LETTER

RTSP-Based Adaptive Sending Control for IPTV Service in Heterogeneous Networks and Experimental Implementation*

Soohong PARK[†a)], *Nonmember and* **Choong Seon HONG**[†b)], *Member*

SUMMARY This letter proposes a new mechanism that supports adaptive sending control using Real-Time Streaming Protocol (RTSP) and Transmission Control Protocol (TCP) for IPTV service over heterogeneous networks. The proposed mechanism is implemented on a mobile IPTV device and its performance is verified for providing seamless television watching in heterogeneous networks, even when in motion.

key words: *mobile IPTV, RTSP, TCP quick-start, adaptive streaming*

1. Introduction

Internet Protocol Television (IPTV) is being rapidly expanded to both mobile and wireless technologies (a.k.a. Mobile IPTV) [1]. Typically, IPTV services target quality of experience (QoE) guaranteed wired networks. A number of obstacles to expanding IPTV services to wireless and mobile networks must be overcome in order to guarantee QoE and foster widespread adoption. In particular, QoS support is crucial for successful Mobile IPTV. In the mobile environment, Mobile IPTV services can frequently suffer from an unreliable network connection and insufficient bandwidth. Thus, service continuity requires an awareness of varying network conditions to support the adaptive IPTV streaming service.

To provide the function of adaptive sending control in Mobile IPTV, we chose Real-Time Streaming Protocol (RTSP) [2] due to its advantages described in Sect. 2. In addition, a quick-start TCP algorithm [3] is combined in order to provide faster sending rates than the slow-start algorithm because the adaptive sending control feature is tightly coupled to the transmission layer either UDP or TCP. Despite RTSP over UDP, we tried to optimize RTSP operation over TCP since TCP is used as a transport protocol for the implementation of stable IPTV service. In fact, TCP can prevent packet loss at the application layer due to its connection-oriented feature. However, TCP also cause large delay under such loss events and significant impact to QoE [4]. For the validation of the proposed mechanism in TCP in terms of time delay, we evaluated stabilization time in conjunction with data throughput as described in Sect. 4. During handover, data communications between the mobile device and the network are unstable and shaky for a while and going to be settled with a stable status after few seconds. Then, bit rate is going to be increased continuously. The gap between the handover initiation and the starting point of stable bit rate is defined here as stabilization time.

TCP's slow-start algorithm controls the sending rate using a congestion window (*cwnd*), which limits the amount of data that can be transmitted to the network before receiving an acknowledgement. This algorithm requires a significant number of round-trip times (RTT) and large amounts of data to open the *cwnd* and efficiently use available bandwidth. The quick-start design alleviates the slow-start delay for connections in under-utilized environments like heterogeneous networks. The determination of an appropriate sending rate is fundamental to the function of IP networks including heterogeneous IPTV networks. The TCP sender computes its quick start congestion window by

$$cwnd = \frac{\text{Rate} * \text{RTT}}{\text{MSS} + \text{H}} \tag{1}$$

where Rate is the approved rate request in bps, RTT is the recently measured round-trip time in seconds, MSS is the maximum segment size for the TCP connection in bytes and H is the estimated connection header overhead in bytes. However, the quick-start algorithm requires support from all of the routers along a path. Additionally, the most appropriate method for determining the date rate (Rate in (1)) according to network characteristics remains an unresolved question. Nevertheless, the quick-start TCP is good algorithm when link optimization is low and network changes happen frequently over heterogeneous networks, and well fit for Mobile IPTV.

2. Reason for RTSP-Based Seamless IPTV

This section describes why RTSP is chosen for Mobile IPTV, and advantages of RTSP comparing to HTTP over heterogeneous IPTV networks.

(i) RTSP has similarities to HTTP, but RTSP is a stateful protocol and a session identifier is used to keep track of sessions when needed. However, HTTP is stateless, thus the stateful protocol is suitable for providing IPTV service especially real-time service. In fact, HTTP is originally designed to reliably transfer web resources over internet, not streaming protocol. Since HTTP streaming is operated over

Manuscript received May 29, 2012.
Manuscript revised November 19, 2012.

[†]The authors are with the Department of Computer Engineering, Kyung Hee University, Korea.

