Invited Paper

QoS-enabled streaming of multiple description coded video over OpenFlow-based networks

Thuyen Minh Thi¹, Thong Huynh¹, and Won-Joo Hwang²a)

¹ Department of Information and Communication System, Inje University
197, Inje-ro, Gimhae, Gyeongnam, Korea

² Department of Information and Communications Engineering
HSV-TRC, Inje University, 197, Inje-ro, Gimhae, Gyeongnam, Korea

a) ichwang@inje.ac.kr

Received September 30, 2014; Revised December 8, 2014; Published April 1, 2015

Abstract: This paper examines the problem of quality enhancement for video streaming in OpenFlow-based network. OpenFlow can be used in future Internet; it can manage and perform routing for a large number of network flow. We employ OpenFlow to propose a routing mechanism to enhance quality of video streaming. We use expected video distortion as routing metric and apply multipath routing to Multiple Description Coded video. Our simulation results highlight the benefit of exploiting OpenFlow to transmit video.

Key Words: video streaming, OpenFlow network, multiple description coded video, multipath routing, optimization of routing

1. Introduction

In the future Internet, the network traffic grows and the number of kind of traffic also increases. The following problems are challenging: managing network flows, performing quality-of-service (QoS) routing and avoiding congestion. For real-time services (e.g., video telephone, IP-TV, video conferencing), the QoS drops when congestion occurs and leads to high delay and loss. However, best-effort network provides unsteady services to users. In order to have high quality video service, we need to consider QoS enhancement when transmitting video data.

Software-Defined Networking (SDN) is a new approach that can be used in design and deployment of future Internet. OpenFlow is a modern SDN protocol that can be employed to manage and perform QoS routing for network flow. Built-in components of current OpenFlow standard only support to implement simple QoS operations, e.g., per-flow meters are for implementing rate-limiting [1]. Thus, some research proposed to build more highly developed functionalities. In [7] and [6], the authors proposed a framework for QoS routing of scalable video streaming over OpenFlow network. In [24], the authors proposed to automatically control QoS in OpenFlow based on applications/services requirements. Their approaches are flow rate-limiting with flow priority assigning. These previous works are not applicable in the case of our problem: QoS routing for enhancing quality of Multiple
In this paper, we propose a new routing mechanism and try to stream video with low loss rate, delay and therefore low distortion. To this end, we employ OpenFlow to manage video flows, then perform multipath routing on them. The OpenFlow controller (the rest of this paper refers to OpenFlow controller as controller) keeps up-to-date information of the flows and network state, then perform multipath routing for video flows. Non-video flows are routed by conventional routing methods.

In order to apply multipath routing, we use MD video coding which splits a video stream into several independent sub-streams or descriptions. As in [5, 12, 16, 18], this paper use the MD video of two descriptions. The controller finds two paths for two descriptions of each video flow. In this way, we exploit error-resilience characteristics of MD. Our routing objective is to find two paths that minimize expected video distortion. Our model can be applied to some networks such as a backbone network. It allows operators to control quality of their services, especially video streaming.

Our contributions in this paper are summarized as follows:

- We propose a routing mechanism for enhancing QoS of video streaming in OpenFlow-based network (Section 3).
- We formulate an optimization problem for routing with MD coded video (Section 4). We propose a modified Dijkstra algorithm and use it in a subgradient-based algorithm to solve the multipath routing problem (Section 5).

2. Related works

We describe some previous works relating to video transmission and application of OpenFlow.

In [12], the authors proposed to select adaptive MD mode based on network condition and video characteristics. In order to estimate end-to-end distortion, they used a function of source bit rate, distortion in encoding and distortion in transmission. Source bit rate and distortion in encoding were known at the encoder. To estimate distortion in transmission, they used Gilbert model to measure loss. They considered the MD video of two description. Mao et al. [18] minimized video distortion by multipath routing for two MD descriptions. They used distortion-rate function and Gilbert model to estimate distortion. In [5], the authors proposed a multipath selection method for MD video with two descriptions. Distortion was estimated by a function of source bit rate. They had this approximate function by experiment. Our method estimates end-to-end video distortion based on end-to-end delay, packet loss rate and play-back deadline. Packet loss rate is measured by nodes and is collected by OpenFlow controller. Probability of play-back deadline missing is estimated using a Markov chain.

There was some research that suggested methodologies to apply OpenFlow to some new network environments. Li et al. [15] suggested to use SDN in Long Term Evolution core networks. They proposed an idea to use SDN for managing jobs (e.g., resource allocation, fine-grained monitoring, policing). Their work differs from ours since they did not consider QoS enhancement and they did not focus on providing a mathematical model or framework. In [4] and [19], the authors applied OpenFlow to some new network environments (e.g., WLAN, cellular network). They focused on implementation and testing.

The problem of video transmission over OpenFlow-based network is examined in several studies. In [7] and [6], the authors proposed a method to transmit scalable coded video over OpenFlow-based network. They split base layer and enhancement layer. They used OpenFlow and implemented routing algorithm in the controller. Their work differs from ours since they considered scalable coded video and routed it with two priority levels. We use MD coding in which descriptions are encoded and decoded independently. Besides, the authors in [7] use loss and delay to measure path’s cost while we use distortion.

