Structured Scalable Meta-formats (SSM) for Digital Item Adaptation

Debargha Mukherjee, Amir Said
Imaging Systems Laboratory
HP Laboratories Palo Alto

HPL-2002-326
November 18th , 2002*

E-mail: {debargha, said}@hpl.hp.com

scalable bit-streams, adaptation, transcoding, network, codec-independent, compressed domain, secure

This paper motivates and develops an end-to-end methodology for representation and adaptation of arbitrary scalable content in a fully content non-specific manner. Scalable bit-streams are naturally organized in a symmetric multi-dimensional logical structure, and any adaptation is essentially a downward manipulation of this structure. Higher logical constructs are defined on top of this multi-tier structure to make the model more generally applicable to a variety of bit-streams involving rich media. The resultant composite model is referred to as the Structured Scalable Meta-format (SSM). Apart from the implicit bit-stream constraints that must be satisfied to make a scalable bit-stream SSM-compliant, two other elements that need to be formalized to build a complete adaptation and delivery infrastructure based on SSM are: a binary or XML description of the structure of the bit-stream resource and how it is to be manipulated to obtain various adapted versions; and a binary or XML specification of outbound constraints derived from capabilities and preferences of receiving terminals. By interpreting the descriptor and the constraint specifications, a universal adaptation engine can adapt the content appropriately to suit the specified needs and preferences of recipients, without knowledge of the specifics of the content, its encoding and/or encryption. With universal adaptation engines, different adaptation infrastructures are no longer needed for different types of scalable media.

* Internal Accession Date Only Approved for External Publication
©Copyright 2003 Society of Photo-Optical Instrumentation Engineers. This paper will be published in the Proceedings of the SPIE, vol. 5018, 2003 and presented at IS&T/SPIE's 15th Annual Symposium Electronic Imaging Science and Technology, Internet Imaging IV, 20-24 January 2003, Santa Clara, CA. and is made available with permission of SPIE. One print or electronic copy may be made for personal use only. Systematic or multiple reproduction, distribution to multiple locations via electronic or other means, duplication of any material in this paper for a fee or for commercial purposes, or modification of the content of the paper are prohibited.

Structured Scalable Meta-formats (SSM) for Digital Item Adaptation

Debargha Mukherjee and Amir Said
Hewlett Packard Laboratories, Palo Alto, CA 94304.
Email: {debargha, said}@hpl.hp.com

ABSTRACT

This paper motivates and develops an end-to-end methodology for representation and adaptation of arbitrary scalable content in a fully content non-specific manner. Scalable bit-streams are naturally organized in a symmetric multi-dimensional logical structure, and any adaptation is essentially a downward manipulation of this structure. Higher logical constructs are defined on top of this multi-tier structure to make the model more generally applicable to a variety of bit-streams involving rich media. The resultant composite model is referred to as the Structured Scalable Meta-format (SSM). Apart from the implicit bit-stream constraints that must be satisfied to make a scalable bit-stream SSM-compliant, two other elements that need to be formalized to build a complete adaptation and delivery infrastructure based on SSM are: a binary or XML description of the structure of the bit-stream resource and how it is to be manipulated to obtain various adapted versions; and a binary or XML specification of outbound constraints derived from capabilities and preferences of receiving terminals. By interpreting the descriptor and the constraint specifications, a universal adaptation engine can adapt the content appropriately to suit the specified needs and preferences of recipients, without knowledge of the specifics of the content, its encoding and/or encryption. With universal adaptation engines, different adaptation infrastructures are no longer needed for different types of scalable media.

Keywords: Scalable bit-streams, adaptation, transcoding, network, codec-independent, compressed domain, secure.

1. INTRODUCTION

Users access the Internet today using devices ranging from puny handhelds to powerful workstations, over connections ranging from 56 Kbps modems to high speed 100 Mb/s Ethernet. Even though the available bandwidth, display and processing capabilities may continue to grow following Moore's law, the heterogeneity and the diversity of capabilities at any point in time is here to stay. On the other hand, as bandwidth and other factors grow, so will the richness of media that would need to be delivered to users. Under these circumstances, a rigid media representation format, producing content only at a fixed resolution and quality is clearly inappropriate. A delivery system based on such a compression scheme can only deliver content satisfactorily to a small subset of users interested in the content. The rest, either does not receive anything at all, or receives poor quality and/or resolution relative to the capabilities of their network connections and /or accessing devices. The inability to cater to this diversity has been a determining factor that stunted growth of new rich media, because static rich content would cater only to power users comprising a small fraction of the whole. The bottom line is, without adequate focus on seamless content adaptation, accessibility and usability of media content will always remain limited.

1.1 Multiple versions

A practical approach to catering to heterogeneity is one where multiple versions of any piece of media, suiting a variety of capabilities and preferences, are maintained simultaneously. While this approach works well with delivery models where the recipient directly connects to a media originator, for any other multi-hop, multi-recipient delivery scenario, there is inevitable redundancy leading to wastage of bandwidth and storage. This is especially so, when the media creator intends to provide a wide range of choices for adaptation catering to a large consumer base, and therefore needs to maintain a large number of versions differing in a variety of ways. In other words, this approach does not scale well with the amount of flexibility a media creator would like to provide.

Note however, that since the multi-version case is a fully redundant special case of true scalability to be described next, the framework proposed in the paper still applies. This is also true for hybrid bit-streams that combine multiple versions with true scalability. In all cases, we still need protocols to describe content adaptation choices and request adaptations in a flexible way that the proposed framework allows.

1.2 True scalable bit-streams

In order to provide a solution more elegant than maintaining multiple versions to cater to diversity, scalable compression formats have been proposed. In a scalable bit-stream, smaller subsets of the whole produce representations at lower resolution, quality, etc. Different subset bit-streams extracted from the full parent bit-stream, can readily accommodate a variety of users by automatically maximizing multimedia experience for a given user's computing power, connection bandwidth, and so on. By adapting rich media content written for high-end machines to less powerful machines in various ways, the overheads involved in producing different versions for different scenarios can be virtually eliminated. Furthermore, content created today at the highest possible quality, remains 'timeless' when represented in a scalable format, and the experience it provides gradually increases, as the power of machines, connection speeds, etc. improve.

There are various types of bit-stream scalability that can be designed, depending on the type of media. For example, *SNR* (quality) scalability refers to progressively increasing quality as more and more of the bit-stream is included, and applies to most types of media. *Resolution* scalability refers to fineness of spatial data sampling, and applies to visual media such as images, video, 3D etc. *Temporal* scalability refers to fineness of sampling in the time-domain, and applies to video and other image sequences. There are several types of scalability pertaining to audio, such as number of channels, width of the frequency band. In the future, with the evolution of newer, richer and more interactive types of media, there will be newer types of scalability, for e.g. different kinds of interactivity scalability, which we do not even know yet.

In recent years, there has been a great deal of interest in the research community on scalable compression of various types of digital media. Here the challenge is to obtain a scalable representation without sacrificing compression efficiency. So far however, it is only in the area of still image compression that it has been possible to obtain efficient scalable coders that even improve compression performance (Ex. EBCOT[1], SPIHT[2], VSPIHT[3], EZW[4]). EBCOT[1] led to the evolution of the new JPEG2000 [5] standard for contone images. The JBIG and JBIG-2 standards for binary images are also scalable. Besides images, there has been considerable effort to obtain efficient and compact scalable representations of video, audio, and other types of media. In fact, most existing media encoding standards[6, 7, 8, 9, 10, 11] today, incorporate various scalability modes, although generally there is a loss in compression efficiency to use them, and they are not fully scalable. It is only recently that new fully scalable video codecs, such as 3D-ESCOT[12] and MC-EZBC[13, 14, 15] have been shown to be viable. Fully scalable video coding is currently under exploration in MPEG.[16, 17] Also, there is ongoing activity in MPEG-21 Part 7 Digital Item Adaptation (DIA)[18, 19] related to adaptation and delivery of scalable bit-streams.[20, 21, 22, 23, 24, 25] With evolution of new types of media, it is conceivable that there will be emphasis on scalable representations for them as well, but not everything can be standardized.

A scalable bit-stream does not always have a single type of scalability. In fact, different types of scalability may co-exist in a multi-dimensional structure, so as to provide a wide range of adaptation choices. For example, while SPIHT[2], its predecessor EZW[4], and several of their derivatives[3] only support SNR scalability in their original form, EBCOT[1] endeavors to combine quality scalability and resolution scalability in a common format, to enable distribution and viewing over a wider variety of connections and devices.

Furthermore, in new rich media, different media elements are often clubbed together to provide a composite media experience. For example, an image with audio annotation and some animation provides a composite experience of a presentation using three elemental media elements (an image, an audio clip, some animation data). The composite rich media leads to newer types of scalability specific to the media, because certain non-critical elements may be dropped to accommodate other more critical ones within the limited resources of a recipient.

