PAPER

The Enhanced Encapsulation Architecture to Improve TV Metadata Encoding Performance by Schema Optimizing Mechanism*

Bongjin OH[†a)], ***Member***, **Jongyoul PARK**[†], **Sunggeun JIN**[††b)], ***and*** **Youngguk HA**[†††], ***Nonmembers***

SUMMARY We propose simple but efficient encapsulation architecture. In the architecture, clients can better decode Extensible Markup Language (XML) based service information for TV contents with schema digest. Our experimental results show the superiority of the proposed architecture by comparing the compression ratios and decoding times of the proposed architecture and the existing architectures.

key words: *TV contents guide, XML encoding, metadata encapsulation, schema optimization*

1. Introduction

Extensible Markup Language (XML) based service descriptions are widely used by many international digital broadcast standards thanks to their extensibility and readability. For this reason, many TV broadcasting service providers adopt XML to deliver their contents. Meanwhile, TV-Anytime (TVA) is one of the most commonly employed international digital broadcast standards to encode the contents. Accordingly, in the TVA standards, schemas and contents delivery models are defined between a server and clients depending on various types of communication media. The documents described by the TVA schema are called TVA descriptions [1]–[3]. For proper services, the TVA descriptions are divided into independent sub-descriptions, i.e., fragments, and are encapsulated into containers to be provided to clients.

Actually, it has been always necessary to reduce the load of Internet traffic if possible. Besides, recent proliferation of smartphones has brought us an urgent need to reduce the amount of Internet traffic than before for the following reasons. (1) First of all, it is necessary to save the charges while we are enjoying mobile Internet services. Typically, the charges by network service providers are in proportion to the amount of traffic. (2) TV broadcasting services are in services all day long. It implies that the traffic for TV broadcasting services may occupy a large portion of traffic load. Therefore, network service providers also need to reduce the amount of traffic. For this purpose, we try to reduce the TV traffic load by designing an efficient encoding architecture.

Typically, Efficient XML Interchange (EXI) or GnuZIP (GZIP) has been utilized to convert the TVA descriptions into encoded binary information during encapsulation process. Especially, the EXI encoding scheme is adopted as a reference because it is known to have the best performance among other encoding schemes including Binary MPEG format for XML [4]–[8]. Nevertheless, as detailed later, there is a room for improvement for the EXI encoding scheme.

In other words, we find a schema consisting of redundant attributes and elements, which are used to generate TVA descriptions at a server. From the finding, we can optimize the schema by pruning the redundant parts. For this purpose, we design an efficient encapsulation method for the EXI improvement method, called schema digest. The experiment results will show that the proposed method significantly improves the encoding performance compared with the existing EXI.

This paper is organized as follows: Section 2 provides related work. Section 3 describes the proposed architecture in detail. Section 4 explains experimental results, and Sect. 5 shows a reference system using the proposed mechanism. Finally, Sect. 6 concludes this paper.

2. Related Work

2.1 TV-Anytime

TVA specifies the schema to describe details of service and contents guide for digital TVs and Personal Video Recorder (PVR) services. TVA defines several elements for a description of service and contents as shown in Fig. 1. TVA element tree can be expressed as an acyclic tree composed of the elements. The elements for Electronic Program Guide (EPG) are confined to fields related to Service Information (SI), Program Location (PL) and Program Information (PI), and are painted gray in this figure. That is, if a service provider wants to provide users with an EPG for only digital live TV

Manuscript received April 9, 2014.
Manuscript revised November 12, 2014.
Manuscript publicized May 22, 2015.

[†]The authors are with ETRI, 161 Gajeong-dong, Daejeon, 305–350 Korea.
[††]The author is with Daegu University, 201 Daegudaero, Daegu, 712–714 Korea.
[†††]The author is with Konkuk University, 1 Hwayang-dong, Seoul, Korea.