In our method, we estimate the distortion based on delay, packet loss rate and play-back deadline. We propose a mechanism, build a mathematical model and provide its formulation. Besides, we propose a multipath routing algorithm in order to benefit from multi-description property of MD coding.
3. System model

3.1 Network description

The data will travel from sender, through OpenFlow-based network, then arrive at receiver. In this paper, ingress forwarder is referred to as source forwarder, egress forwarder is referred to as destination forwarder (Fig. 1(a)).

The main idea of our mechanism can be briefly described as in Fig. 1(b). Each forwarder serves two types of traffic: traffic of video flows and traffic of non-video flows. Non-video flows are served using conventional forwarding mechanisms. Video flows are routed by our proposed method. When an ingress forwarder receives the packet of connection setup message (S1 in Fig. 1(b)), this forwarder requests controller for forwarding information (S2 in Fig. 1(b)). In OpenFlow protocol, forwarders make this request by sending packet-in [1]. After receives the request, controller finds routing solution which is a pair of optimal paths. Then the controller sends this routing result to all the forwarders on those two paths (S3 in Fig. 1(b)). Other packets of that video flow which come afterward will be forwarded on these paths (S4 in Fig. 1(b)).

We model an OpenFlow-based network as a directed graph \( G(V,E) \), where \( V \) is the set of nodes, \( E \) is the set of links. \((u,v)\) is the link between node \( u \) and \( v \). \((s,d)\) denotes the pair of source and the destination node. They are connected by a set of paths \( R \). The forwarders are referred to as nodes. Table I summarizes important symbols in our paper.

![Fig. 1. OpenFlow-based network and its operations.](image-url)
Table I. Notation.

| Symbols | Definitions |
|---------|-------------|
| $E$     | set of all links $(u, v)$ |
| $R$     | set of all paths $r$ |
| $(u, v)$| a single link between node $u$ and node $v$ |
| $V$     | set of all nodes |
| $r^*$   | optimal path |
| $\delta_e(r)$ | expected end-to-end delay of path $r$ |
| $\Delta$ | video play-back deadline |
| $D(r)$ | expected end-to-end distortion of a source sample when it is transmitted on path $r$ |
| $s,d$  | source and destination node |
| $D_{0,0}, D_{0,1}, D_{1,0}, D_{1,1}$ | distortions when non of descriptions, first description, second description, both descriptions are lost, respectively |
| $p_{0,0}, p_{1,0}, p_{0,1}, p_{1,1}$ | probability with which non of descriptions, first description, second description, both descriptions are lost, respectively |
| $p_{int}^i$ | probability that the packet of $i$-th description arrives at destination intact |
| $p_{ont}^i$ | probability that the packet of $i$-th description arrives at destination on-time |
| $f_{(u,v)}(k)$ | probability function of delay on single link $(u, v)$ |
| $f_e(k)$ | probability function of end-to-end delay |
| $\delta_{(u,v)}$ | delay on single link $(u, v)$ |
| $\delta_e(r)$ | end-to-end delay of path $r$ |
| $\lambda_{ED}$ | arrival rate of examined description on a node |
| $\lambda_{OT}$ | arrival rate of all traffic on a node except examined description |

3.2 Problem

Our problem is to find a pair of optimal paths that transmit video flow with minimal distortion $D(r)$. The problem can be stated as

\[
\text{minimize} \quad D(r), \quad r \in R \\
\text{subject to} \quad \delta_e(r) \leq \Delta, \quad l_{uv} - \sum_{\{v: (v, u) \in E\}} l_{vu} = \begin{cases} 
1 & \text{if } u = s, \\
-1 & \text{if } u = t, \\
0 & \text{otherwise},
\end{cases} \\
l_{uv} \geq 0, \quad \forall (u, v) \in E,
\]
where \( l \) is a vector associated with a path \( r \); \( l \) has components given by
\[
l_{uv} = \begin{cases} 
  1 & \text{if} (u, v) \text{ belongs to} \ r, \\
  0 & \text{otherwise}.
\end{cases}
\]

The first constraint guarantees that play-back delay requirement is met, i.e., expected end-to-end delay \( \delta_c(r) \) must be less than or equal to play-back deadline \( \Delta \). The value of \( \Delta \) is estimated as in [13]. The play-back deadline of each frame is estimated based on inter frame interval and number of frames inter-coded with that frame. Second constraint guarantees that every path \( r \) is formed by connected links and each path connects source node \( s \) and destination node \( t \).

In order to estimate distortion \( D(r) \) of MD coded video, each forwarder in OpenFlow-based network will have built-in MD decoder. There are several methods of MD coding currently being used. Nevertheless, our model is applied to encoding mode that generates two descriptions (Fig. 1(a)). Specifically, it is spatial splitting mode [12]. Each source sample or symbol is coded to form a pair of network packets \([5, 16, 21]\). In formulation, a sample is represented by a pixel. In this way, for each sample \((sb_1 \text{ in Fig. } 1(a))\), we consider two packets \((pk_1 \text{ and } pk_2 \text{ in Fig. } 1(a))\) which carry that sample’s data. These two packets are transmitted on two descriptions. Routing algorithm will find two paths for such two descriptions.