1.3 Scalable content adaptation and delivery infrastructures

In order to unlock the full potential of a scalable bit-stream, the format alone is insufficient. It is necessary to develop and deploy complete infrastructures that support appropriate adaptation and delivery of such content, so that a diverse recipient base can experience it with seamless ease of use.

For example, even though the JPEG2000[5] format itself is very powerful, the lack of a complete end-to-end infrastructure that supports appropriate adaptation of JPEG2000 content and delivery to a heterogeneous recipient base has severely restricted the usability of its scalable features. In recent years, a great deal of attention has been focused on delivering streaming video over the Internet or wireless.[26, 27, 28] In order to reach heterogeneous recipients in a dynamic transmission environment, video standards of MPEG-X (mostly MPEG-4)[6, 7, 8, 9] and H.26X[10, 11] families incorporate various forms of scalability. Although limied in scope, functionality and efficiency, as compared to JPEG2000, they hold considerable promise for supporting diversity. Nevertheless, scalable video over the Internet has been limited to maintaining multiple versions for a few different types of connections, because complete infrastructures that support transcoding and transport of scalable video formats are non-existent. It is very recently that a new standardization effort has started in MPEG on fully scalable video coding. However, without adequate focus on content adaptation in the network, its scalability features would remain unexploited.

1.4 Need for media-type agnostic adaptation Infrastructures

Any infrastructure, is expensive to deploy, and requires significant financial commitments from patron companies or patron consortia. Under these circumstances, use of a standardized format for the content is desirable in order to guarantee constancy. On the other hand, standards evolve all the time. There are often extensions added to existing standards to support better and enhanced features, and it is conceivable that as new types of media beyond traditional images, video and audio evolve it would be necessary to create new standards for their representation. It can also be argued that since standards take several years to come into effect, typically much longer than is commensurate with the normal pace of change in the multimedia industry, it would become more and more difficult to expect standards to support the representation and delivery of new types of content.

Even if compact scalable formats evolve for every new type of media, the inevitable difference in the structure of the content would necessitate use of different infrastructures or components thereof for scalable delivery of different types of media. The expenses involved present a very formidable obstacle in adoption of such new media and supportability of its scalability features.

The only way out is to develop infrastructures for content adaptation that are *media-type agnostic*. Such universal adaptation infrastructures only need to be deployed once to support adaptation and delivery of all types of scalable media, as long as they conform to certain loose restrictions on the encoding structure. Use of universal infrastructures that support delivery and transcoding of a wide variety of media types in a convenient manner is the key to successful adoption of new scalable media.

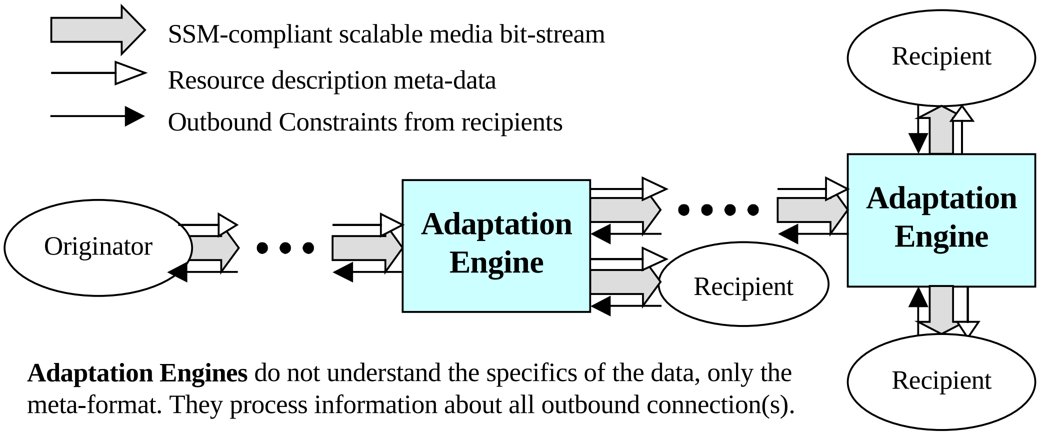
1.5 Security

The need for secure communication of media content is already being felt, and in the future will be the order of the day. In order of guarantee full end-to-end security, it will be necessary to use delivery architectures where no codec-specific elements are used in the entire path from the content server and perhaps including it, to the receiving terminal. Anywhere in the network that a codec-specific element is used, is potentially a security breach point.

Even in such a secure transmission scenario, midstream content adaptation to cater to diversity would be a necessity. Secure end-to-end streaming using scalable packets has been demonstrated earlier.[26, 27] However, to enable secure content adaptation in a content-agnostic manner, it is necessary to empower network adaptation engines to make decisions about possible adaptations, even when they do not understand the semantics of their decision. The only way to do this is to provide information needed to make decisions in a compact way using a mathematical abstraction that is content-agnostic, and incorporate them into generic descriptors that an adaptation engine can process.

1.6 Motivation for this work

In order to enable universal *media-type agnostic adaptation* infrastructures for scalable media, we propose a framework for scalable media representation that formalizes a loosely defined model or meta-format for all scalable media types rather than a single format for a specific media type (such as JPEG2000 for images). The model is called **S**tructured **S**calable **M**eta-format (**SSM**). Although the meta-format needs to be standardized, it operates at a more abstract level than traditional standards, and requires only format compliance in a loose manner. All compressed scalable bit-streams that need be adapted using the universal adaptation infrastructure must be *SSM-compliant, i.e.* must conform to the meta-format. Besides the bit-stream constraints imposed by SSM-compliance, in order to enable end-to-end adaptation and delivery of SSM compliant media, it is also necessary to standardize the languages for conveyance of information, both between an adaptation engine and the media server, as well as between the adaptation engine and the media


SSM-compliant scalable media bit-stream
Resource description meta-data
Outbound Constraints from recipients
Originator
Adaptation Engine
Recipient
Adaptation Engine
Recipient
Recipient

Adaptation Engines do not understand the specifics of the data, only the meta-format. They process information about all outbound connection(s).

Figure 1. Media-Type-Independent Adaptation and Delivery Chain

receiver. Adaptation engines in the delivery chain need to be able to interpret this communication for adaptation purposes.

Media adaptation and delivery infrastructures based on SSM and standardized communication with adaptation engines would be truly media-type- and content-independent, in that they can adapt different types of content, both that are currently available (images, video, audio) as well as those that would evolve in the future (different types of new 3D media, composite media etc), in a secure manner, as long as they comply with the model.

2. DIGITAL ITEM ADAPTATION WITH SSM

2.1 SSM based delivery architecture

Consider Figure 1, which shows a generic media delivery model, where media data created by the originator is routed through an arbitrarily long chain of adaptation engines before reaching an eventual recipient. It is assumed that both the originator of the media as well as the software or hardware system used to experience it at the recipient end understand the actual media-encoding format. It is likely that either the same company created both the media content and the experiencing system, or the creator opened up its technology for vendors to develop the experiencing system, or the media format is an SSM-compliant open standard.

Irrespective of the actual content-type and its encoding however, the scalable resource bit-stream is conformant with SSM, which all intermediate adaptation engines can interpret and manipulate. These engines receive SSM compliant scalable content, and deliver adapted content over multiple outbound streams. All content after adaptation is also SSM meta-format compliant so that it can be re-adapted at a subsequent stage of delivery.

Along with the SSM-compliant media bit-stream an adaptation engine also processes a description meta-data (shown in thin white arrows in Figure 1) that contains vital information for the adaptation engine about all possible adaptations. The SSM model restricts the possible adaptation choices and allows a compact representation of this description. The adaptation engine not only adapts the media bit-stream but also the description meta-data, so that a subsequent stage of adaptation can be applied. There are a variety of possibilities for representing and conveying this information to adaptation engines. While in MPEG-21 DIA the trend is to use XML[25], it is to be noted that representing this information in binary form as part of the media bit-stream itself is a straightforward extension, and may even be preferred based on considerations of compactness and manageability. This information is referred to as *resource description metadata*.

It is also assumed that each adaptation engine has knowledge of the aggregated capabilities and preferences of all eventual recipients connected to each of its outbound streams. This information mostly originates from the recipients (shown in thin black arrows in Figure 1), but parts may be sensed by transcoders themselves, as it is aggregated up the adaptation chain by the delivery infrastructure involved. For a particular engine for a particular outbound connection at adaptation time, this information is referred to as its *outbound constraints*, which in general may change dynamically.