*An earlier version of this paper was published in *Proc. ICCE'11*, 2011.

a) E-mail: bjoh@etri.re.kr
b) E-mail: sgjin@daegu.ac.kr (Corresponding author)

DOI: 10.1587/transinf.2014EDP7113

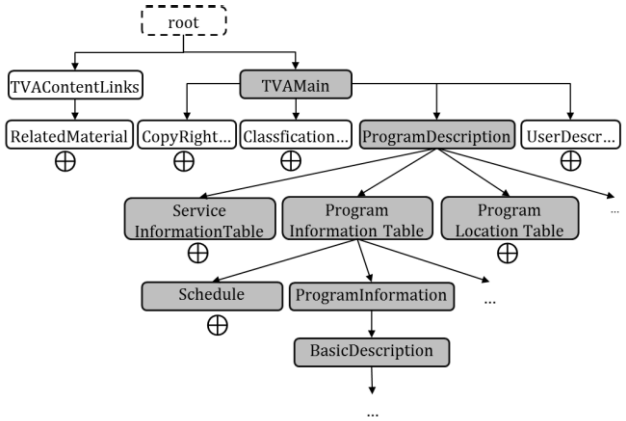

Fig. 1 TVA element tree composed of metadata for contents guide.

services, then they have only to transmit metadata which contains the fields related to three tables to display EPG for a TV contents guide [9]–[11]. In this case, it is recommended to customize the TVA schema according to the service provider's policies because a TVA schema consists of redundant elements unnecessarily describing broad kinds of services and contents.

TVA descriptions are transmitted to clients using a TVA delivery protocol which consists of fragmentation, encoding, encapsulation and transmission steps. Sub-descriptions of TVA descriptions are called fragments. They are encapsulated into containers after being compressed by a selected encoding mechanism. The containers are transmitted to clients through unicast or multicast transport protocols. The server should notify clients of the encoding environment before transmitting the encoded fragments, for proper decoder operation [1]–[3]. This paper proposes an encapsulation architecture using customized schema in order to provide only descriptions which are adaptive to each service provider for schema-based encoding algorithms.

2.2 EXI Encoding Method

The EXI encoding method uses a grammar generated from the schema when TVA fragments are encoded [2], [4], [5]. An exemplary EXI automata graph for an XML schema is shown in Fig. 2.

An EXI-encoded stream consists of event codes, encoded values, and an EXI header. Event codes are assigned to each direct link among nodes comprising the schema automata. Whenever a source node moves to a target node according to an input token of TVA fragments, the mapped event codes are inserted into the EXI-encoded stream. Table 1 shows the events code allocated to the links of above exemplary EXI automata.

The number of links toward target nodes from a specific node determines the length of the event codes representing those links. The codes for the events are encoded with binary number. In fact, each TVA fragment consists of events, i.e., tokens, triggering node transitions so that we can assume set T of the tokens generating a TVA fragment.

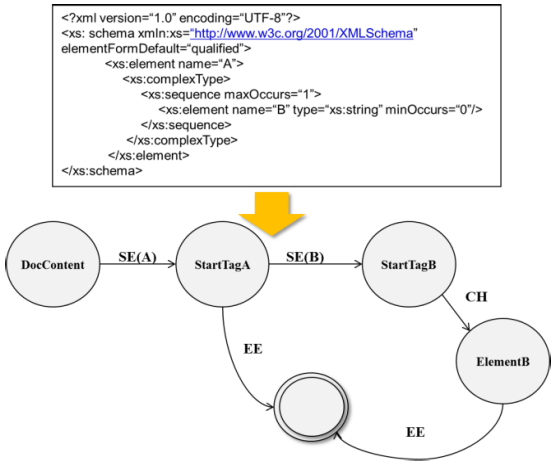

Fig. 2 Exemplary EXI automata and schema.

Table 1 The Events Codes of Exemplary Automata

Node	Events	Event Code
DocContent	Start Element (A)	0
StartTagA	Start Element (B)	0
	End Element (A)	1
StartTagB	Characters (CH)	0
ElementB	End Element (B)	0

If a node contains a specific token $t\,(\in T)$, the code length for the token t is determined by the number of links toward target nodes at the node. Therefore, given N is the number of target nodes directly connected from a source node, we can derive the code length $(= L(t))$ of a token t at the source node by:

$$L(t) = \lceil \log_2 N \rceil, \tag{1}$$

Then, we can obtain the total size of EXI-encoded fragment:

$$size = H + \sum_{t \in T} L(t) + \sum_{t \in T} encode(value_t), \tag{2}$$

where H is EXI header and $encode(value_t)$ is the encoded values of token t. From Eqs. (1) and (2), we can recognize that the size of a TVA document consisting of multiple fragments is influenced by how to reduce the code length denoted with $L(t)$. For this reason, we focus on reducing the average length of event codes by optimizing EXI automata.