Video distortion mostly depends on packet loss rate. As in [18], distortion \( D \) of each pair of paths \((r_1, r_2)\) is
\[
D = \sum_{i=0}^{1} (p_{i,1}D_{i,1} + p_{1-i,1}D_{1-i,1}). \tag{2}
\]

\( D_{i,1} \) is distortion of a pixel when its data from two descriptions are lost and encoder compensates that pixel by other pixel from previous frame. \( D_{0,0} \) is central distortion when non of descriptions is lost; \( D_{1,0}, D_{0,1} \) are side distortions when first description, second description is lost, respectively. Similarly, \( p_{0,0}, p_{1,0}, p_{0,1} \) and \( p_{1,1} \) are probability with which non of descriptions is loss, first description is lost, second description is lost, both description are lost, respectively.

In order to estimate distortion \( D_{i,j} \) in (2), we apply the ROPE [23] to MD as in [17]. It recursively calculates values of symbols based on packet loss rate and error concealment method.

4. Problem formulation

4.1 Derivation of packet loss rate \( p_{i,j} \)
We consider two causes of packet loss: bad link state (e.g., congestion) and play-back deadline missing.

We examine each pair of paths to choose an optimal pair. Bad link state on each path is calculated from loss rate of its single links. Probability of play-back deadline missing is estimated using discrete time Markov chain.

\( p_{i,j} \) is calculated as
\[
p_{0,0} = p_{1 \text{int}}P_{1 \text{ont}}^2, \tag{3a}
p_{1,0} = (1 - p_{1 \text{int}})P_{1 \text{int}}P_{1 \text{ont}}^2 + p_{1 \text{int}}P_{1 \text{ont}}^2, \tag{3b}
p_{0,1} = p_{1 \text{int}}(1 - p_{1 \text{ont}})p_{1 \text{ont}}P_{1 \text{ont}}^2 + p_{1 \text{int}}P_{1 \text{ont}}P_{1 \text{ont}}^1(1 - p_{1 \text{ont}}), \tag{3c}
p_{1,1} = 1 - p_{0,0} - p_{0,1} - p_{1,0}, \tag{3d}
\]

where \( p_{1 \text{int}} \) is the probability that a description on path \( i \) arrives at destination without loss; \( p_{1 \text{ont}} \) is the probability that a description on path \( i \) arrives at destination on-time.

These equations are interpreted as follows. \( p_{0,0} \) is probability with which two packets arrive at destination intact and on-time. \( p_{1,0} \) is probability with which only second packet arrives at receiver intact and on-time. In this case, first packet may be lost by bad link state (with probability \( 1 - p_{1 \text{int}} \)) or it does not arrive at destination on-time (with probability \( 1 - p_{1 \text{ont}} \)). Probability \( p_{0,1} \) can be calculated in the same way with \( p_{1,0} \). When non of above events happen, we loss both packets (with probability \( p_{1,1} \)).
Fig. 2. Markov chain of examined description’s packets in each node.

**Probabilities relating to link state** \((p_{int}^i)\) In simulation, \(p_{int}^i\) of each path is estimated by multiplying successful transmission rate of links in that path. We use following supports of OpenFlow to calculate such rate. OpenFlow provides measures of drop counts and link speed. Its counter field \([1]\) in flow tables statistically maintain for each port or queue. This field can be implemented in software and it polls hardware counters.

**Probabilities relating to missing play-back deadline** \((p_{ont}^i)\) We consider queuing as the main cause of delay. Markov chain (Fig. 2) is adopted to estimate service time at each nodes \([22]\). State space of the Markov chain is the number of packet in a node’s queue. This chain is discrete time Markov chain with time unit \(T\). This chain is \([X]\) and its state set is \(Z\{Z_{nZ}, Z_{nZ-1}, \ldots, Z_0\}\). When finding a path for a description, we examine service time of the nodes on which the description may be transmitted. Note that we apply \([X]\) to packets of the considered description, not all kinds of packets.

To estimate waiting time of a considered packet, we estimate service time of all packets in queue which arrive earlier than that packet. Formulation of the Markov chain is given below.

Arrival rate of examined description is \(\lambda_{ED}\), arrival rate of other traffic is \(\lambda_{OT}\). Arrival rate of other traffic is arrival rate of all traffics except examined description. \(P_Z\) is probability with which the node is serving another packet rather than a packet of examined description. \(P_Z\) is given as

\[
P_Z = \frac{\lambda_{OT}}{\lambda_{ED} + \lambda_{OT}}.\tag{4}
\]

\(p_{\lambda}\) is probability that there is one packet of examined description arrives in a time unit. We assume arrival process is Poisson process, then \(p_{\lambda}\) is given as

\[
p_{\lambda} = Pr(1 \text{ arrival} / 1 \text{ time unit } T) = 1 - e^{-\lambda_{ED}T}.\tag{5}
\]

We assume that \(T\) is chosen such that there is at most one arrival in each \(T\).