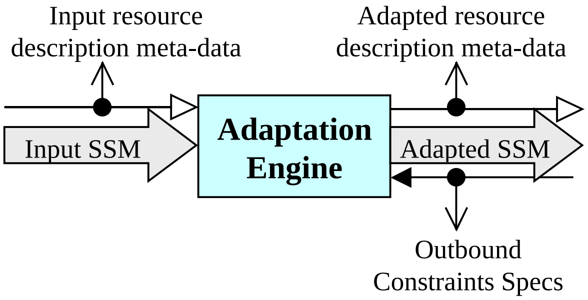


Figure 2. Adaptation Engine external model

Note that while the originator/creator of the media as well as the recipients/consumers of the media must have specific knowledge about the encoding in order to provide an experience for the end-user, the intermediate infrastructure does not need to know what the content is and how it has been encoded in order to adapt appropriately. The adaptation operation is based purely on an interpretation of the resource descriptor metadata and the outbound constraints, and does not depend on the specifics of the actual content. Furthermore, the content itself can be encrypted, and transcoding can still proceed as before in the encrypted domain.

While adaptation engines in Figure 1 are solely functional blocks, in reality they can be part of media servers from where offline or online content originates; midstream routing servers through which scalable content is transcoded and routed; or edge servers that connect directly to eventual recipients. Also, the generic delivery model considered can collapse to as simple as a client-server delivery system where a client requests content from a media server with specified capabilities and preferences, and gets appropriately adapted content directly from it. In this case, the functional adaptation engine would be part of the media server itself.

2.2 Isolated adaptation engine model

In order to understand the scope of the proposal, we isolate a single input single output functional adaptation engine from the end-to-end delivery model, and show its external model in Figure 2. As discussed above, the adaptation engine receives a SSM compliant piece of scalable media, which it must adapt appropriately and forward in a SSM compliant manner to an eventual consumer or another adaptation engine. Along with the media bit-stream it also receives a *media description* metadata (XML or in the bit-stream) providing the specifics of how the bit-stream is to be adapted for various adaptation options, as well as an *outbound constraints specification* (possibly XML) conveying the capabilities and preferences of its output connection. Based on the information contained in the descriptor and the specification, an

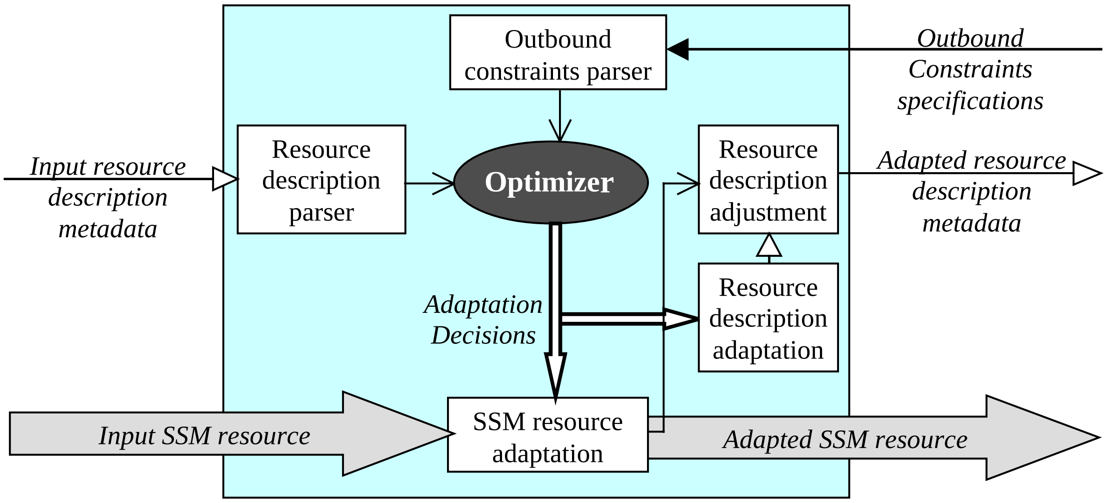


Figure 3. Adaptation Engine Internal model

adaptation engine makes certain adaptation decisions, performs the adaptation operation based on these decisions on the input SSM-compliant stream, and delivers SSM-compliant adapted content to its outbound connection. In addition, it can also update the media description meta-data for use in a subsequent adaptation stage.

The internal model for the adaptation engine is shown in Figure 3. In particular, it consists of the following functional blocks: 1) A parser for the resource description meta-data; 2) a parser for the outbound constraints specification, 3) An optimizer to decide on transcoding options, 4) a SSM resource adaptation engine to adapt the resource based on adaptation decisions made by the optimizer, 5) a resource description metadata adaptation engine to modify the resource description based on decisions made, and 6) a second phase resource description metadata adjustment engine to make adjustments to the resource description (needed only in certain cases).

It is to be noted that the SSM framework does not determine the operation of the optimizer module in the adaptation engine. The way the adaptation engine arrives at the optimal adaptation decisions based on the resource description and outbound constraints is open to implementation. However, everything else in the adaptation engine is more or less determined by the proposed adaptation framework. Furthermore, once the decisions have been made, the resource and description adaptation is also deterministic.

2.3 Relation to network packetization

It is important to realize that while the SSM framework is about modeling generic scalable content, and metadata describing the model parameters and how model-compliant scalable content is to be adapted, in an actual delivery scenario the content would probably need to be packetized and transmitted. In this regard, among various design choices, there are two that are of particular interest, one based on interpretation of SSM as a file-format, and another based on interpretation as a packet-format.

In the file-format usage case, the scalable resource is actually much larger than a typical network packet. Either the adaptation engine adapts an entire SSM file in one shot before network packetization and transmission, or the adaptation is conducted down-stream possibly in multiple stages. In the latter case however, it is important to realize that it is not necessary that the entire SSM compliant resource be available at the adaptation engine before the adaptation operation can commence. In fact, the resource description and the outbound constraint specifications are all that are needed for an adaptation engine to decide how to adapt the media content. As long as the meta-data has been received in full, the scalable bit-stream resource in Figure 3 may come in stages in multiple network packets, and either forwarded or dropped or partially dropped by the engine as they arrive, based on the adaptation decisions already made. Thus, the same adaptation model applies both to files transcoded in one shot as well as to a streamed file.

In the packet-format case, the entire SSM compliant content comprises one packet, which can be adapted by a mid-stream adaptation engine and transmitted. Packet based scalable adaptation based on truncation has been considered before.[26, 27] In this scenario, it may make more sense to include the resource description as part of the packet, using a form of binary encoding rather than XML.

In the rest of this paper, we will describe the specifics of the SSM framework: how a generic scalable bit-stream is modeled and what the required constraints are, what information is contained in the resource description metadata that goes with the resource, and how the capabilities and preferences are conveyed in the outbound constraints specifications. We will also describe how the adaptation decisions are made at a network adaptation engine.

3. MODELING SCALABLE BIT-STREAMS

A scalable bit-stream is one where smaller subsets of the whole produce representations at lower quality, resolution etc. Different types of scalability (e.g. SNR, Resolution, Temporal, Interactivity, etc.) apply to different types of media, and often more than one kind is combined. Furthermore, in rich media content several independent elements can be combined, (e.g. video, audio). From an understanding of how a generic scalable bit-stream is naturally organized, we propose a logical model for all scalable bit-streams, referred to as the SSM model.

3.1 Symmetric nested scalability structure

The proposed SSM framework is based on the assumption that any scalable bit-stream component inherently contains logical nested tiers of scalability, as shown in Figure 4. Using zero-based indexing, the bit-stream is first divided logically into multiple layers of tier 0 scalability. Here tier 0 is an abstraction, and

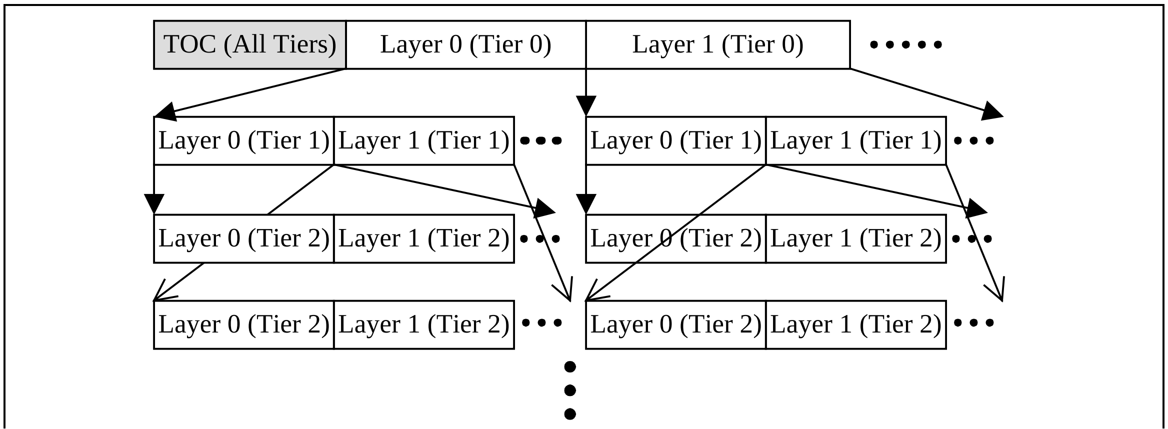


Figure 4. Symmetric nested scalability model

depending on the actual content it may mean any one of resolution, temporal, SNR and so on. Each data segment in each tier 0 layer, is further divided into layers of tier 1 scalability, and so on. Again, tier 1 is an abstraction, and may mean different things based on the actual media content. And so on.