3. Schema Optimization Mechanism

The proposed architecture consists of several components to optimize TVA schema before TVA fragments are delivered. The optimized information should be shared between a contents guide server and clients to synchronize the grammar for decoding fragment.

3.1 Proposed Architecture

The TVA description is analyzed to remove the elements

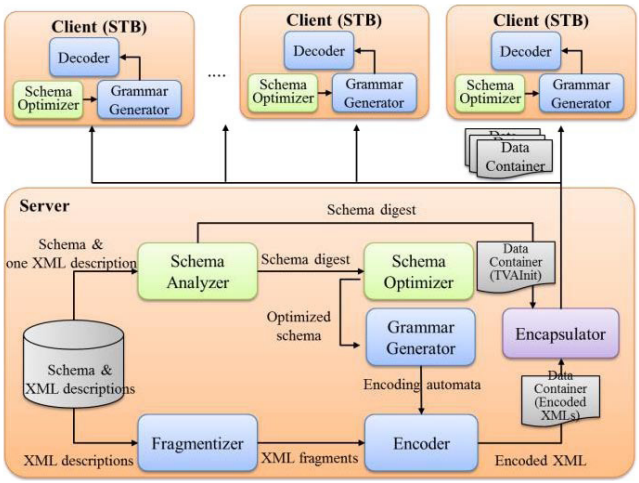

Fig. 3 Encapsulation architecture with schema optimizing mechanism.

not being used to describe details of contents provided by service providers. The schema analyzer digests needed elements from the original TVA schema based on tokens inputted from the service provider's descriptions. The schema optimizer uses the schema digest to generate optimized schema, and transmits the optimized schema to the EXI encoder.

TVA descriptions are fragmented by the fragmentizer, and then inputted to the EXI encoder to be encapsulated for delivery. As described in Sect. 2, the schema-based grammar is used to encode the fragments by replacing their elements with mapped event codes. The size of the grammar affects the volume of the encoding stream, since the average length of event codes is determined by the average number of links among nodes of the grammar. It is clear that the set of links for an optimized schema is smaller than that of the complete TVA schema. Therefore, the encoded stream using optimized schema is always smaller than the original TVA schema-based encoding stream. The schema digest should be transferred to the clients for optimizing TVA schema before encoded fragments arrive for decoding, because the encoded fragments can only be decoded using the same grammar as that of the encoder.

3.2 Optimizing TVA Schema

TVA schema consists of three types of definitions, i.e., (1) basic type definitions, (2) complex type definitions and (3) element definitions. The schema analyzer generates a schema digest by analyzing a service provider's TVA descriptions. Then, the schema optimizer regenerates the TVA schema with only those types and elements included in the schema digest.

The schema analyzer configures an element tree composed of basic types and complex types described in the TVA schema. Visit flags are allocated to all nodes of the element tree with false for their initial values, as shown in Fig. 4.

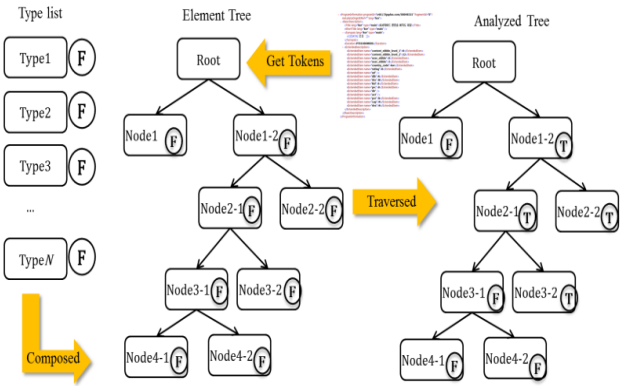

Fig. 4 Diagram of TVA analysis to find elements being used.