Transition probability matrix of \([X]\) is \(P_X\). \(P_X\) is given as

\[
P_X = \begin{pmatrix}
1 - p_{\lambda} & p_{\lambda} & 0 \\
(1 - p_{\lambda})(1 - P_Z) & (1 - p_{\lambda})(1 - P_Z) + (1 - p_{\lambda})P_Z & p_{\lambda}P_Z \\
0 & \cdots & (1 - p_{\lambda})(1 - P_Z) \\
& \cdots & P_Z
\end{pmatrix}.\tag{6}
\]

Element \(P_X(i,j)\) of \(P_X\) is transition probability from state \(Z_i\) to state \(Z_j\). For example, in the event that \([X]\) transits from \(Z_1\) to \(Z_2\), there must be new arrival and the node must not serve examined description; probability of this event is \(p_{\lambda} \otimes P_Z\).
\[ \Pi_X = \Pi_X P_X. \]  

(7)

The chain \( \{ Y \} \) is derived from \( \{ X \} \) as follows. \( \{ Y \} \) is absorbing chain of \( \{ X \} \), so that transition probability \( P_Y \) is derived from \( P_X \) by replacing \( p_\lambda \) with 0. Moreover, when a packet enters a node, \( \{ Y \} \) is in a specific state \( Z_i \). In order to leave the node, that packet must wait until \( \{ Y \} \) reaches state \( Z_0 \). The time that packet waits is called mean absorption time. Before reaching state \( Z_0 \), \( \{ Y \} \) spends time to transit among its transient states. We need to measure this amount of time, so that we remove the first row and first column of \( P_Y \). The relationship of \( P_Y \) and \( P_X \) is shown in Fig. 3.

\[ f_{(u,v)}(k) = (\Pi_Y(u,v))^T (P_{Y(u,v)})^{k-1} P_{Y'}, \]

(8)

where \( \Pi_Y(u,v) \) and \( P_{Y(u,v)} \) are \( \Pi_Y \) and \( P_Y \) of node \( v \); \( \Pi_Y \) is calculated in the same way with \( \Pi_X \). \( P_Y' \) is the vector of transition probability from \( Z_i \) to \( Z_0 \) \((i = 1, \ldots, n_Z)\). The relationship of \( P_Y' \) and \( P_X \) is shown in Fig. 3.

\( p_{\text{ont}} \) in (3a)–(3c) is probability that a packet arrives at destination on time. Therefore, it is given by

\[ p_{\text{ont}} = Pr(k \leq \frac{\Delta}{T}) = \frac{\Delta}{T} \sum_{k=1}^{\hat{\delta}} f_e(r)(k), \]

(9)

where \( f_e(r)(k) \) is end-to-end delay and it is obtained by an iterative procedure. It is given as a result of convolution operator [22] as follows:

\[ f_e(r)(k) = f_e(s,t)(k) = \sum_{i=s+1}^{n_V} f_{e(i,t)}(k) * f_{e(i,j)}(k), \]

where \( s \) and \( t \) are source and destination nodes of path \( r \); \( n_V \) is number of node of \( r \); \( f_{e(i,j)}(k) \) is probability in which delay of single link \((s, i)\) equals to \( k \), it is given in (8).

### 4.2 Derivation of end-to-end delay

Constraints of (1) requires end-to-end delay to be calculated. We also use modeled Markov chain to estimate this delay.

\( \delta_{(i,0)} \) is the amount of time to transit from state \( Z_i \) to state \( Z_0 \). Its value is multiple of time unit \( T \). For instance, if \( \delta_{(i,0)} = k \), its value is \( k \) time unit(s). It is given as

\[
\delta_{(i,0)} = \begin{cases} 
0, & \delta_{(0,0)} = 0, \\
1, & \delta_{(1,0)} = 1, \\
\sum_{j=1}^{i-1} P_{X(i,j)} \delta_{(j,0)}, & \delta_{(i,0)}.
\end{cases}
\]

(10)

where \( P_{X(i,j)} \) is element \((i, j)\) of matrix \( P_X \) in (6).

For each node \( v \), the mean absorption time is
\[
\delta_{(u,v)} = \frac{n_Z}{\sum_{i=0}^{n_Z} \delta(v,i,0)\Pi_X(v)(i)} ,\quad 1 \leq u \leq n_Z - 1,
\]

where \(\delta(v,i,0)\) is time for node \(v\) to transit from state \(Z_i\) to state \(Z_0\), \(\Pi_X(v)(i)\) is \(i\)-th element in \(\Pi_X\) (Eq. (7)) of node \(v\).

End-to-end delay of a path is the sum of queuing delay at all nodes of that path. End-to-end delay of path \(r\) is

\[
\delta_e(r) = \sum_{u=1}^{n_v-1} \delta_{(u,v)} , \quad (u,v) \in r.
\] (10)

5. Solution procedure

Our solution procedure has two parts: (1) controller’s processing to help controller communicates with forwarders, (2) heuristic-subgradient algorithm to search for routing solution. The controller’s processing help to communicate with forwarders, manage flows and call the heuristic-subgradient algorithm. The heuristic-subgradient algorithm searches for two loopless paths to transmit each video flow.