*This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012-0006421).

a) E-mail: soohongp@gmail.com
b) E-mail: cshong@khu.ac.kr (Corresponding author)

DOI: 10.1587/transcom.E96.B.905

TCP, it is much more likely to cause major packet drop-outs and greater delay due to TCP with the characteristic which keeps TCP trying to resend the lost packet before sending anything further. Therefore HTTP streaming protocols suffer from the inefficient communication established by TCP's design and they are not well suited for delivering nearly the same amount of streams as RTSP transmission.

(ii) Playback control allows user to interact with streaming contents to control presentation operation in IPTV networks such as fast forward, rewind, time-shift, play in slow motion, etc. RTSP streaming provides such capability to control and navigate the streaming session when the client receives the streaming contents. Unlike RTSP streaming, current HTTP streaming does not provide such capability for playback control that users are accustomed to IPTV viewing which significantly impacts the viewing experience.

(iii) RTSP is already used for the IPTV real networks. That means IPTV providers and standard organizations adopt RTSP to transmit IPTV clients' last-mile link characteristics to multimedia servers as the servers adjust to the adaptive streaming rate.

Recently, MPEG-Dynamic Adaptive Streaming over HTTP (DASH) [10] has been standardized and ready for alternative adaptive streaming in case progressive download may be used for media delivery from standard HTTP Web servers. As described in [10], there are several disadvantages of HTTP-based adaptive streaming such as (i) bandwidth may be wasted if the user decides to stop watching the content after progressive download has started (i.e., switching to another content), (ii) it is not really bitrate adaptive and (iii) it does not support live media services. However, each approach as RTSP-based adaptive streaming and HTTP-based adaptive streaming has own advantages and disadvantages, therefore the selection of available solution should be considered for the targeting scenario and service environment.

3. RTSP-Based New Mechanism

This section describes the RTSP-based adaptive sending control for determining the appropriate sending rate in conjunction with TCP transmission layer. The key points in the proposed mechanism are to deal with the two important parameters as Speed header in RTSP and Rate in TCP according to the changeable network conditions, where the type of the network connection is used for detecting the appropriate sending rate in conjunction with two parameters. It can be replaced with appropriate ones that confirm to the requirements required by a specific IPTV service provider.

3.1 Detailed Flow

Figure 1 shows the operation steps between an IPTV client and a multimedia streaming server. Once network properties are obtained via the Device API (step 1, 2), Speed value in RTSP stack and Rate parameter in TCP stack are set according to the network type (step 3, 4). When the network

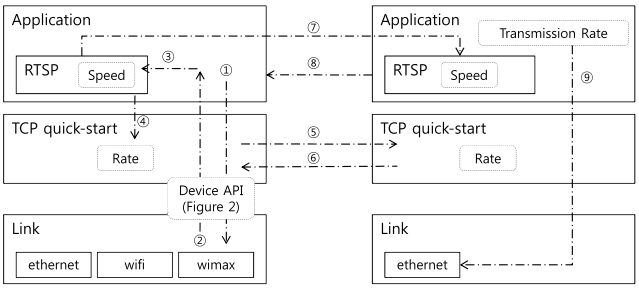

Fig. 1 Work flow of the proposed RTSP-based mechanism between the IPTV client and the multimedia server.

type is identified as the IPTV client, Speed request-header and TCP's quick-start request are triggered in order to inform the server of the client condition changes (step 5 to 8) immediately. Then, the transmission rate is adjusted accordingly (step 9). The detailed work flow the mechanism is as follows.

The first work is to control Speed header in RTSP. Speed request-header field instructs the server to deliver specific amounts of nominal media time per unit of delivery time, contingent upon the server's ability and desire to serve the multimedia stream at the given speed. Speed header determines the bandwidth used for data delivery and is meant for use in specific circumstances, such as when higher or lower rate delivery of a presentation is desired. The server is able to indicate its level of support through a feature-tag (*play.speed*) and Speed parameter values are expressed as positive decimal values. The Speed value of zero is invalid, and the range is specified in the form "lower bound — upper bound."

We conducted a functional experiment to test the efficacy of our mechanism. For this implementation, Speed value range is specified as 'Speed: 0.5–2' where '1' is normal speed, '0.5' is low speed, and '2' is high speed. Once detecting any new network interface, Speed value is determined based on the configured parameters like "network type:ethernet = speed value ('2'), network type:wifi = speed value ('1'), network type:wimax = speed value ('0.5')." These network properties are extracted by a new device application interface (API) in Fig. 2 to trigger quick-start request and identify RTSP speed-request header with the appropriate sending rate when multiple network interfaces are equipped within the IPTV device.

