC remains valid. Details are skipped due to brevity.

VI. Evaluation with Real 3DTI Setup

A. Evaluation Setup

We setup 4 3DTI sites (in Figure 8(a)) in different types of networks. Site-A is in a *home network*, where the OpenFlow switch is placed next to the gateway. Site-B is in a *campus network*, where the OpenFlow switch is placed within the campus network. Site-C is in a *company network*, where the OpenFlow switch is placed closer to the company's network edge. Site-D is in a *department network*, where OpenFlow switch is placed closer to the department's network edge.

Each site contains 8 video streams capturing participants 3D stereoscopic image from 45^0 apart and constructs a mesh-based color-plus-depth 3D image. Instead of using physical cameras, we read streams from stored files, which already contain the 3D video of the participants for 1 hour session. Streams are read at the original frame rate. Average bandwidth

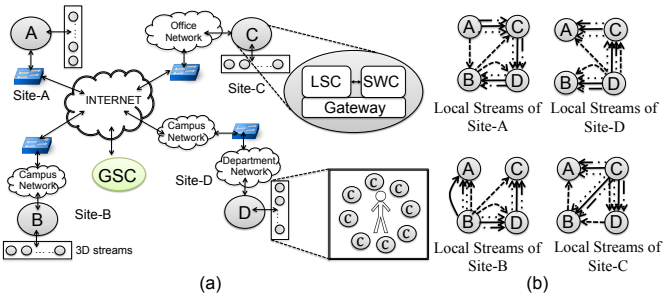


Fig. 8. (a) 3DTI setup with 4 sites, and (b) priority-driven multi-stream content distribution overlay graphs for local streams of each site.

of each 3D stream is about 1.5 to 2.1Mbps. Each application node (site) hosts a gateway to multiplex transmission of local streams. We place LSC on the same gateway host and implement SWC as a module of Floodlight [12] controller running on a separate host at each site. All control commands use TCP and data traffic uses UDP. We use OpenFlow software switches with 10GBps port capacity.

Initially, all sites request the same set of remote streams. Due to the notion of stream priority, each view demands only 4 streams from each remote participant (covering 180^0 space). The initial multi-stream overlay is constructed using the topology shown in Figure 8(b). However, during the session run-time, the overlay is updated to meet the view change demands. We compare the performance of OpenSession to a **"no OpenSession"** case, where the gateway performs the sole data plane functionality without using the network support.

B. Performance Metrics

To measure the application performance, we run the same 3DTI session with and without the OpenSession control plane. No view change is requested. Each performance value is measured, in percentage, with respect to the measurement collected without using OpenSession (i.e., no OpenSession).

Local bandwidth: OpenSession achieves a significant reduction in bandwidth within the local network. We divide the bandwidth usage into two parts. First, we plot how much bandwidth we use within the local network between the gateway and the OpenFlow switch at each site. We notice that (in Figure 9) Site-B, Site-C and Site-D achieve about 55% reduction in bandwidth for local streams in OpenSession. Second, we plot how much bandwidth we use within the local

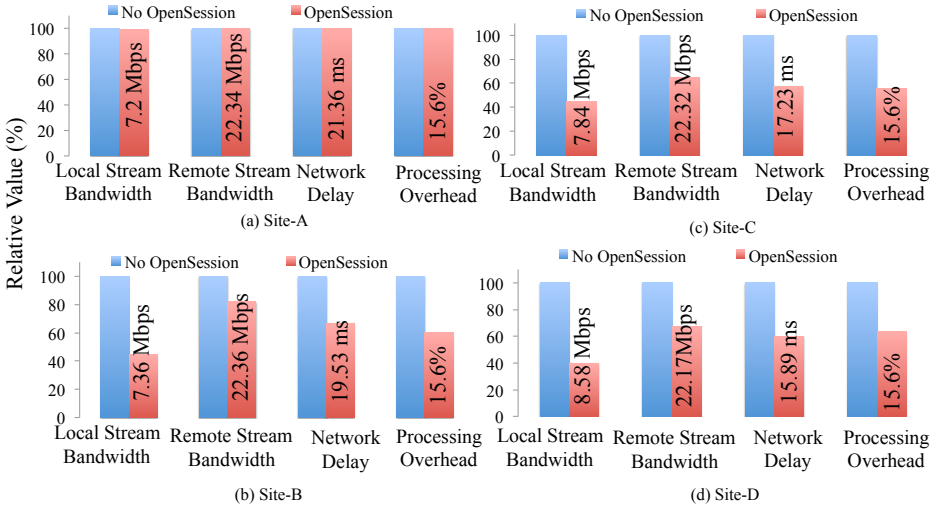


Fig. 9. (a)-(d) The relative value (with respect to the measurements without OpenSession) of performance metrics in a 3DTI session at different sites.