5.1 Controller’s processing

In the controller’s processing, when there is a new flow, controller puts it into a list to monitor; when a flow is terminated, it is removed from the list. The controller saves recent routing solution, before calling the heuristic-subgradient algorithm, the controller checks whether there is an available and up-to-date solution. In case there is not any new flow but the network state changes, the controller also updates the paths. The controller’s processing is given in Algorithm 1.

The rate at which the controller polls network state is based on network operator’s policy. In the simulation, we set it as in [9]. Network state changes are defined as two following forms.

\begin{algorithm}
\caption{Controller’s processing}
\begin{algorithmic}
\If {packet-in message received = true}
\State put into flow list
\EndIf
\State first_update ← true
\While {flow ended = false}
\If {first_update = true}
\If {\( (r^*_1, r^*_2) \) is available = true and \( (r^*_1, r^*_2) \) is out of date = false}
\State send forwarding information obtained from existing \( (r^*_1, r^*_2) \)
\Else
\State \( (r^*_1, r^*_2) \) ← Algorithm 2
\State send forwarding information obtained from new \( (r^*_1, r^*_2) \)
\State save \( (r^*_1, r^*_2) \)
\EndIf
\Else
\If {\( (r^*_1, r^*_2) \) is out of date = true or network states changed = true}
\State \( (r^*_1, r^*_2) \) ← Algorithm 2
\State send forwarding information obtained from new \( (r^*_1, r^*_2) \)
\State save \( (r^*_1, r^*_2) \)
\EndIf
\EndIf
\State first_update ← false
\EndWhile
\State remove from flow list
\end{algorithmic}
\end{algorithm}
• Changes on network topology, i.e., new configuration, failure of link or forwarder.

• Changes on parameters of cost function represented by \((11)\). Burst error length (from MGE model) is treated as output of a random process so that we neglects it; only changes in \(p_{\text{int}}, \lambda_{\text{OT}}, \lambda_{\text{ED}}\) are considered.

5.2 Multipath heuristic and subgradient algorithm

Our optimization problem is a combinatorial optimization problem. It is likely to be NP-hard problem \([10, 18]\). This paper employs subgradient method to search for solutions.

Then Lagrangian dual function has the following form

\[
L(\lambda) = \min_{r \in R}[D(r) + \lambda(\delta_e(r) - \Delta)], \lambda \geq 0.
\]  

(11)

An idea to deal with discrete optimization problems is exploiting its structure in order to speed up the search for the solution. We exploit Dijkstra’s Shortest Path Algorithm to find routing solutions based on the dual function \((11)\). We modify Dijkstra’s algorithm to find two concurrent optimal paths, instead of one. The routing cost is based on \((11)\).

Dijkstra’s Shortest Path Algorithm is suitable to our problem. The idea of Dijkstra’s algorithm is to traverse from source node, spread out until it meets the destination node. It is likely that the nodes around the source will satisfy constraint of play-back deadline. As a result, it is preferable to have an searching algorithm which is close to breadth-first strategy. In other words, it is better for the algorithm to have an overview of all candidates and make routing decision based on value of objective function. With traversing mode closer to depth-first strategy, the algorithm has less chance to do that.

Conventional Dijkstra’s algorithm will run from source node and gradually add more nodes to a set of selected nodes. For each iteration, it examines all unselected nodes which directly link to one of selected nodes. The set of those examined nodes is called candidate set. From that candidate set, the algorithm selects the best node with least cost. Our modified algorithm goes through the same processes but it will select a pair of nodes when it starts at the source node (Fig. 4). After that, when estimate cost of a candidate node, the algorithm requires two node; we use that candidate node and a previously selected node. Figure 4 illustrates an example of selecting a new node. The algorithm traverses the graph link by link to guarantees that second constraint of \((1)\) is satisfied.

The multipath heuristic and subgradient algorithm is given in Algorithm 2. At step 0, \(\lambda_0\) is initialized so as to satisfy \(\lambda_0 \in R\) and \(\lambda_0 \geq 0\). \(\varepsilon\) is set based on requirement of accuracy and computation time. \(L^*\) is the best known dual value obtained so far. Step 0 and step 4 initialize and update decreasing adaption parameter \(\omega_i\) as in \([11]\); it decreases over iteration and satisfies

\[0 < \omega_i \leq 2.\]

At step 3, due to complementary slackness property \([2]\), when subgradient \(g_i\) equals to zero, we have solution \(L(\lambda_i)\). The pair of paths \((r_1^*, r_2^*)\) are current routing solution.