The second work is to control Rate parameter in TCP. To adjust the data rate within TCP quickly, Rate parameter is dynamically mapped to Speed parameter as follows: "speed value ('2')" = 15 Mbps, "speed value ('1')" = 7.5 Mbps and "speed value ('0.5')" = 2.5 Mbps. The defined value of Rate parameter is based on the ITU-T definition [4]. These requirements are for user acceptability of the IPTV services from QoE perspective. The recommended minimum application layer performance can be measured by data throughput and stabilization time used to evaluate the experimental results in Sect. 4.

The third work is to calculate the appropriate sending

rate based on the obtained Speed and Rate parameters in the client side. In TCP, the congestion window is calculated by Eq. (1), and initiates the transport session. In RTSP, the obtained Speed parameter is sent to the server using Speed request-header to request a particular sending rate. Implementation of sending rate changes depend on the server, and we implement Speed parameter features in the IPTV streaming server (speed value (2) = 15 Mbps, speed value (1) = 7.5 Mbps and speed value (0.5) = 2.5 Mbps according to the ITU-T definition.

The last work is to adjust the sending rate in the server side according to RTSP and TCP connections initiated by the client for the new IPTV session. The practical sending rate in the server is subject to change based on the TCP congestion window since the calculated window size can be less than the initiated sending rate from RTSP.

3.2 API Implementation

In order to trigger Speed parameter in RTSP and Rate parameter in TCP quick-start, information about the network type must be sensed immediately when the network type is changed. For Device API, we adapted the Windows operating system Network Driver Interface Specification (NDIS) [5] and the W3C Device API specification [6] as an intermediate driver to support the network switch between Wi-Fi and WiMAX as heterogeneous network. The NDIS is used to communicate with network card drivers and to support basic services such as allowing a protocol module to send raw packets over a network device and to be notified of incoming packets received by a network device. As depicted in Fig. 2, Device API lies between the legacy protocol driver and the lower adapter driver. It behaves like a lower adapter driver from the perspective of the upper level transport driver, and like a protocol driver from the perspective of the lower adapter driver. The intermediate driver supports 1 (virtual network miniport) : 2 (both Wi-Fi and WiMAX) multiplexing. Since we implemented our proposed mechanism on a Linux platform, we adapted a DriverLoader, which is a compatibility-wrapper allowing standard Windows NDIS drivers that are shipped to be used as-is on the Linux systems, for use in this experiment. There are alternatives ways to provide such API to access network interface information like Android OS and Apple iOS. Therefore, the proposed mechanism can be applied for those API according to device features.

The detailed flow of our experimental implementation is as follows: the intermediate driver uses a single virtual network miniport, which is the network driver architecture of the Windows operating system on top of the double lower adapters (i.e., real miniports). The legacy protocol (TCP quick-start and RTSP) driver binds to the virtual network miniport as if it is a real adapter and the intermediate driver binds to the lower adapters (i.e., Wi-Fi and WiMAX) as a generic protocol. The intermediate driver sets the lower adapters (Wi-Fi and WiMAX) to promiscuous mode to receive all frames, and also sets the virtual network miniport Media Access Control (MAC) address to the wireless MAC address. With this configuration, the legacy protocol can communicate with the virtual network miniport as if it is a real lower adapter. All data received from the lower adapter are forwarded to the legacy protocol via the intermediate driver and the virtual network miniport, and vice versa. If the link between the Wi-Fi lower adapter and the driver is interrupted, then the intermediate driver immediately switches the outbound path to the WiMAX lower adapter. Upon enabling the Wi-Fi adapter, the intermediate driver switches the outbound path to Wi-Fi. Importantly, this switching scheme does not change network addresses (i.e., IP and MAC address) at the upper layer since it conceals them from the legacy protocol as well as other hosts within the current network by using the single MAC address. In our implementation, the single MAC address of WiMAX network adapter is configured for the virtual network miniport as well as Wi-Fi network adapter, where Wi-Fi network adapter is configured as the promiscuous mode. More specifically, under the Wi-Fi adapter, packet filtering property is set to promiscuous mode, which accepts all packets. Also, under the WiMAX adapter, the property is set to direct/multicast/broadcast modes, which are typical. Hereinafter, the intermediate driver's MAC address is configured with the WiMAX adapter's MAC address. In other words, under the Wi-Fi adapter, since promiscuous mode is set, corresponding packets can be transmitted and received to the application layer using the WiMAX adapter's MAC address although the inbound packets' MAC address is the Wi-Fi adapter's MAC address. Meanwhile, under the WiMAX adapter, since typical mode is used, the WiMAX adapter's MAC address is just used. Therefore, to the upper protocol drivers (TCP and RTSP), only one virtual adapter set with the MAC address is shown. This operation can skip over the Address Resolution Protocol (ARP) updating procedure, which originated from the differences in MAC addresses between both interfaces. Therefore, an ongoing session can be maintained without data loss while the interface is changed. Also, assigning address dynamically using the dynamic host configuration protocol (DHCP), MAC address is used as one of options of DHCP protocol to identify the corresponding host. Here, by using the same MAC address again, DHCP server recognizes host as the same host and thus the same IP address gets assigned over and over [11], [12]. Moreover,