As an example, consider a JPEG2000 bit-stream, which can be readily cast into this logical-bit-stream-format. In one of the scalability progression modes in JPEG2000 – RLCP – the highest tier is resolution scalability, and within the resolution scalable layers there are nested SNR scalable layers. In an alternative scalability progression mode – LRCP – the highest tier is SNR, and within SNR layers there are nested resolution layers. However, the multi-tier nested scalability structure is common in both.

The above-described logical meta-format is analogous to that of a book, where there are nested layers for chapters, sections, sub-sections and so on. It is conceivable that the book-format be common across all books irrespective of content. Likewise, all scalable bit-stream representations can be cast into a common nested scalability structure that can be standardized into a bit-stream-format, irrespective of content.

Note that the above nested structure is merely logical, in the sense that in the actual bit-stream there is more freedom in where the data segments lie. Normally however, the layers at the deepest tier, called *atoms*, form a single contiguous segment of the bit-stream that can be easily dropped as part of an adaptation process. There can also be arbitrary filler code in between the atoms.

3.2 Notation and data-cube representation

Formalizing the notation for the bit-stream, if the data has L nested tiers of scalability, and the *i*th tier contains l_i layers, we can say that the data consists of an ordered concatenation of $l_0 \times l_1 \times \ldots \times l_{L-1}$ data chunks B(j_0, j_1, …, j_{L-1}), where j_0=0,1,…, l_0–1; j_1=0,1,…, l_1–1; …; j_i=0,1,…, l_i–1;…; j_{L-1}=0,1,…, l_{L-1}–1. A way to visualize this data is to consider a L-dimensional *data cube* of size $l_0 \times l_1 \times \ldots \times l_{L-1}$, the ($j_0$, j_1, …, j_{L-1})th element of which is the data chunk B(j_0, j_1, …, j_{L-1}), called the *atom*. The full bit-stream is essentially a concatenation of these data chunks if the indices are scanned in order from j_{L-1} towards j_0.

Using an example of the first two tiers of JPEG2000 RLCP progression mode, we can visualize the data as

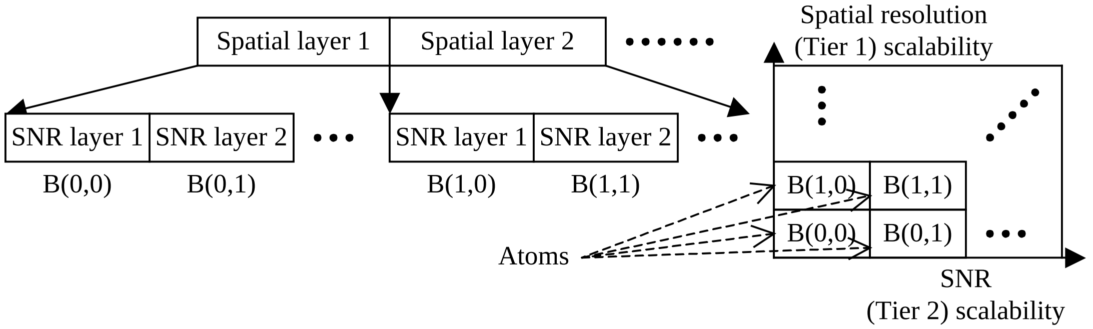


Figure 5. JPEG2000 example bit-stream

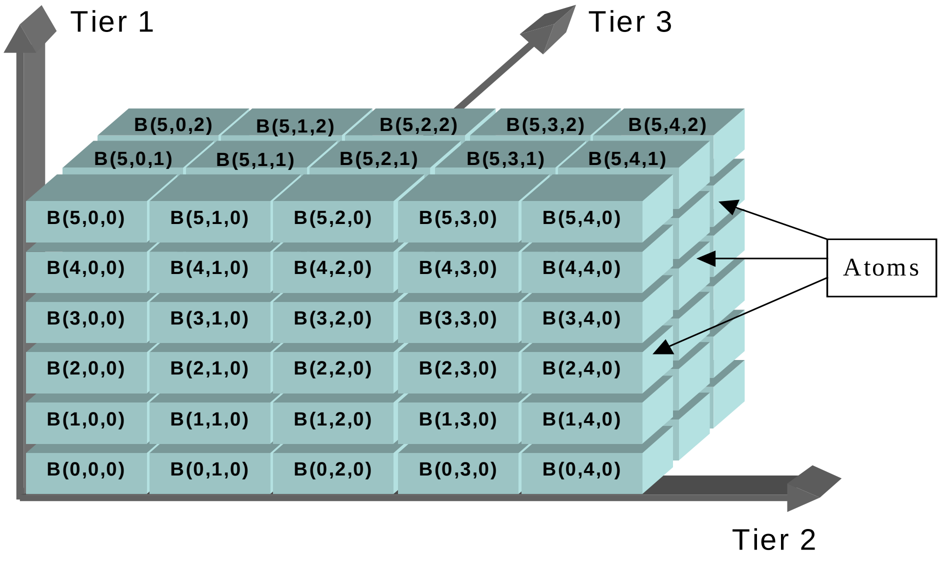


Figure 6. Data cube representation of multi-tier scalable media

organized in a 2-dim cube (L=2) as shown in Figure 5. The bulk of the bit-stream apart from any filler code and an optional TOC can be visualized as being obtained by scanning the atoms in the data cube in some order. The same concept generalizes readily to more than two dimensions or nested tiers. An example of a three-dimensional data cube is shown in Figure 6.

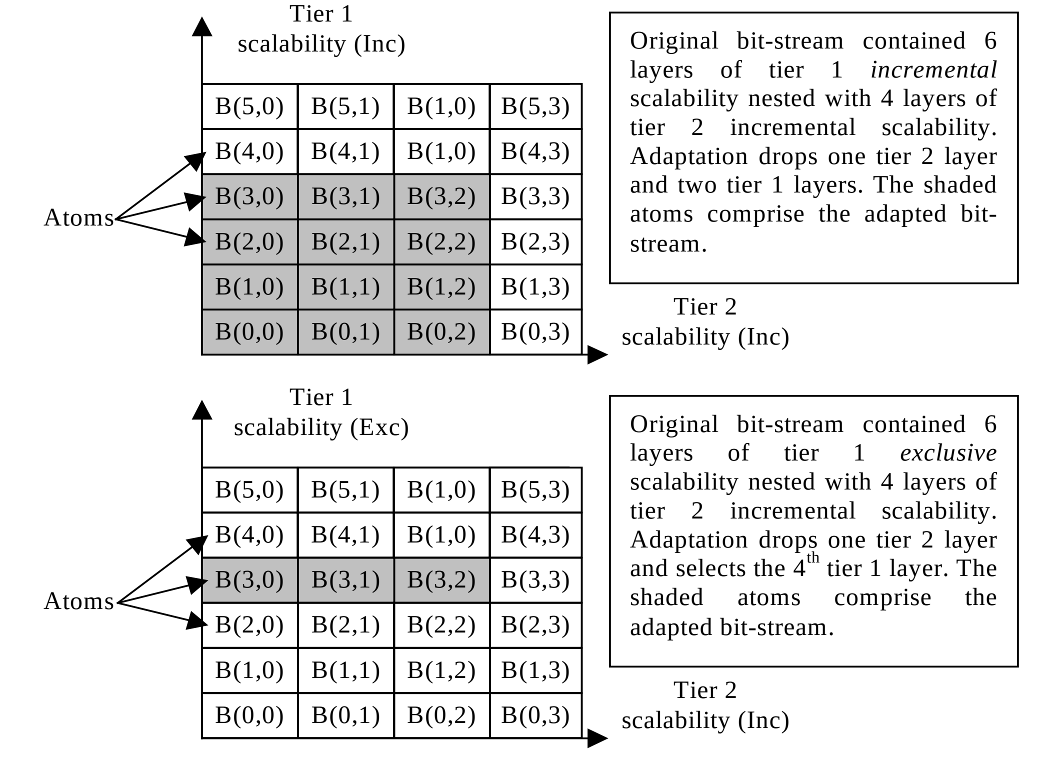
While the data cube representation has been defined above for true scalable bit-streams where successive layers in each tier are handled incrementally by a decoder, the same nested format and representation applies to the case when one or more tiers are handled exclusively. This is essentially equivalent to multi-version scalability, where multiple independent versions are maintained simultaneously in the layers of these tiers, but an eventual recipient would use only one of them. Generalizing, each tier in the meta-bit-stream format can be either *incremental* or *exclusive* in terms of scalability. The header contains a flag for each tier to denote whether the layer is multi-version or incremental. If all tiers are *exclusive*, the bit-stream is fully multi-version where each atom is an independent version. If all tiers are *incremental*, the bit-stream is truly scalable (Eg. JPEG2000). In the most general case, tiers could be mixed between *incremental* and *exclusive* scalability. In all cases however, the same logical bit-stream format and data cube representation applies.