\(r_{1(i)}\) is the first optimal path, \(r_{2(i)}\) is the second optimal path of Dijkstra’s algorithm, i.e., \(r_{1(i)}\) is better than \(r_{2(i)}\). For this reason, step 1 uses \(r_1\) to estimate \(L(\lambda_i)\) and step 4 updates Lagrangian

![Fig. 4. Node selection of modified Dijkstra’s algorithm.](image)
Algorithm 2: Multipath heuristic and subgradient

\textbf{Step 0: Initialization}
\[ i \leftarrow 0, \lambda_0 \leftarrow 0 \]
\[ \text{initialize } \gamma_0, \varepsilon, \omega_0 \]
\[ L^* \leftarrow 0 \]

\textbf{Step 1: Run modified Dijkstra shortest path algorithm}
\[ (r_1(i), r_2(i)) \leftarrow \left[ \arg \min_{(r) \in R} \left[ D(r) + \lambda_i(\delta_e(r) - \Delta) \right] \right] \]
\[ L(\lambda_i) \leftarrow \min_{(r) \in R} \left[ D(r) + \lambda_i(\delta_e(r) - \Delta) \right] \]

\textbf{Step 2: Update subgradient}
\[ g_1(i) = \delta_e(r_1(i)) - \Delta \]
\[ g_2(i) = \delta_e(r_2(i)) - \Delta \]

\textbf{Step 3: Check stopping criterion}
\[ \text{if } g_1(i) = 0 \text{ and } g_2(i) = 0 \text{ then} \]
\[ (r_1^*, r_2^*) \leftarrow (r_1(i), r_2(i)) \]
\[ \text{break} \]
\[ \text{end if} \]
\[ \text{if } L^* < L(\lambda_i) \text{ then} \]
\[ L^* \leftarrow L(\lambda_i) \]
\[ (r_1^*, r_2^*) \leftarrow (r_1(i), r_2(i)) \]
\[ \text{end if} \]
\[ \text{if } \gamma_i < \varepsilon \text{ then} \]
\[ \text{break} \]
\[ \text{end if} \]

\textbf{Step 4: Update Lagrangian multiplier}
\[ \gamma_i = \frac{\omega_i(L^* - L(\lambda_i))}{\|g_1(i)\|^2} \]
\[ \lambda_{i+1} \leftarrow \max\{0, \lambda_i + \gamma_i g_1(i)\} \]
\[ \text{update } \omega_i \]

\textbf{Step 5:}
\[ i \leftarrow i + 1 \]
\[ \text{goto Step 2} \]

multiplier \( \lambda \) and step size \( \gamma \) by \( g_1(i) \). At step 4, \( \gamma \) is updated using diminishing step size rules [8, 11], then \( \gamma_i \) will satisfy

\[ \lim_{i \to \infty} \gamma_i = 0 \]

and

\[ \sum_{i=0}^{\infty} \gamma_i = \infty. \]

6. Simulation results

We compare our method, named MRVO - Multipath Routing for Video Streaming in OpenFlow-based network, with conventional Dijkstra’s Shortest Path routing (DSP) and Maximally Link-disjoint routing (MLD) [5, 25]. DSP mixes all kinds of traffic and video descriptions, then runs single path routing. MLD finds two paths which have as less link in common as possible. If there is more than one disjoint pair, it chooses the pair with least loss rate. This loss rate is from \( p_{int} \). The evaluating metrics are loss rate, delay and PSNR. We use two video sequence: \textit{Container}, \textit{Foreman}. Their information is shown in the Table II.

In order to evaluate performance of our method, we build a simulated OpenFlow-based network in Opnet modeler. MRVO is implemented at a logically centralized node which has the role of an
Table II. Information of sequence Foreman.

|                      | Container | Forman |
|----------------------|-----------|--------|
| Number of frames     | 300       | 400    |
| Size (bit)           | 668800    | 1763512|
| Time (s)             | 12        | 16     |
| Mean frame size (bit)| 2229      | 4408   |

OpenFlow controller. In order to evaluate the accuracy of our algorithm, we compare the estimation of our algorithm with real performance in simulation. We use Matlab to compute numerically the estimated loss rate and estimated end-to-end delay. These values are compared with real measured loss rate and delay in the run of simulation.

6.1 Simulation environment
Simulation environment is described as follows.

- Input data: we use Quarter common intermediate format (QCIF). Trace file of this video is imported into Opnet. A coded frame of Foreman sequence fits a single packet [3]. The packetization is based on this property so that packet loss will lead to frame loss. Evalvid library is used to deal with video data coding.

- Node and link model: OpenFlow switch is created by designing new node model in Opnet. These nodes drop packets based on their drop rate. This drop rate is estimated from traffic load (i.e., $\lambda_{ED}$ and $\lambda_{OT}$ in (4) and (5)), processing rate and queue length as the equation of $M/M/1/K$ system in [14]. Link model is created to have capacity of 2.5 Gbps and propagation delay of 0.

- Topology: The simulation is conducted with three network topologies. The first topology is a real topology from AT&T route server [20]. It includes 24 nodes. The other topologies are randomly generated with node degree ranges from 2 to 20. The second topology is comprised of 300 nodes and the third one is comprised of 400 nodes.