network for the remote streams. The OpenSession reduces about 35% bandwidth for the remote streams.

Processing Load: We measure processing load of the gateway, which includes stream receiving, scheduling and forwarding load of the application. As shown in Figure 9, OpenSession lowers stream scheduling and forwarding load at the application layer data plane by offloading part of the forwarding functionalities to the network layer data plane. On average, the OpenSession improves CPU overheads by 42% per site.

End-to-end delay: The gain in network transmission delay is achieved in OpenSession as streams do not traverse the local network for multicast-based forwarding at each multicast hop. We achieve about 45% gain (about 8ms) in the end-to-end network delay (will be higher for wireless end host) for transmitting 3D frames from the source to the destination site.

C. Impact of View Change

Views are changed according to the Zipf distribution of 20 pre-selected view orientations (22.5^0 apart). 100 view changes are requested from one site (Site-B) within 1 hour at equal interval. Routes for the impacted streams are updated considering constraints RUC (§V-B5). Valid peers (not violating RUC) to update streaming paths are selected randomly. No streams are dropped after session adaptation.

View change latency: View change latency is measured as the difference between the time when a view change is requested by Site-B and the time when Site-B receives new streams according to the view demand. In "no OpenSession" case, after computing the updated overlay at the GSC, the application data plane at the gateway is updated without considering the impact on victim sites. In OpenSession, the view change latency includes the steps shown in Figure 7. We plot CDF of view change latency in Figure 10 with OpenSession and without OpenSession. The additional overhead in view change by OpenSession control is very small (<50ms).

View change resiliency: OpenSession achieves view change resiliency since victim sites are not overloaded or disconnected during view changes. However, in "no OpenSession", if distributed flow tables are updated without maintaining any particular order, packet losses occur at the victim sites. To measure the gain in view change resiliency with OpenSession, we run the same 3DTI session with and without OpenSession. Packets are dropped if disconnectivity occurs in streaming at the victim sites. Table I shows the total number of packet losses per view change for all streams at all victim sites. It is evident that OpenSession provides higher resiliency in view change.

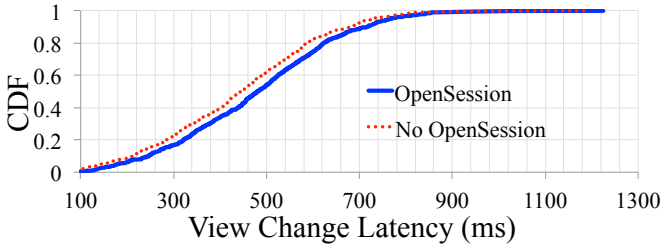


Fig. 10. View change latency (latency between the sites is 15-20ms).

TABLE I. NUMBER OF PACKET LOSSES DUE TO VIEW CHANGE

Approaches	Avg	Stdev	Max	Min
OpenSession	0	0	0	0
No OpenSession	20.35	9.12	45	9

VII. EVALUATION WITH PLANETLAB

We use PlanetLab to evaluate the performance of OpenSession under real Internet artifacts in real-time 3DTI streaming.

A. Evaluation Setup

To run 3DTI in PlanetLab nodes, we collect 15 nodes from different geographical regions. For each 3DTI session, we randomly select 10 nodes. Each node hosts one gateway, one LSC and one SWC. As we do not have access to the network components of the remote nodes, we develop a switch application (keeps FT in memory) that uses f (from Figure 2(b)) to simulate the rule insertion latency. The end-to-end delays are less than 180ms and inbound and outbound bandwidths are within 10Mbps and 250Mbps, respectively. We consider three overlay construction approaches shown below due of their close relation to multi-stream 3DTI applications.