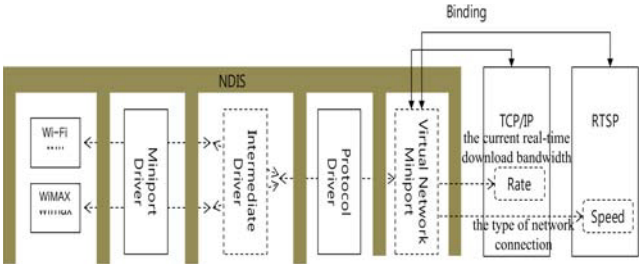

Fig. 2 Devices API design for communications between upper layer protocol stacks and the lower layer via NDIS implementation.

when using MAC address in IPv6 environment assign IPv6 address through stateless auto configuration [13], since the MAC address is the same, it is possible to assign an identical address.

As soon as the new interface is switched by the intermediate driver, the type of network connection for Speed parameter in RTSP and Rate parameter in TCP are exposed and delivered to the higher protocol as TCP and RTSP immediately.

4. Experimental Results

Within our experimental implementation, the mobile device is used for manipulating Speed parameter in RTSP and Rate in TCP quick-start via multiple wireless interfaces, such as Wi-Fi and WiMAX. The IPTV multimedia streaming server's initial transmission rate remains unchanged until Speed and Rate parameters are obtained from the mobile device.

The recommended QoE requirements [4] are measured by data throughput and stabilization time in this letter, because these performances mainly contribute to QoE of the media stream in IPTV as illustrated in [4]. The term bit rate is well established and basically means the number of bits of data that can be transported over the networks. In the case of compressed video like IPTV contents there are two primary processing activities for which the term throughput is often applied, that being encoding and decoding. Data throughput is applied to mean how fast that encoding or decoding process can occur. Therefore, we chose data throughput parameter instead of simple bit rate for our evaluation. Practically, the value of data throughput is practically bigger than the real value of bit rate.

In addition, Table 2 shows stabilization time from the case in which the mobile device moves from a high-bandwidth network to a low-bandwidth network and vice versa. Bit rate equals the amount of data received for stabilization time. In fact, stabilization time is very sensitive factor for QoE while watching IPTV [7]. This can be a major point of dissatisfaction. In particular, how quickly and correctly the subscribers can change channels is an important part of IPTV QoE, and this technical factor (a.k.a. channel zapping) is closely related to stabilization time in this letter. Acceptable channel zapping delay is generally considered to be around 1 sec total, end-to-end. A channel zapping time of 100∼200 ms is considered by viewers to be instantaneous [8]. Consequently, all stabilization time, 143.2 ms and 4.3 ms, measured by the proposed mechanism is in scope of acceptable delay when network interface is changed.

We assessed data throughput and stabilization when the mobile device moved from Wi-Fi to WiMAX and from WiMAX to Wi-Fi and measured a total of ten times for two different cases as follows: (i) a manipulating network condition without Speed and Rate parameter notification, and (ii) a manipulating network condition after transmitting Speed and Rate parameter values. Results of these experiments are presented in Table 2. Through this evaluation, we can see the enhanced throughput and delay against the existing mechanism as RTSP without Speed feature and TCP slow-start combination.

Table 1 Performance requirements for IPTV video service in ITU-T.