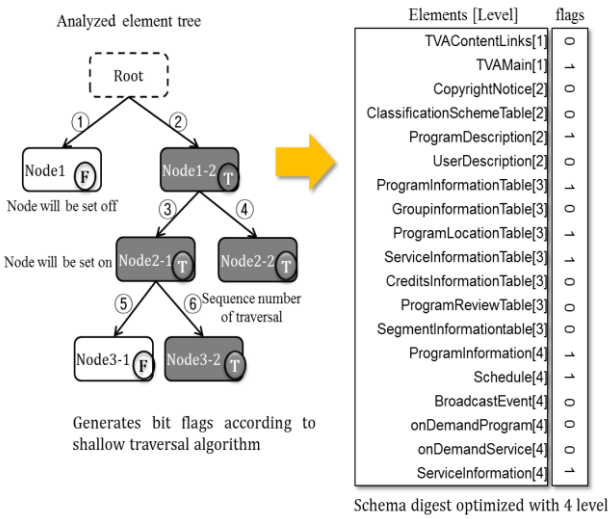

Fig. 5 Schema digests with depth 4 of TVA schema with nodes set as true in the analyzed tree

The schema analyzer gets tokens sequentially from TVA descriptions, and traverses the element tree from the root node to child nodes according to the input tokens. The visit flags of the nodes are set to true when the nodes are visited at least one time, so the nodes with a false visit flag at end time will finally be removed from the original TVA schema.

The schema analyzer traverses the analyzed element tree by shallow traversal algorithm, and the bits for nodes with a true visit flag are set to represent the mapped elements that should be included in the optimized schema. The child nodes of a parent with bits set to off are skipped during the traversal as well as omitted from the bit flag array because it is impossible to describe elements without describing their parents in TVA descriptions.

As shown in Fig. 5, the length of a bit flag array for elements to describe EPG with depth 4 is only 19 bits. The prepared schema digest is transmitted to a schema optimizer for optimizing the original TVA schema. Then, the schema optimizer regenerates a TVA schema using only the elements with true flags of the bit flag array.

The digest should be shared with clients to allow them to decode binary fragments based on the same schema. TVA defines the TVAInitMsg table to notify clients with hints of decoding fragments such as encoding algorithm, buffer size etc. The bit flag array is stored to the TVAInitMsg table, and transmitted to clients before anything else.

3.3 Analysis of Proposed Architecture

We do not consider the schema digest for the derivation of the efficiency for two reasons as follows: (1) the amount of the schema digest is negligible compared with total document size. In detail, the schema digest sizes are about 100 bytes depending on the fragments. However, fragment sizes could be longer than tens of kilobytes and the total document size including multiple fragments is typically longer than hundreds kilobytes. (2) The schema digest is transferred only once at initial service time. Therefore, its influence is negligible considering total document size for the entire service. Keeping the reasons in mind, we continue to derive the effect factor of the proposed encapsulation architecture for encoding rate of event codes is determined as follows:

$$E_f = \sum_{t\in T} L_o(t) \Big/ \sum_{t\in T} L(t), \tag{3}$$

where $L_o(T) = L(T) - L_r(T)$. $L_r(T)$ is the number of links of the node t removed in the optimized schema but which exist in the original schema. If $L_o(T)$ is replaced with $L(T) - L_r(T)$, then E_f is defined as follows:

$$\begin{aligned} E_f &= \sum_{t\in T} (L(t) - L_r(t)) \Big/ \sum_{t\in T} L(t) \\ &= 1 - \sum_{t\in T} L_r(t) \Big/ \sum_{t\in T} L(t). \end{aligned} \tag{4}$$

Therefore, if $L_r(T) = L(T)$ then $E_f = 0$, and this means that there are no event codes in the encoded stream for the optimized schema. If $L_r(T) = 0$ then $E_f = 1$, and this means that there are no improvements in the optimized schema. Therefore, it is valid that $0 \le E_f \le 1$ for any case of removed links in the optimized schema for encoding rate.