Also note that exclusive tiers may be regarded as a special case of incremental tiers, but the transcoding is no longer efficient unless this distinction is made apparent to a transcoder by header information.

3.3 Adaptation

We next define formally an adaptation operation on the multi-tier scalable bit-stream as defined above. The multi-tier format not only allows multiple dimensions of scalability to co-exist in a bit-stream, but also enables a simplified form of adaptation. In particular, with a scalable bit-stream conformant with SSM, any adaptation is simply implemented as dropping atoms, repacking the bit-stream and updating any TOCs appropriately, while preserving the generic multi-tier structure so that it can be re-transcoded. For incremental tiers, layers can only be dropped from the outer end whereas for exclusive tiers, all but one layer is dropped.

Using our previous notation, if there are L tiers in an SSM component, then an *adaptation point* is denoted by the L-tuple (d_0, d_1,…, d_{L-1}), where d_i indicates to a decoder either d_i layers from the beginning are included in the ith tier if the ith tier is incremental, or only the d_i^{th} layer is included if the ith tier is exclusive. The adapted subset bit stream that reaches the decoder would then be given by some form of concatenation of the atoms B(j_0, j_1,…, j_{L-1}), where for tier i = 0,1,…, L–1 either j_i = 0,1,…, d_i–1 for incremental tier i, or j_i=d_i–1 for exclusive tier i. Note that if the transmitted data-stream has to be non-null, in all tiers at least one layer must be transmitted. In other words, all non-empty adapted bit-streams must contain at least one of the atoms B(z_0, z_1,…, z_{L-1}), where z_i=0 for incremental tier i.



B(5,0)	B(5,1)	B(1,0)	B(5,3)
B(4,0)	B(4,1)	B(1,0)	B(4,3)
B(3,0)	B(3,1)	B(3,2)	B(3,3)
B(2,0)	B(2,1)	B(2,2)	B(2,3)
B(1,0)	B(1,1)	B(1,2)	B(1,3)
B(0,0)	B(0,1)	B(0,2)	B(0,3)

Original bit-stream contained 6 layers of tier 1 *incremental* scalability nested with 4 layers of tier 2 incremental scalability. Adaptation drops one tier 2 layer and two tier 1 layers. The shaded atoms comprise the adapted bit-stream.

B(5,0)	B(5,1)	B(1,0)	B(5,3)
B(4,0)	B(4,1)	B(1,0)	B(4,3)
B(3,0)	B(3,1)	B(3,2)	B(3,3)
B(2,0)	B(2,1)	B(2,2)	B(2,3)
B(1,0)	B(1,1)	B(1,2)	B(1,3)
B(0,0)	B(0,1)	B(0,2)	B(0,3)

Original bit-stream contained 6 layers of tier 1 *exclusive* scalability nested with 4 layers of tier 2 incremental scalability. Adaptation drops one tier 2 layer and selects the 4th tier 1 layer. The shaded atoms comprise the adapted bit-stream.

Figure 7. Visualization of layer drops for 2-tier examples.

Using the data cube visualization, dropping layers from the end in an incremental tier is equivalent to chopping off the ends of the data cube in units of layers. Selecting a particular layer from an exclusive tier is equivalent to extracting a slice from the data cube. In general, a reduced cube from the original is transmitted after adaptation. A couple of examples for the case of 2 nested tiers are shown in Figure 7.

3.4 Causality Requirement

Because adaptation can be implemented as simple dropping of layers, an adaptation engine does not need to *decode* or *decrypt* content in order to perform adaptation. However, an encoder or an encrypter must maintain causality of the data atoms, so that a decoder or decrypter can still handle adapted content. In general, it is necessary to ensure that there are no dependencies across layers in excusive tiers, and the dependency across layers in incremental tiers is limited to being causal.

Specifically, the causality constraint for encoding ensures that for encoding data atom B(j_0, j_1, …, j_{L-1}), the encoder only uses information from atoms B(k_0, k_1, …, k_{L-1}), where for $k_i \leq j_i$ and at least one $k_i \neq j_i$ for incremental tiers i, and k_i=j_i for exclusive tiers i, within the usual limits $0 \leq j_i$, $k_i \leq l_i - 1$. This ensures that for any usable adaptation, the decoder at the receiving end can decode the content unambiguously.

The causality constraint for encryption is that the starting state of the encryption engine for atom B(j_0, j_1, …, j_{L-1}), is derived from the ending states of the encrypter for adjacent causal atoms of incremental tiers B(k_0, k_1, …, k_{L-1}), where $0 \leq j_i - k_i \leq 1$ at least one $k_i \neq j_i$ for incremental tiers i, and $k_i = j_i$ for exclusive tiers i, within the usual limits $0 \leq j_i$, $k_i \leq l_i - 1$. Progressive encryption enabling adaptation by packet truncation in encrypted domain has been considered in prior work. [26, 27]

3.5 Parcels and components

The discussion so far essentially pertained to a single coded scalable *component*. A component is a coded unit of data that may be represented in a scalable logical bit-stream-format represented by a data cube. However, in a composite bit-stream, multiple coded components can co-exist. For example, one component may be an image and a second component may be audio annotation that goes with it; both components may be packaged together in the bit-stream to provide an experience of image viewing with audio annotation. Such a combination of components is called a *parcel*.

Generally, a *parcel* is a super construct of components that essentially define adaptation boundaries. It may be comprised by multiple *independent* scalable components to provide a composite experience. The overall bit-stream consists of multiple parcels, often all of the same type. Parcels are adapted almost independently and often sequentially in an adaptation engine, with limited dependency between successive parcel adaptations. The size of a parcel is really a design choice, and may range from an entire scalable compressed file to a network transmission packet. Continuing with our previous example of image with audio annotation, both the image and audio components may constitute a parcel, but there may be multiple parcels in a composite bit-stream to produce a slide show with audio. An alternative example arises in scalable video coding where each Group of Pictures (GOP) is represented independently as a scalable component. If GOPs are to be adapted independently, then the parcel is the GOP, and contains a single component.

It is typical for different parcels in the same bit-stream to be comprised by the same components. However, the encoding structure for each component may vary from parcel to parcel, depending on characteristics for the specific content.

Finally we note that the parcel construct in SSM is not merely a unit of adaptation. The entire framework built around it is designed to fit a variety of delivery models with little or no adjustments. For example, SSM accommodates the typical streaming scenario where information for each parcel from both the descriptor side as well as the constraints side, along with the parcel bit-stream, comes into an adaptation engine separately and sequentially. Alternatively in an interactive application, parcels may be adapted and delivered randomly based on user interaction.

3.6 Bit-stream layout

Parcels, components and atoms within a component are essentially logical constructs that may exist anywhere in the bit-stream. While to make SSM adaptation viable for a given bit-stream, these constructs must exist and be defined, it is not necessary to impose syntactic restrictions on the bit-stream based on this hierarchy. An example of a bit-stream segment with two parcels, each comprised by two components is shown in Figure 8.

4. ADAPTATION VARIABLES

4.1 General

Having have seen what an adaptation operation involves, we next need to talk about ways an adaptation engine can decide on an appropriate adaptation point, without knowledge of the specifics of the media and the coding. An adaptation engine is expected to have knowledge of certain relevant scalability properties of a SSM resource through the *resource description* metadata. At the same time it expected to know the capabilities and preferences of its outbound connection through the *outbound constraints* specifications. The bridge between the two on the media creator/originator side and the receiver side of the adaptation engine is established through *adaptation variables*. The *resource description* and the *outbound constraints* speak the same language through these variables, so that an adaptation engine can decide how to drop layers to match the two sides.

If a receiver knew exactly the structure of the content it expects to receive through an adaptation engine, it could exactly specify the requested adaptation point in the engine's *outbound constraints* specifications. However, to assume that to be always the case is too restrictive. For example, if a receiver is expecting JPEG2000 images, but does not know what the dimensions and encoding parameters are for a particular image, it is not possible for it to request a specific adaptation point based on considerations of display resolution, quality, and so on. That is why, it is important to dissociate the *resource description* and the *outbound constraints* from the structure as much as possible, and *adaptation variables* allow us to achieve that.