Sources type	Video codec standard (Non-inclusive list)	Minimum bit rate (video elementary stream level)
SD broadcast program sources	H.262-Main profile at main level (MP@ML)	2.5 Mbit/s CBR
	H.264 (Main profile at level 3.0), SMPTE 421, AVS	1.75 Mbit/s CBR
H.262 SD, VoD and premium program sources	H.262-Main profile at main level (MP@ML)	3.18 Mbit/s CBR
	H.264 (Main profile at level 3.0), SMPTE 421, AVS	2.1 Mbit/s CBR
HD broadcast program sources	H.262-Main profile at main level (MP@ML)	15 Mbit/s CBR
	H.264 (Main profile at level 3.0), SMPTE 421, AVS	10 Mbit/s CBR

Abbreviations: SD(Standard Definition), HD(High Definition), MP(Measured Point), MP3(MPEG-1 audio layer 3), SMPTE(Society of Motion Picture and Television Engineers), AVS(Audio and Video coding Standard(Chinese)), CBR(Constant Bit Rate).

Table 2 Evaluation results and comparisons.

Measurement	*WiMAX to Wi-Fi*		*Wi-Fi to WiMAX*	
	Existing Mechanism	Proposed Mechanism	Existing Mechanism	Proposed Mechanism
Bit Rate (bps)	4.39 M	7.64 M	1.05 M	2.32 M
Stabilization Time (ms)	636.34	233.24	78.78	9.21

In particular, data throughput increases to 7.64 Mbit/s, and 2.32 Mbit/s, as shown in Table 2. Both resulting data throughput approximately satisfy the minimum requirements of the IPTV services in heterogeneous networks from bit rate perspective since these values include encoding/decoding process. Therefore, the actual bit rate without encoding/decoding is not less than the minimum requirements of ITU-T as depicted in Table 1.

We used Vovida SIP-1.5.0, which is comprised of Session Initiation Protocol (SIP), Real-Time Protocol (RTP), Real-Time Control Protocol (RTCP) and RTSP within the Linux kernel, version 2.6.20.11. In [9], the Linux kernel has been extended in order to support TCP's quick-start extension. Our experimental implementation, within a real TCP/IP stack, allows us to test quick-start in combination with real applications within a real network.

5. Concluding Remarks

We've proposed the new RTSP-based mechanism to support the adaptive sending control according to the network conditions in Mobile IPTV system. The implementation results have shown that the proposed mechanism meted the requirements of the minimum application layer performance recommended by ITU-T for the quality of experience over heterogeneous networks.

References

[1] S. Park and S. Jeong, "Mobile IPTV approaches, challenges, standards, and QoS support," IEEE Internet Comput., vol.13, no.3, pp.23–31, 2009.

[2] H. Schulzrinne, A. Rao, R. Lanphier, and M. Stiemerling, "Real time streaming protocol 2.0 (RTSP)," draft-ietf-mmusic-rfc2326bis-23, IETF Internet Draft, 2010.
[3] S. Floyd, M. Allman, A. Jain, and P. Sarolahti, "Quick-start for TCP and IP," IETF RFC 4782, 2007.
[4] ITU-T Recommendation G.1080, "Quality of experience requirements for IPTV services," ITU-T Dec. 2008.
[5] D. MacDoland and W. Barkley, "Microsoft windows 2000 TCP/IP implementation details," Microsoft White Paper, 2000.
[6] S. Chitturi and R. Berjon, "The network information API," W3C Working Draft, June 2011.
[7] A. Takahashi, D. Hands, and V. Barriac, "Standardization activities in the ITU for a QoE assessment of IPTV," IEEE Commun. Mag., vol.46, no.2, pp.78–84, Feb. 2008.
[8] "Ensure IPTV quality of experience," Agilent Technologies, White Paper, 2005.
[9] H. Strotbek, "Design and implementation of a TCP extension in the Linux kernel," Diploma thesis (in German), University of Stuttgart, IKR, 2007.
[10] T. Stockhammer, "Dynamic adaptive streaming over HTTP — Design principles and standards," Proc. 2nd ACM Conference on Multimedia Systems, pp.133–144, 2011.
[11] R. Droms, "Automated configuration of tcp/ip with dhcp," IEEE Internet Comput., vol.3, no.4, pp.45–53, 1999.
[12] S. Microsystems, "Dynamic host configuration protocol," Whitepaper, 2000.
[13] A. Durand, "Deploying IPv6," IEEE Internet Comput., vol.5, no.1, pp.79–81, 2001.