Decoding performance is also one of the important issues for user terminals because most user terminals are low-end devices. The grammar to decode tokens should be created before decoding input tokens in the EXI mechanism. Therefore, the decoder creates functions of all the grammars which can be selected from the current node because the decoder has no information about which grammar will be selected for the next tokens. The decoder processes the selected grammar according to the input token, and moves to the next node after destroying grammars created for the previous node. The number of nodes which can be moved to from the current node plays a big role in determining the total decoding time. The effect factor for decoding performance is described as follows:

$$\begin{aligned} E_d &= \sum_{t\in T} (L_o(t)\times(P_c+P_d)+P_e) \Big/ \sum_{t\in T} (L(t)\times(P_c+P_d)+P_e) \\ &= 1 - \sum_{t\in T} (L_r(t)\times(P_c+P_d)+P_e) \Big/ \sum_{t\in T} (L(t)\times(P_c+P_d)+P_e), \end{aligned} \tag{5}$$

where P_c, P_d, and P_e are the processing time to create, destroy and decode grammars at every node moved by token t. Therefore, it is clear that $0 \le E_d \le 1$ for any case of removed links in the optimized schema for decoding performance according to the same kind of inference as that of the encoding rate.

4. Experimental Results

Table 2 shows the details of an experimental environment used to evaluate the performance of the proposed architecture. EXIficient 0.6 is used to encode the TVA fragments with the EXI encoding algorithm. The decoding speed and encoded stream size are evaluated to show the enhancement provided by the proposed architecture.

Table 3 shows the list of PI descriptions used to evaluate the proposed encapsulation architecture. In this table, there are seven PI descriptions used for program details of some channels of the BBC between 2010.09.02 and 2010.09.08.

First, Fig. 6 shows the measured times required to decode the descriptions by using simulator. The proposed mechanism enhances decoding speed, making it about 30% faster than the original EXI. The removed links are the key factor leading to these enhancements, since the proposed mechanism has a smaller average number of links compared with the original EXI.

Second, Table 4 shows the reduced bits of elements

Table 2 Experimental environments

Environments	Details
specification of PC	1.6GHz CPU, 1G RAM, UMPC
evaluation software	EXIficient 0.6 based simulator over JDK1.6
evaluated TVA fragments	PIs published by BBC in 2010.09 (Size 1KB-61KB)
compared algorithms	EXI 1.0 original algorithm, GZIP (JDK GZIPOutputStream)

Table 3 PI Descriptions Used to Evaluation

No	Name	Size (bytes)
1	20100908OneExtra_pi	447
2	20100907BBCRThree_pi	1,737
3	20100908BBCRFiveX_pi	3,616
4	20100903BBCFour_pi	8,623
5	20100903BBCRFiveL_pi	15,831
6	20100902BBCSeven_pi	31,967
7	20100904BBCWrld_pi	61,718

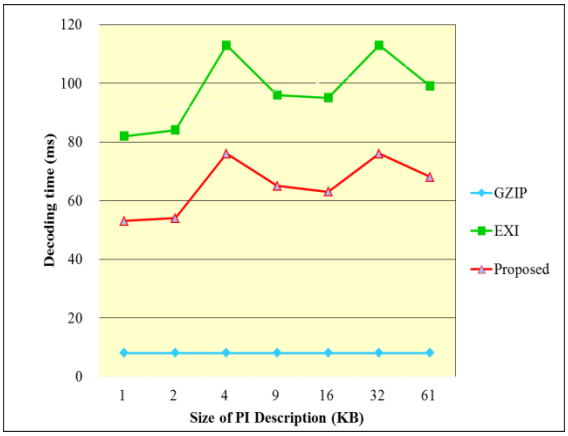

Fig. 6 Decoding performance.

Table 4 Event Codes in Optimized Schema with Depth 6

Event	Source element	Used Elements (No. of original links)	Event Code (No. of bits) EXI	Pr.
SD	Null	SD (1)	0	0
SE	SD	TVAMain (3)	2	0
SE, EE	TVAMain	ProgramDescription , EE (5)	3	1
SE, EE	Program Description	PI, EE (6)	3	1
SE	Program InformationTable	PI, EE (3)	3	1
SE	Program Information	BasicDescription (1)	0	0
SE	BasicDescription	Title, Synopsis, Genre (20)	5	2

of "20100908BBCRFiveX_pi" to show the event code improvements in the optimizing schema method. The event codes are generated by an EXIficient based emulator using the original TVA schema and the customized schema for SI, PI, and Schedule of PL.