The most important property of all adaptation variables is that they are expressed quantitatively in terms of *non-negative* (floating) numbers, referred to as *variable values,* defined over the discrete space of all possible adaptation choices. Whatever method is used to quantify the variables must be communicated to the developer of the media experiencing system. However, an adaptation engine itself does not need to know what they mean. Values have different interpretations for the media creator, the consumer, and the adaptation engines in between. To the media creator/originator, they are quantified properties based on which a content may be adapted. To a media consumer they are quantified properties to indicate its

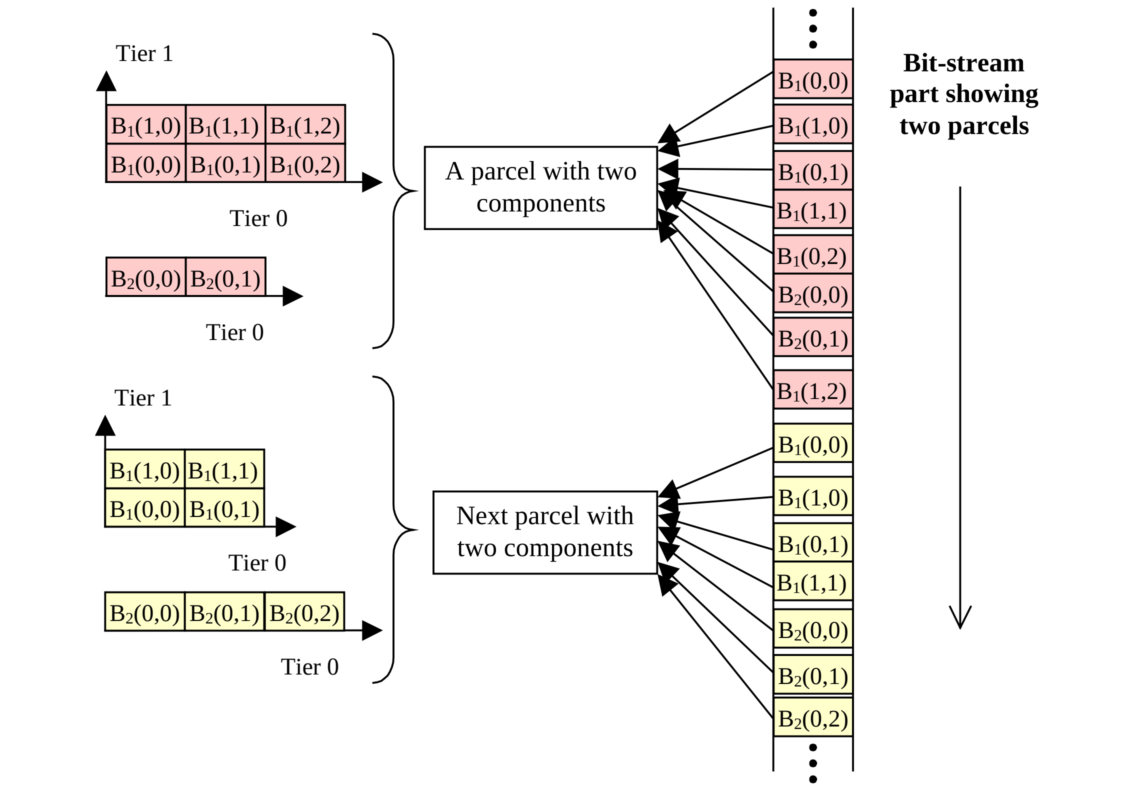


Figure 8. Bit-stream layout example for two parcels each containing two components

limitations and preferences. To an adaptation engine, they are simply numbers based on which it must decide how to drop layers and adapt an input bit-stream.

4.2 Feature variables

4.2.1 Definition

Feature variables are certain quantifiable properties relevant to the experience of a single media component or jointly for a set of media components. Features defined for a single component are called *elemental features*, while those defined over more than one component are called *product features*. Some examples of elemental feature variables are: *Codesize*, *MeanSquaredError*, *SpatialResolution*, *TemporalResolution etc.* An example of a product feature for a parcel with audio and image components is: *PerceptualRichness* which is a product feature of the adaptation points of audio and image components of a parcel, but which cannot be expressed as a function of individual features from the two components.

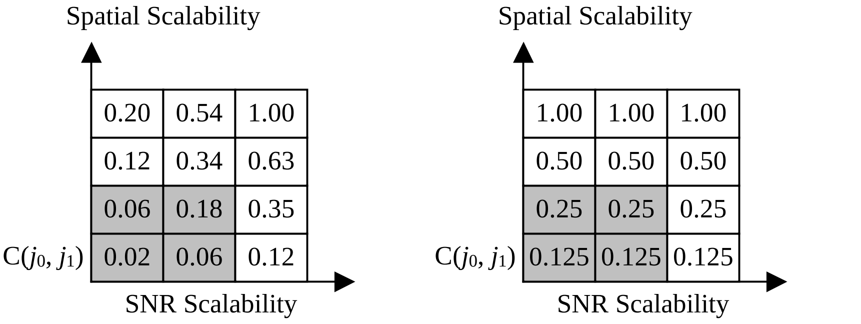
4.2.2 Identification

Each feature is associated with a name (in case of XML descriptor) or a code (in case of binary descriptor) that uniquely identifies the feature. The uniqueness is within the context of the media parcel being adapted/delivered. Thus, the feature names/codes used in the resource description metadata and the outbound constraints specifications for the same parcel of media must be consistent, but across different media-types or parcels there is no restriction on the names/codes used because when the names/codes are resolved at the adaptation engine there is no scope of a conflict.

The media creator, who provides the resource description metadata, defines features relevant to the media the way he/she chooses, and communicates their unique names/codes, meanings, and value spaces to the media experiencing system developer so that the latter can generate meaningful outbound constraints.

4.2.3 Values and distribution

The resource description metadata conveys for each feature the quantitative values the feature would have for all possible *adaptation points* of the SSM components over which it is defined. This set of values is referred to as the *feature distribution*. We first consider *elemental* features, and next consider *product* features.



0.20	0.54	1.00
0.12	0.34	0.63
0.06	0.18	0.35
0.02	0.06	0.12

(a) Feature: *Codesize*

1.00	1.00	1.00
0.50	0.50	0.50
0.25	0.25	0.25
0.125	0.125	0.125

(b) Feature: *DisplayResolution*

Figure 9. Exact Distribution Specification for JPEG2000 example

4.2.3.1 Elemental features

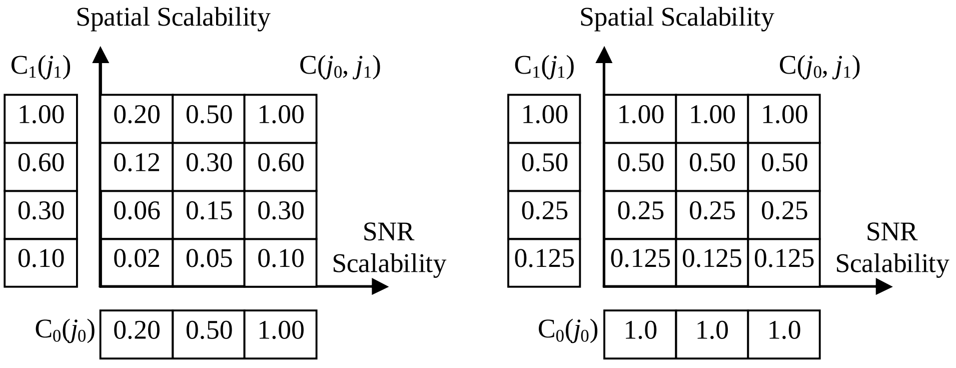
If there are L nested tiers in a component with l_i layers in the *i*th tier, it is necessary to provide a L-dimensional matrix of size $l_0 \times l_1 \times \ldots \times l_{L-1}$, whose $(j_0, j_1, \ldots, j_{L-1})$ th element denoted $C(j_0, j_1, \ldots, j_{L-1})$, for $j_0 = 0,1,\ldots, l_0-1$; $j_1 = 0,1,\ldots, l_1-1$; …; $j_i = 0,1,\ldots, l_i-1$;…; $j_{L-1} = 0,1,\ldots, l_{L-1}-1$, is a non-negative number specifying the value of the feature if $(j_0, j_1, \ldots, j_{L-1})$ is the adaptation point, along with an *empty* feature value C_φ specifying the feature value the component would have when the entire component is dropped, *i.e.* none of the layers are transmitted. The total number of values that need to be sent is therefore $1 + l_0 \times l_1 \times \ldots \times l_{L-1.}$ In practice, all these values are specified with respect to a reference feature value for convenience. In this case, the elements $C(j_0, j_1, \ldots, j_{L-1})$ multiplied by the reference value provides the true feature value for adaptation point $(j_0, j_1, \ldots, j_{L-1})$. In any case, the reference multiplied by the last fraction $C(l_0-1, l_1-1, \ldots, l_{L-1}-1)$ yields the *full feature value*, or the value the feature would have if the content were transmitted without any layer-dropping adaptation for incremental tiers and with the highest layer versions included for exclusive tiers. The same principle of multiplying with the reference applies to the empty feature value C_φ.