- Background traffic: we model three background traffic levels. Background traffic is generated by creating random flows in the network. We assume each of these flow is aggregated traffic of many flow in reality. Therefore, these flows have high data rate. Load level 1 has flows with data rate ranging from 512 Kbps to 20 Mbps. Load level 2 and 3 have flows with data rate ranging from 20 Mbps to 60 Mbps, 60 Mbps to 100 Mbps, respectively.

6.2 Illustrative results
We estimate PSNR on YUV files; in result figures, we only show PSRN of Y component. The simulation time is 16 minutes for each run. During each run, the sequence is streamed back multiple times. The PSNR is average value of per-frame PSNR. The loss rate is average value of per-packet loss rate. Similarly, the delay is per-packet end-to-end delay. For each metrics loss rate, delay and PSNR, we make scenarios on three topologies. For each topology, we simulate three load levels.

Loss rate measurement In this section, we conduct a simulation to measure loss.
We stream the sequence Container and collect cumulative received data at destination node. In Fig. 5, our method achieves higher receiving rate than DSP. We add the solid line of play-back rate. When comparing with this line, DSP’s line is lower so that DSP must have initial delay if it wants to stream video without interruption.
In Fig. 6, our method yields less loss than DSP and MLD. In each topology, when traffic load raises, loss rate of the three methods increases but MRVO is lowest. When traffic load is low, the three methods nearly have the same performance. In this case, they are not impacted badly by overloaded links. With the first and third topologies, loss rate of MRVO increases fast with the
increase of load. This can be explained that the first topology (Fig. 6(a)) is small so that most of its links become saturated with high load, making MRVO unable to find good routing paths. In third topology (Fig. 6(c)), highly dense links make MRVO’s performance degrade because of computational complexity. Overall, in comparison with DSP and MLD, MRVO achieves highest performance at second topology. MLD has better performance than DSP but lower than MRVO. Unlike MRVO, it finds disjoint pair of paths to transmit data. If there is not any disjoint pair, it chooses joint pair; if there is more than one pair, it chooses the one with lowest loss rate. It has higher performance than
DSP because it employs multipath transmission.

Figure 6 also illustrates the loss rate from estimation of our method. The estimated values are close to the measured values in the first topology (Fig. 6(a)). This means the estimation achieves high accuracy in this topology. With larger topologies, loss rate is estimated from a greater number of single link’s loss rate, making the accuracy reduce.

Delay measurement The calculation time at controller is affected by network topology. If we do not modify accuracy parameter of the algorithm, the computational complexity makes delay raise significantly. We limit the number of iteration in algorithm in case the complexity is unacceptable. In this case, the algorithm uses the best locally optimal solution. In Fig. 7, when the load of network raises, the queuing delay will suffer, making end-to-end delay of the three methods increase. Overall, MRVO has lowest delay for all load levels. When topology becomes denser, the delay of the three methods is higher. Similar to loss rate evaluation, with the load level 3, the difference between delay of MRVO and two others is highest at second topology (Fig. 7(b)). In routing, MLD does not consider delay so that its performance on delay is lower than its performance on loss. With load level 3 in second and third topology, its performance is lower than DSP.

The Fig. 7 also illustrates end-to-end delay from estimation of our method. Similar to loss rate estimation, our algorithm has high accuracy with small topology and has lower accuracy with larger topologies. When compare to delay estimation, loss rate estimation has relatively higher accuracy. This can be explained that loss rate is estimated based on measurements of OpenFlow, while delay is estimated based on probabilistic model. Measurements of OpenFlow are more likely to have higher
This section uses video sequence Foreman. For PSNR calculation, we use the tool and library from Sunray Image. Figure 8 shows PSNR of three topologies. Each PSNR value of each run is average PSNR of 24000 frames (each run streams the sequence 60 times). The figure shows that performance of the three methods on PSNR is relatively related to loss rate. When the traffic load or the network size increases, the loss rate raises and the PSNR degrades. The figure indicates that our method achieves higher PSNR than DSP and MLD. Especially at load level 3 of the second topology (Fig. 8(c)), our method has 1.11 dB higher than DSP.

7. Conclusion
In this paper, we propose a routing mechanism to enhance QoS of MD video streaming. We benefit from OpenFlow that it supports flow-base control and network state monitoring. Routing metric is estimated video distortion. That estimation is done with regard to end-to-end delay and loss. Loss is estimated by link state measure; delay is estimated by Markov model.

We propose to solve the problem by combining routing algorithm and optimization methodologies. The solution procedure has two parts: controller’s processing and heuristic-subgradient algorithm. The controller’s processing manages flow and communicate with forwarders. The heuristic-subgradient algorithm solves routing problem to find paths to transmit video flows.

In the simulation, we compare our method with a conventional Dijkstra’s method. Our approach gains better performance in term of data rate, delay and PSNR.