Random: In the Random approach, a random multi-stream topology is constructed during session initiation. It does not

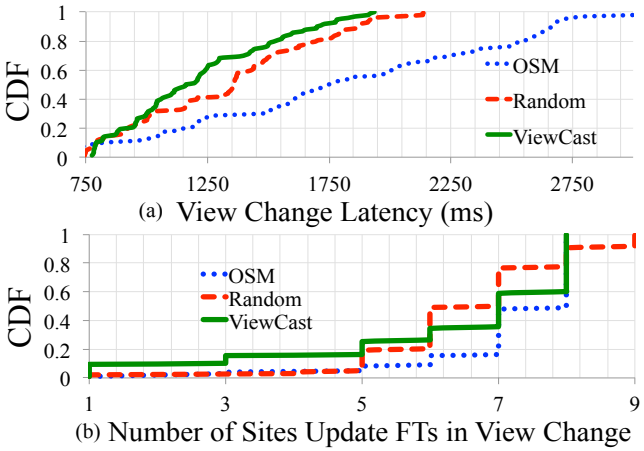


Fig. 11. For different overlay computation approach in PlanetLab setup, (a) view change latency, and (b) number of sites update flow tables.

consider stream priority. During view changes, streaming sources are selected randomly.

ViewCast: In ViewCast [37], initial multi-stream overlays are constructed considering the stream priority. During view changes, the higher priority streams (among the impacted streams) are restored (randomly) first. Though ViewCast is a distributed algorithm, we implement it in a centralized. way.

OSM: In OSM [5], the initial multi-stream overlays are constructed using evolutionary optimization to ensure a globally optimized overlay considering stream priority. During view changes, optimal streaming paths are selected for the streams.

B. Impact of View Change

View change overhead: Figure 11(a) shows CDF of view change latency incurred in OpenSession with three different overlay construction algorithms. Though OSM causes larger view change latency, the overhead mainly comes from the optimization process for the computation of the new overlay after a view change is requested. The average overlay computation latencies for OSM, Random and ViewCast are 1.3sec, 870ms and 763ms, respectively. Since the update of flow tables for a view change occurs at all sites in parallel except at the source site, the communication latency is proportional to the two round-trip delays between the participating sites and the GSC. Figure 11(b) shows the CDF of number of sites update their flow tables over 100 view changes. Most of the cases, 70% of the sites are required to update their FTs. The number of sites here corresponds to the OpenSession message overhead during a view change. The message overhead of OpenSession protocol is very small since the size of a route update message is 400B, which leads to about 2.8KB overhead per view change.

View change resiliency: To understand the impact of view change resiliency in PlanetLab setup, we run 10 3DTI sessions among the PlanetLab nodes with and without OpenSession. In "no OpenSession", FT rules are updated in two different orderings across the distributed site-gateways when view changes are requested. In the first case (case-I), we update FT rules allowing overloading but no disconnectivity on the streaming path. In the second case (case-II), we update FT rules allowing disconnectivity, but no overloading on the streaming path. Over 1 hour experiment, we modify the number of view changes

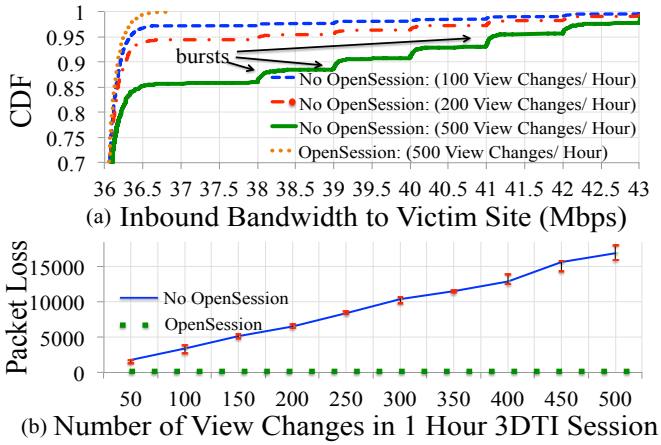


Fig. 12. Impact of view change frequency on (a) inbound bandwidth to the victim site, and (b) total number of packet loss for all streams during 1 hour.

to get different view change frequency. We consider Random overlay construction.