From the nature of features that may typically be defined, it is often the case that the distribution is either monotonic non-increasing or monotonic non-decreasing. For example, *Codesize* (rate) is a feature that is always monotonic non-decreasing. For a monotonic non-decreasing type feature, the values $C(j_0, j_1, \ldots, j_{L-1})$ would be analogous to the cumulative distribution of a multi-dimensional discrete random vector, if the reference multiplier value is the same as the full feature value. This explains the use of the term distribution.

For example, considering the first two tiers of JPEG2000 RLCP progression mode, the distribution specifications for features *Codesize* and *DisplayResolution* may look as in Figure 9. Both are non-decreasing monotonic. Here we have four spatial scalability layers nested with three SNR scalable layers each. Note that in Figure 9(b), the *DisplayResolution* attribute does not change with SNR scalable layers. As a result of transcoding, if a SNR layer and two spatial layers are dropped, the *Codesize* attribute of the transcoded bit-stream shown shaded in Figure 9 would be 0.18 times the reference *Codesize* value, while the *DisplayResolution* attribute would be 0.25 times the reference *DisplayResolution* value.

Oftentimes, it is be more convenient and less expensive in terms of overheads to express the cumulative distributions only approximately using products of one or more individual lower-dimensional *marginal* distributions. In this case, the element $C(j_0, j_1, \ldots, j_{L-1})$ is obtained approximately as $\hat{C}(j_0, j_1, \ldots, j_{L-1})$ using a product combination of marginal distributions. That is, the specification involves P lower dimensional cumulative distributions $C_i(.)$ that cover L dimensions together: $\hat{C}(j_0, j_1, \ldots, j_{L-1}) = C_0(\ ) \times C_1(\ ) \times \ldots \times C_{P-1}(\ )$. The empty feature C_φ is transmitted separately.

For the JPEG2000 example of Figure 10 the approximate specifications using two one-dimensional marginals and the eventual approximate distributions generated are shown in Figure 10. As seen in Figure 10 (b), the *DisplayResolution* feature has been represented exactly using the approximate approach, while the *Codesize* feature in Figure 10(a) is represented only approximately.



$C_1(j_1)$	$C(j_0, j_1)$		
1.00	0.20	0.50	1.00
0.60	0.12	0.30	0.60
0.30	0.06	0.15	0.30
0.10	0.02	0.05	0.10
$C_0(j_0)$	0.20	0.50	1.00

(a) Feature: *Codesize*

$C_1(j_1)$	$C(j_0, j_1)$		
1.00	1.00	1.00	1.00
0.50	0.50	0.50	0.50
0.25	0.25	0.25	0.25
0.125	0.125	0.125	0.125
$C_0(j_0)$	1.0	1.0	1.0

(b) Feature: *DisplayResolution*

Figure 10. Approximate Distribution Specification for JPEG2000 example

4.2.3.2 Product features:

The resource description metadata conveys for each product feature the quantitative values the product feature would have for all possible joint *adaptation points* of the SSM components involved in the product feature. This set of non-negative values is the product *feature distribution*. If there are C components involved in the product feature, with L^c nested tiers in the *c*th component (c = 0,1,…,C-1) it is necessary to provide 2^C-1 non-empty distributions corresponding to the case where at least one component is non-empty (included), along with an all empty feature value C_φ corresponding to the case when all components are empty. Among the non-empty distributions there is one corresponding to the case where all components are included. In this case, the distribution specifies a $(L^0+L^1+\ldots+L^{C-1})$-dimensional matrix with number of elements equal to the product $\prod (l^c_i)$ over c = 0,1,…,C–1 and i = 0, 1, …, L^c–1, where l^c_i is the number of layers in tier i of component c. There are other 2^C-2 non-empty distributions corresponding to the cases when one or more components but not all are empty. Any such partial empty distribution is specified as a reduced dimensional distribution over the non-empty components. The total number of values that need to be sent comprising all the non-empty distributions and the empty value is the product $\prod(1 + l^c_0 \times l^c_1 \times \ldots \times l^c_{L^c-1})$ over all c.

Each non-empty distribution can be individually specified using a product of marginals as in the elemental feature case.

4.3 Component variables

4.3.1 Definition

Just as features have unique names/codes so do components within the context of a parcel. The media creator not only conveys the names/codes of feature variables, their meanings and value spaces to the experiencing system developer, but also the names/codes of components. Based on the component name/code, certain variables are defined by default. These are called component variables.

4.3.2 Indicators and values

The first variable family defined by default is the *inclusion indicator*. This variable has a value of 1 for all non-empty adaptation points, and zero only if all the atoms are dropped. A component is assumed to be included if at least one of its atoms is included. This variable can be used to specify complex constraints based on inclusion or exclusion of whole components, for e.g. if a certain component is included another one must be included, and so on.

The second family of indicators is called *layers in tier indicators*, which convey the number of layers in the adaptation point for a specified tier index parameter. Thus if the adaptation point is $(j_0, j_1, \ldots, j_{L-1})$, then the value of this variable corresponding to tier i is j_i.

The third family of indicators is called the *current number of layers in tier indicators,* which conveys a constant whose value is the total number of layers currently in the bit-stream for a specified tier index parameter.

The fourth family called the *original number of layers in tier indicators*, which conveys a constant whose value is the original number of layers in the bit-stream for a specified tier index parameter, before any adaptation step.

4.4 Combination variables

4.4.1 Definition

Sometimes the media creator can define combination variables in the resource description metadata, which are essentially mathematical real and/or Boolean expressions involving feature variables, component variables or other combination variables from a variety of components. The combination variables are conveyed to the experiencing system developer in the same way as are feature variables, and serve as a short cut for the outbound constraints specifications.

One example of combination variables is *TotalCodesize*, which may be defined as the sum of the *Codesize* features for individual components in a parcel. Another example, involving the component inclusion indicator variables is a Boolean expression that indicates if component_1 is included, component_2 must be included (x=>y is equivalent to x´+ y)

4.4.2 Identification

Combination variable are identified in the same way as feature variables, i.e. with unique names or codes. The resource description metadata provides a name/code for each combination variable as they are defined.

4.4.3 Expression specification and evaluation

The mathematical expression for each combination variable is specified in the resource description metadata by means of an ordered list of numeric constants, adaptation variables and operators that must be pushed into an expression stack for evaluation of the expression. Variables pushed into the stack can be feature variables, component variables or previously defined combination variables, each identified by its unique name/code. Operators pushed into the stack can be either unary or binary operators.

Evaluation of an expression at an adaptation engine for a given set of adaptation points corresponding to components of a parcel is done as follows. When a constant is pushed its numeric value is pushed into the stack as a real numeric element. When a variable is pushed, the numeric value of the variable for the given set of adaptation points is evaluated, and pushed into the stack as a numeric element. When a unary operator is pushed into the stack, the current top operator element as well as the next top stack element, which must be a numeric one, are popped out immediately. The operator operates on the numeric operand, and the result is pushed back into the stack as a numeric element. When a binary operand is pushed into the stack, the current top operator element and the two next top stack elements, both of which must be numeric, are popped out immediately. The binary operator operates on the numeric operands, and the result is pushed back into the stack as a numeric element. When all the elements in the expression ordered list has been processed, the topmost stack element yields the value of the expression.

A small set of useful real and Boolean operators is allowed for forming expressions. When real and Boolean operators and operands are mixed, the following conventions are used to make the necessary transformations between the two domains: a Boolean 0 has real value 0.0, a Boolean 1 has real value 1.0, any real non-zero value has Boolean value 1, and a real zero 0.0 has Boolean value 0.

4.5 Reserved adaptation variables

In the discussion so far, we have been talking about so called *custom* or content specific adaptation variables that can be defined and named/coded at will by content creators so that the experiencing system at the receiving end can issue appropriate outbound constraints based on them. It is worth mentioning however that there is value in standardizing the identifying names/codes and the quantification method for certain variables with universal meaning across different types of content. These variables are called *reserved adaptation variables,* because their identifying names/codes are effectively reserved and disallowed for use as custom adaptation variables. Examples of adaptation variables that could be standardized in name/code and quantification method are: *Bandwidth*, *NumerOfSpeakers*, *DisplayResolution*, *ProcessorSpeed* etc. Use of reserved variables enable creation of *content independent profiles* for terminals, that allow adaptation say at an edge server, without an explicit request from a receiving terminal in the form of outbound constraints, for every type of content that is received. In other words, it enables passive reception of various types of content by uploading a stationary content-independent profile involving reserved variables to an adaptation engine once and for all.

Reserved adaptation variables may be either feature variables or combination variables defined differently for different types of content. But the eventual meaning and the value space of the adaptation variable should be the same for different content types.

Note that since the SSM framework per se does not change depending on whether an adaptation variable is custom or reserved, we do not dwell on this issue further.

5. RESOURCE DESCRIPTION METADATA

Because the resource description metadata has mostly been covered already in the discussion so far, here we summarize the key points, and mention the additional features. This metadata originates from the media creator and contains a full description of the bit-stream that enables an adaptation engine to decide how to drop layers.