Acknowledgments
This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the Global IT Talent support program (NIPA-2014-H0904-14-1006) supervised by the NIPA (National IT Industry Promotion Agency).
References

[1] Version 1.3.0, Openflow switch specification, 2012.
[2] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, Network flows: theory, algorithms, and applications, Prentice Hall, 1993.
[3] J. Apostolopoulos, T. Wong, Wai tian Tan, and S. Wee, “On multiple description streaming with content delivery networks,” In INFOCOM 2002, Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 3, pp. 1736–1745, vol. 3, 2002.
[4] Manu Bansal, Jeffrey Mehmlan, Sachin Katti, and Philip Levis, “Openradio: a programmable wireless dataplane,” In Proceedings of the first workshop on Hot topics in software defined networks, HotSDN ’12, pp. 109–114, New York, NY, USA, 2012, ACM.
[5] Ali C. Begen, Yucel Altunbasak, Zlem Ergun, and Mostafa H. Ammar, “Multi-path selection for multiple description video streaming over overlay networks,” Sig. Proc.: Image Comm., pp. 39–60, 2005.
[6] S. Civanlar, M. Parlakisik, A.M. Tekalp, B. Gorkemli, B. Kaytaz, and E. Onem, “A qos-enabled openflow environment for scalable video streaming,” In GLOBECOM Workshops (GC Wkshps), 2010 IEEE, pp. 351–356, 2010.
[7] H.E. Egilmez, S. Civanlar, and A.M. Tekalp, “An optimization framework for qos-enabled adaptive video streaming over openflow networks,” Multimedia, IEEE Transactions on, vol. 15, no. 3, pp. 710–715, 2013.
[8] U. Faigle, W. Kern, and G. Still, Algorithmic Principles of Mathematical Programming, Texts in the Mathematical Sciences, Springer, 2002.
[9] V.S. Frost and B. Melamed, “Traffic modeling for telecommunications networks,” Communications Magazine, IEEE, vol. 32, no. 3, pp. 70–81, 1994.
[10] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Series of books in the mathematical sciences, W.H. Freeman, 1979.
[11] Michael Held, Philip Wolfe, and HarlanP. Crowder, “Validation of subgradient optimization,” Mathematical Programming, vol. 6, no. 1, pp. 62–88, 1974.
[12] Brian A. Heng, John G. Apostolopoulos, and Jae S. Lim, “End-to-end rate-distortion optimized md mode selection for multiple description video coding,” EURASIP J. Appl. Signal Process., 2006, pp. 261–261, January 2006.
[13] Mei hsuan Lu, Peter Steenkiste, and Tsuhan Chen, “Video streaming over 802.11 wlan with content-aware adaptive retry,” 2005.
[14] L. Kleinrock, Queueing Systems: Theory, Number v. 1 in A Wiley-Interscience publication, Wiley, 1976.
[15] L.E. Li, Z.M. Mao, and J. Rexford, “Toward software-defined cellular networks,” In Software Defined Networking (EWSDN), 2012 European Workshop on, pp. 7–12, 2012.
[16] Ying Li, Chao Tian, Suhas N. Diggavi, Mung Chiang, and A. Robert Calderbank, “Network resource allocation for competing multiple description transmissions,” Trans. Comm., vol. 58, no. 5, pp. 1493–1504, May 2010.
[17] Yiting Liao and J.D. Gibson, “Rate-distortion based mode selection for video coding over wireless networks with burst losses,” In Packet Video Workshop, 2009, PV 2009, 17th International, pp. 1–10, 2009.
[18] Shiwen Mao, Y.T. Hou, Xiaolin Cheng, H.D. Sherali, S.F. Midkiff, and Ya-Qin Zhang, “On routing for multiple description video over wireless ad hoc networks,” Multimedia, IEEE Transactions on, vol. 8, no. 5, pp. 1063–1074, 2006.
[19] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner, “Openflow: enabling innovation in campus networks,” SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, March 2008.
[20] Francesco Palmieri and Ugo Fiore, “Containing large-scale worm spreading in the internet by cooperative distribution of traffic filtering policies,” Computers & Security, vol. 27, no. 1–2,
pp. 48–62, 2008.

[21] V.A. Vaishampayan, N. Sloane, and S.D. Servetto, “Multiple-description vector quantization with lattice codebooks: design and analysis,” *Information Theory, IEEE Transactions on*, vol. 47, no. 5, pp. 1718–1734, 2001.

[22] Yunbo Wang, Mehmet C. Vuran, and S. Goddard, “Cross-layer analysis of the end-to-end delay distribution in wireless sensor networks,” *Networking, IEEE/ACM Transactions on*, vol. 20, no. 1, pp. 305–318, 2012.

[23] Rui Zhang, S.L. Regunathan, and K. Rose, “Video coding with optimal inter/intra-mode switching for packet loss resilience,” *Selected Areas in Communications, IEEE Journal on*, vol. 18, no. 6, pp. 966–976, 2000.

[24] Kim, Wonho and Sharma, Puneet and Lee, Jeongkeun and Banerjee, Sujata and Tourrilhes, Jean and Lee, Sung-Ju and Yalagandula, Praveen, Automated and Scalable QoS Control for Network Convergence, *In Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, INM/WREN’10, pp. 1–1, Berkeley, CA, USA.

[25] Sidhu, Deepinder and Nair, Raj and Abdallah, Shukri, “Finding disjoint paths in networks,” *SIGCOMM Comput. Commun. Rev.*, vol. 21, no. 4, pp. 43–51, 1991.