For example, the BasicDescription element of the original TVA schema consists of 20 child elements; however only 3 elements (Title, Synopsis and Genre) are used in the BBC's descriptions. Therefore, the proposed architecture generates only 2 bits for event codes from the BasicDescription element to its child elements (the original schema generates 5 bits for event codes). As a result, 3 bits will be saved whenever the migrations from BasicDescription occur, for every PI fragment.

Figure 7 shows the compression performance. From Figs. 6 and 7, we can observe that the proposed scheme is better than EXI in terms of decoding time while it has better compression performance than GZIP.

5. Implementation

This paper shows a Reference Implementation (RI) of the service provisioning platform with the schema optimizing mechanism. Figure 8 shows the diagram for the implemented RI interoperated with Korean IPTV services.

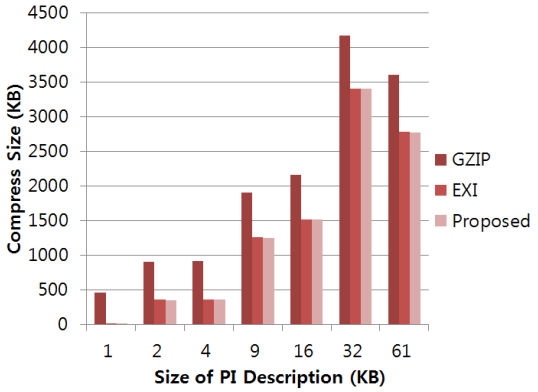

Fig. 7 Compression performance.

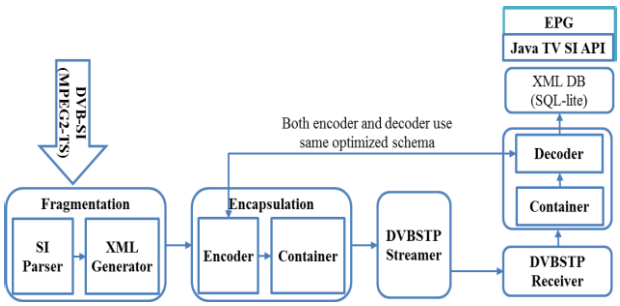

Fig. 8 EPG RI architecture using schema optimizing mechanism.

Korean IPTV service providers adopted the TVA as a standard for their contents guide description, and DVB-SI Transport Protocol (DVBSTP) [12] as their delivery protocol. Nevertheless, DVB-SI [10] based MPEG-2 TS has been used for the contents guide services. For this reason, DVB-SI is used to generate the RI stream for implementation. The SI parser analyzes the input DVB-SI MPEG-2 TS streams, and an acyclic tree is newly configured with the service description tables. Event information tables for all service description tables are extracted from the input DVB-SI stream, also. Those tables are analyzed by XML generator to generate TVA fragments. Service description tables are used to generate SI fragments, and event information tables are used to generate PI and PL fragments, respectively. The generated fragments are encoded using the EXI encoder with optimized TVA schema. They are encapsulated into several containers to be transmitted to a set-top box.

The decapsulated fragments are decoded using the same optimized schema as that of the encoder, and stored into the XML database. An EPG application displays the contents guide using Java TV API according to a user's requests. The Java TV interfaces are implemented to extract information about service channel, program and schedules from TVA fragments [11], [12].

Figure 9 shows the reference implementations of the IPTV server and Fig. 10 shows the IPTV set-top box, respectively. The IPTV server consists of (1) service guide server, (2) provisioning server, (3) stream server and (4) file update

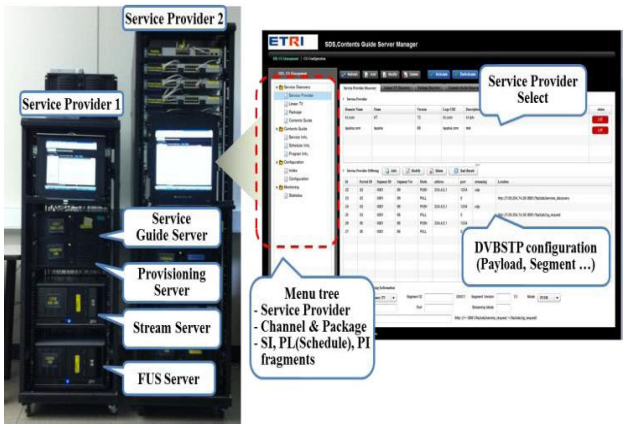

Fig. 9 IPTV server RI and web based management UI to configure the encapsulation and delivery environments.