Figure 12(a) plots the CDF of average inbound bandwidth at the victim sites measured in 5 seconds interval during 1 hour session for case-I. Without OpenSession, overloading creates a sudden burst in the inbound bandwidth of the victim sites. The number of bursts increases with the increase of view change frequency. However, in OpenSession, the inbound bandwidth is fairly constant even with higher view change frequency. Figure 12(b) shows the total average number of packet losses at the victim sites for all streams for case-II over 1 hour session. Without OpenSession, the number of packet losses increases linearly with the increase of view change frequency due to the frequent disconnectivity. However, OpenSession does not create any packet loss at the victim sites due to view changes. Similar resiliency can be shown for site failures.

C. OpenSession Overhead

As we implement SWC inside the Floodlight OpenFlow controller, there is an extra processing overhead in the controller. To measure this overhead, we use the *cbench controller benchmarking tool* [23]. We run cbench in *throughput* mode, emulating 1 switch connected to one site. We are able to process 225283 `packet_in` events per second as compared to 241768 in Floodlight without OpenSession (6% overhead).

VIII. Related Work

Multicast routing protocol: IP multicasting protocols (e.g., PIM-SM [1], MBONE [11]) do not provide any application layer programmable interface at the routers and global control for run-time dynamic configurations. To physically deploy PIM-SM, a dependency across ISPs is required, which ISPs do not allow. MBONE overcomes the ISP dependency by creating a virtual multicast network. However, it requires a fixed additional backbone and does not scale. Via OpenSession, we successfully enable quality of service control over some parts (last miles) of the Internet end-to-end data transmission paths without inter-domain control across ISPs. OpenSession control resides only inside the local network and only for 3DTI application-specific traffic.

Multimedia session control protocol: SIP is a session layer signaling protocol [29] used for session management in IP telephony and VoIP services. SIP does not provide mechanisms

to configure data plane for streaming. Other session protocols like RTP [31] uses in-band control functionalities over data to assist streaming, and RTCP [31] is out-of-band control protocol for RTP. Liu et al. [18] propose to decouple data and control architecture for VoD streaming, however the focus is more on the server (CDN) selection. 4D TeleCast [4] provides a control architecture for session management in 3DTI, but it only aims for non-immersive viewers.

Multimedia flow management: The efforts of cross-layer multimedia flow management can be divided into three categories: resource reservation (e.g., [21], [8]), routing (e.g., [35]) and flow scheduling (e.g., [33]). Unlike RSVP, OpenSession does not require control of network routers inside the ISPs for the feasibility concern. OpenSession considers layer-3 control over wide-area-networking and so, MPLS [28] and other layer-2 quality control protocols do not work. Along with different overlay routing approaches [2], various application layer resource reservation and scheduling techniques can still be performed within OpenSession framework.

Application support with OpenFlow: OpenFlow is part of the Stanford University's clean slate project. OpenFlow switches are controlled and modified by OpenFlow controller, enabling flexible functions such as flow forwarding, redirection, multicast, routing, and others (e.g., [20], [25], [30]). OpenFlow has been used in data centers (e.g., [13], [36], [16]) and assisting in scalable video streaming (e.g., [10]).

IX. Discussion and Conclusion

OpenSession protocol adds following features in current real-time collaborative applications (considering Figure 9):

Simplified data plane: The separation of data and control plane in OpenSession makes application logic simpler. Furthermore, the partial offloading of application data plane to the network layer reduces multi-stream management and forwarding overhead of the application host (about 42%).

Improved application performance: The splitting of data plane comes with the benefits of lower bandwidth overhead (over 55% for local streams and 35% for remote streams) within the local network (i.e., last mile) and lower network transmission delay (over 45%) in multicast-based streaming.

Seamless session adaptation: OpenSession handles victim sites in session adaptation process and seamlessly updates the running session without introducing any streaming interference (packet loss or traffic burst) at the victim sites. The improvement is shown by a demo video in [24].

Optimized control plane: OpenSession optimizes the flow table size at the OpenFlow switches. The switch controller does not keep any state, and the processing overhead at the controller is very small (about 6%).

We have shown that in addition to the great potential of SDN in data center, it can be used very efficiently for wide area interactive networking applications like 3DTI. OpenSession successfully leverages the network layer functionality to improve application performance in 3DTI.