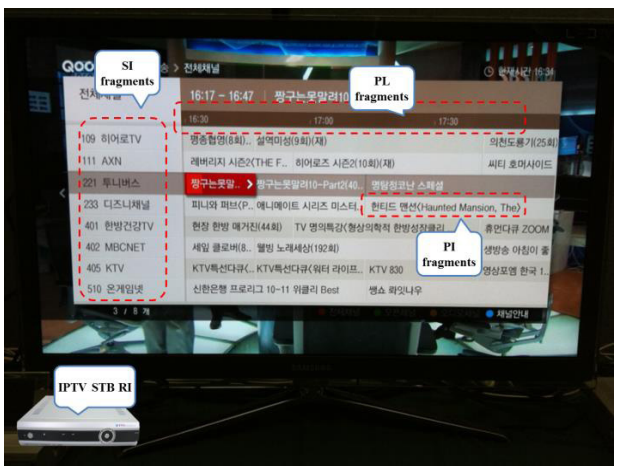

Fig. 10 IPTV set-top box RI and EPG UI to analyze the received TV metadata and display them on the TV Screen.

server. When a user turns on the IPTV set-top box, the device tries to connect to a provisioning server for the device provisioning process. During the device provisioning process, the applications and configuration files are transferred from a file update server to the IPTV set-top box. Thereafter, the IPTV set-top box receives metadata of the contents guide from the multicast address defined in the configuration files. The service guide server transmits the containers of TVA fragments to the multicast address periodically. A web-based management tool is provided to control service guide servers.

The web-based tool configures the service guide servers depending on how the TVA fragments are encapsulated, and delivers the containers. Then, it uploads the configuration file to the file update server. The IPTV set-top box will download the configuration file for a service guide. SI and PL fragments are encapsulated into a single container because their sizes are small enough to be included in one container. However, the PI fragments are encapsulated into several containers because a single container cannot accommodate all the PI fragments.

The layout of EPG UI consists of (1) vertical channel list, (2) single row time line, and (3) title of programs for each channel. The SI fragments are used for the channel list while PL fragments are used for the time line. Additionally, PI fragments are used for the title and details of each program, respectively. The EPG UI is launched over Java based IPTV middleware whenever users want to display the program schedule of each channel on the screen.

6. Conclusion

In this paper, enhanced encapsulation architecture is provided for a TVA-metadata-based contents guide delivery system. The TVA schema optimizing method is used to remove links which are not required to encode and decode service provider's descriptions. Consequently, the average length of event codes is reduced, and the size of the encoded stream is also reduced compared with the original EXI algorithm. Moreover, the removed links reduce total decoding time by removing the time required to create and destroy grammars for the removed links. The proposed method achieves an improved decoding performance 20%~30% better than the original EXI algorithm.

Acknowledgments

This work was supported by the IT R&D program of MKE/KEIT, [KI10039202, Development of SmartTV Device Collaborated Open Middleware and Remote User Interface Technology for N-Screen Service].

References

[1] H. Zhang and S. Zheng, "Personalized TV Program Recommendation based on TV-Anytime Metadata," Proc. International Symposium of Consumer Electronics, LasVegas, USA, pp.242–246, July 2005.

[2] Y.H. Kim, H.-K. Lee, J.S. Choi, and J.W. Hong, "Study on Personalized Data Broadcasting Service using TV-Anytime Metadata," Proc. International Symposium on Consumer Electronics, St. Petersburg, Russia, pp.1–6, July 2006.

[3] B.-J. Oh, Y.-S. Bae, K.-D. Moon, and K.-J. Yoo, "Efficient Retransmission Architecture of Digital Broadcast Services over IPTV Networks," IEEE Trans. Consum. Electron., vol.54, no.1, pp.65–70, Feb. 2008.

[4] T. Podlasek, J. Sliwa, and M. Amanowicz, "Efficiency of compression techniques in SOAP," Proc. Military Communications and Information System Conference, Wroclaw, Poland, pp.199–210, Sept. 2010.

[5] R. Kyusakov, H. Makitaavola, J. Delsing, and J. Eliasson, "Efficient XML Interchange in Factory Automation Systems," Proc. Annual Conference of the IEEE Industrial Electronics Society, Melbourne, Australia, pp.4478–4483, Nov. 2011.

[6] Y.-G. Ha, B.-S. Jang, B.-J. Oh, Y.-S. Bae, and E.-H. Paik, "Effective Encoding of TV-Anytime Metadata Using EXI," Proc. IEEE International Conference on Consumer Electronics, Lasvegas, USA, pp.455–456, Jan. 2011.

[7] B. Oh, S. Jin, E. Baek, and K. Yoo, "A schema Digest Based Metadata Encapsulation Architecture with Shared String Tables," Control and Automation, and Energy System Engineering, Communications in Computer and Information Science, vol.256, pp.154–159,

Springer Berlin Heidelberg, 2011.
[8] S. Cho, D. Shin, H. Jo, D. Choi, D. Won, and S. Kim, "Secure and Efficient Code Encryption Scheme Based on Indexed Table," ETRI J., vol.33, no.1, pp.60–70, Feb. 2011.
[9] B.S. choi, J. Kim, S. Kim, Y. Jeong, J.W. Hong, and W.D. Lee, "A Metadata Design for Augmented Broadcasting and Testbed System Implementation," ETRI J., vol.35, no.2, pp.292–300, April 2013.
[10] S. Morris and A. Smith-Chaigneau, Interactive TV Standards, Elsevier Inc., pp.192–224, 2005,
[11] C. Concolato, "Generation, Streaming and Presentation of Electronic Program Guide," Proc. European Interactive TV conference, Leuven, Belgium, pp.46–49, July 2009.
[12] N.L. Ewald-Arostegui, G. Fairhurst, and A. Yun-Garcia, "A Framework for an IP-Based DVB Transmission Network," International Journal of Digital Multimedia Broadcasting, vol.2010, Article ID 394965, pp.1–13, March 2010.

Bongjin Oh received B.S. and M.S. degrees in computer science from Pusan National University, Busan, Korea in 1993 and 1995 respectively, and the Ph.D. degree from Chungnam National University, Daejeon, Korea in February 2012. Since 1995, he has been with the Electronics and Telecommunications Research Institute (ETRI), where he develops home network middleware and data broadcasting middleware. His research interests are home network middleware, data broadcasting middleware, IPTV, pervasive computing, and big data analytics.

Jongyoul Park received the B.S. degree in computer engineering from Chungnam National University, Korea, in 1996, the M.S. and Ph.D. degrees in information and communication engineering from the Gwangju Institute of Science and Technology (GIST), Korea, in 1999 and 2004, respectively. From 2001 to 2002, he was a visiting researcher at the school of computing, University of Utah. Since 2004, he has been a Research Staff and Director of Analytics SW Research Section of Electronics and Telecommunications Research Institute (ETRI), Korea. His research interest includes IP broadcasting, software middleware, mobile code, distributed computing, big data and analytics platform.

Sunggeun Jin received B.S. and M.S. degrees from Kyungpook National University, Daegu, Korea, in 1996 and 1998, respectively, and the Ph.D. degree from Seoul National University, Seoul, Korea, in August 2008. In 1998, he joined ETRI, Daejeon, Korea. Now, he is an assistant professor at Daegu University. He has participated in standard developments, including IEEE 802.11v, IEEE 802.16j, IEEE 802.16m, and IEEE 802.11ad. Dr. Jin has served as a Technical Program Committee member for the 2008. IEEE Wireless Communications and Networking Conference, the 2009 International Conference on Ubiquitous and Future Networks, the 2010 International Conference on Broadband Communications, Networks, and Systems, IEEE GLOBECOM 2011, and IEEE GLOBECOM 2012.

Youngguk Ha received his BS and MS degrees in computer science from Konkuk University in 1993 and 1995 respectively. He received his PhD degree in computer science from Korea Advanced Institute of Science and Technology (KAIST) in 2006. He worked for Electronics and Telecommunications Research Institute (ETRI) from 1995 to 2008 as a senior member of the engineering staff and currently is an assistant professor in the Department of Computer Science and Engineering at Konkuk University. His research interests are in ubiquitous computing, home network and service middleware, and wireless sensor networks.