References

[1] S. Deering et al. The PIM architecture for wide-area multicast routing. *Transactions on Networking*, 1996.

[2] K. Andreev et al. Designing overlay multicast networks for streaming. In *SPAA*, 2003.

[3] M. Appelman et al. Performance analysis of OpenFlow hardware. http://bit.ly/11SnlGt.

[4] A. Arefin et al. 4D TeleCast: towards large scale multi-site and multi-view dissemination of 3DTI contents. In *ICDCS*, 2012.

[5] A. Arefin et al. Prioritized evolutionary optimization in open session management for 3D tele-immersion. In *MMSys*, 2013.

[6] H. H. Baker et al. Understanding performance in Coliseum, an immersive videoconferencing system. In *TOMCCAP*, 2005.

[7] W. Beyda et al. System and method for reinterpreting TOS bits. USPatent7106737, 2006.

[8] E. Braden et al. Resource Reservation ProtocolRFC 2205, 1997.

[9] Cisco TelePresence Video Conferencing. http://bit.ly/12nJEPE, 2007.

[10] H. E. Egilmez et al. Scalable video streaming over OpenFlow networks: An optimization framework for QoS routing. In *ICIP*, 2011.

[11] H. Eriksson. Mbone: the multicast backbone. *Commun. ACM*, 1994.

[12] Floodlight. http://floodlight.openflowhub.org/.

[13] N. Handigol et al. Plug-n-server: Load balancing web traffic using OpenFlow. In *SIGCOMM*, 2009.

[14] B. Hedlund. On data center scale, OpenFlow, and SDN. http://bit.ly/gLJcIz.

[15] D. Kahneman. Thinking, fast and slow. 2011.

[16] E. Keller et al. Live migration of entire network (and its hosts). In *HotNets*, 2012.

[17] G. Kurillo et al. Real-time 3D avatars for tele-rehabilitation in virtual reality. *Studies in health technology and informatics*, 2011.

[18] X. Liu et al. A case for a coordinated Internet video control plane. In *SIGCOMM*, 2012.

[19] A. Maimone et al. Real-time volumetric 3D capture of room-sized scenes for telepresence. In *3DTV-CON*, 2012.

[20] N. McKeown et al. OpenFlow: enabling innovation in campus networks. In *SIGCOMM CCR*, 2008.

[21] P. L. Montessoro. Efficient management and packets forwarding for multimedia flows. *Network and System Management*, 2012.

[22] OpenFlow. www.openflow.org.

[23] OpenFlow controller benchmark. http://bit.ly/upa5m7.

[24] OpenSession Seamless Route Update. http://bit.ly/13Oep1N.

[25] M. Othman et al. Design and implementation of application based routing using OpenFlow. In *CFI*, 2010.

[26] PlanetLab. http://http://planet-lab.org/.

[27] PolyCom. http://bit.ly/Vkx7PO, 2007.

[28] E. Rosen et al. BGP/MPLS IP Virtual Private Networks (VPNs) RFC 4364, 2006.

[29] J. Rosenberg et al. Session initiation protocol (RFC 3261), 2002.

[30] C. Rotsos et al. OFLOPS: an open framework for OpenFlow switch evaluation. In *PAM*, 2012.

[31] H. Schulzrinne et al. RTP: a transport protocol for real-time applications, (RFC 3550), 2003.

[32] Software-Defined Networking. http://bit.ly/LpV2lD, 2012.

[33] L. Toni et al. Multi-view video packet scheduling. arXiv:1212.4455.

[34] H. Towles et al. 3D tele-collaboration over internet2. In *ITP*, 2002.

[35] R. Vogel et al. Qos-based routing of multimedia streams in computer networks. *Selected Areas in Communications*, 1996.

[36] R. Wang et al. Openflow-based load balancing gone wild. In *HotICE*, 2011.

[37] Z. Yang et al. Enabling multi-party 3D tele-immersive environments with ViewCast. In *TOMCCAP*, 2010.

[38] B. Yu. MyView: customizable automatic visual space management for multi-stream environment. *PhD Thesis*, 2006.

[39] Z. Zhang et al. Demo: Sword fight with smartphones. In *SenSys*, 2011.