5.1 Scalability structure

The metadata specifies the complete hierarchical model of the bit-stream with parcels, components, and atoms, and where the atoms lie in the bit-stream. For each parcel it defines a set of elemental and product feature variables and specifies their distributions, as well as a set of combination variables that apply locally within the parcel. It also defines global combination variables that apply to all parcels.

5.2 Limit constraints from content creator

Besides, defining adaptation variables, the metadata also includes constraints that are to be enforced by an adaptation engine. These are directives from the content creator to restrain the adaptation choices, and apply either locally to a parcel or globally to all parcels. The type of constraints specified here is referred to as *limit constraints*, and would be elaborated in the next section. The adaptation engine combines the constraints specified by the content creator in the resource description metadata with those specified by the receiver in the outbound constraints specifications to obtain the full set of constraints that need to be satisfied by an adaptation point.

5.3 Resource edit information

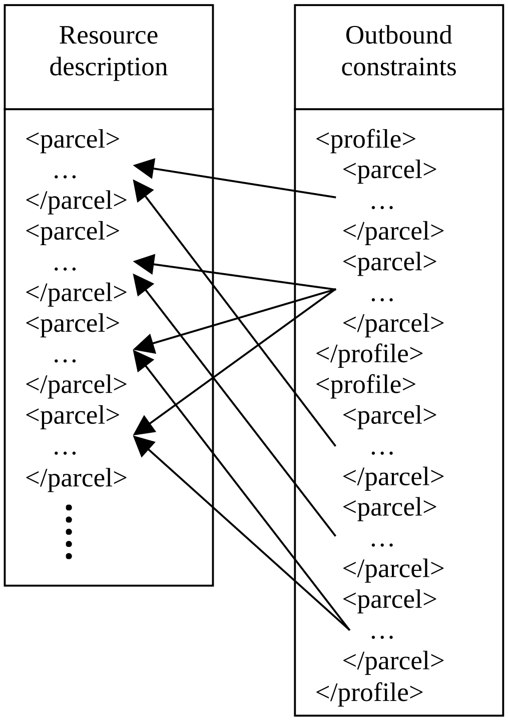
The resource description metadata contains information pertaining to editing of the resource bit-stream based on adaptation decisions made for each parcel. For example, it may often be necessary to modify information such as number of layers included and so on in the bit-stream, after adaptation. The metadata specifies for each parcel exactly where in the bit-stream a certain number of bits have to be replaced after the decisions have been made and the adaptation has been conducted, how many bits the replaced value spans, as well as what the modified value is. The modified value is given by a stack expression as in Sec. 4.4.3 . The output length in bits can be specified through a constant or through a feature variable. Note that this protocol actually allows a wide range of bit-stream modifications based on adaptation decisions. Even when it is not possible to have expressions to evaluate the modified value, the content creator can always define feature variables, one for the content and another for the length to denote what the correct bit-stream should be for each adaptation possibility.

Sometimes in a compressed bit-stream there may be pointers that specify offsets to or locations of other parts of the bit-stream. When SSM atoms are dropped as part of the adaptation process it is likely that this offset/location information will become invalid. Therefore, in order to keep the adapted bit-stream consistent and decodable it is necessary to modify the links as and when atoms are dropped. Because the adaptation engine that must modify the bit-stream, it is necessary to provide this offset/location information in the resource description metadata. The resource descriptor allows specifying locations in the bit-stream where offsets occur, but does not specify exactly what these values are.

6. OUTBOUND CONSTRAINTS SPECIFICATIONS

6.1 Parcel mapping for multiple profiles

The outbound constraints specification provides a specific request to an adaptation engine to adapt the content before delivery. In general, the outbound constraints can be specified for multiple recipient profiles. For each recipient profile, a set of constraints can be specified corresponding to each parcel. However, the number of parcels specified can be less than the number of parcels in the content bit-stream. In that case, there is a sequential one-to-one mapping between the parcels in the outbound constraints specifications and the parcels in the resource description metadata, for as many parcels are specified in the constraints specification. The rest of the parcels as specified in the resource description are adapted based on the last parcel specified in the constraints specification. This principle is shown in Figure 11 assuming XML to be



This example shows two profiles in the outbound constraints XML, with two and three parcel specifications in them respectively.

Figure 11. Parcel mapping between resource description and outbound constraints

the mode of communication. As shown in the figure, the outbound constraints XML can have multiple receiver profiles defined. The parcel-mapping rule applies individually to each profile. We will cover multiple profiles in the next section.

6.2 Adaptation constraints

The adaptation constraints for each parcel can be either *adaptation variable driven* or *structure driven*. The former, which is based on defined adaptation variables, is a more interesting case because it keeps the content creator and the receiver sides more independent of each other, and integrates decision making in an adaptation engine. However, if the receiver knew exactly the structure of the content, it could specify the adaptation points exactly in its constraint specification. This is called *structure driven* adaptation, and in this case, there is no decision making involved on part of the adaptation engine. Furthermore, in structure driven adaptation the framework allows arbitrary adaptations not limited by the dependency structure. That is, an arbitrary subset of atoms from the data cube can be selected and included in the adapted resource bit-stream.

The *adaptation variable driven* constraints consist of a set of *limit constraints*, followed by an optional *optimization constraint*. Each constraint, limit or optimization, is specified in terms of definable functions of adaptation variables, called an *adaptation metric*. The metric is defined using a stack expression involving adaptation variables corresponding to the parcel, as defined in the resource description metadata. The stack expression methodology used is the same as the one used in Sec. 4.4.3 .

Limit constraints are specified as *lower and/or upper* limits of supportable value(s) for outbound connections for a defined metric. When both are specified we have effectively provided a range, and when both limits are the same, we have imposed an equality constraint. An example of a limit constraint is: *Codesize*/latency < 300 KB/s. Here *Codesize* is an attribute, but 1/latency is specified in outbound constraints as a multiplier. Overall, this indicates a bandwidth restriction on received media. Another example is: display resolution<800 diagonal pixels.

An optimization constraint specifies a requested *minimization* or *maximization* of a defined metric. An example of such a constraint is in rate-distortion optimization, where a metric such as *MeanSquaredError* + λ.*Codesize* is minimized. Here the *Codesize* variable corresponds to rate (R), while the *MeanSquaredError* variable corresponds to distortion (D). Encrypted domain transcoding based on packet truncation minimizing D+λ.R has been covered in prior art.[26, 27]

Sometimes it may be necessary to maintain consistency of adaptation across parcels. For example, if each parcel is a GOP from a video sequence, we do not want to have different spatial resolutions for each of them. The outbound constraints specification incorporates a method to preserve limited dependencies across parcels by allowing references to adaptation variables from the previous parcel for the particular decision made for the previous parcel. However, because the outbound constraint part for the next parcel may not be known at the adaptation time for the current parcel, it is necessary to specify explicitly a list of adaptation variables to be remembered by the engine for use in adaptation of the next parcel. The values of these variables are computed based on the adaptation points decided for the current parcel. It is up to the resource description generator to provide relevant information to the outbound constraints generator to make sure that all *previous* references are correctly resolved.

6.3 Adaptation engine operation based on constraints

The information contained in the resource descriptor and the outbound constraints specification is all that an adaptation engine needs to decide how to adapt each parcel. For adaptation variable driven adaptation, when both a set of limit constraints and an optimization constraint are specified, the adaptation engine seeks the minimum or the maximum of the optimization metric, within the allowable adaptation point space carved by the limit constraints. For each parcel, the engine can in principle perform an exhaustive search over the joint space of all possible adaptation choices for all the constituent components. For each candidate decision point, the engine first evaluates the limit constraint metrics to see if they are satisfied. Limit constraints in both the outbound constraints specification (from the receiver side) as well as the resource descriptor (from the content creator side) are considered. If all limit constraints are satisfied, the single optimization metric is evaluated. The optimum decision point is one that not only satisfies all the limit constraints but also maximizes or minimizes the optimization metric over all cases also satisfying the limit constraints.

When only limit constraints are specified but no optimization constraint, there is no unique solution. Normally, the adaptation engine assumes a default optimization constraint. Note however, that the actual decision making algorithm is not a part of the proposed framework. The SSM framework involves ways to convey all the information needed to make decisions, and how the adaptation is to be conducted once decisions have been made. However, there is no restriction on the decision making process.

7. MULTI-STEP MULTI-PROFILE ADAPTATION ARCHITECTURE

If the outbound constraints specification contains exactly one receiver profile, a single adapted version of each SSM component allowing unambiguous decryption/decoding is generated and transmitted. This is typically the case when an adaptation engine directly connects to the eventual recipient.

There is however another scenario where an adaptation engine must adapt and deliver a bit-stream containing a combination of several profiles for several recipients, to be eventually extracted by other downstream adaptation engines. In this situation, the former adaptation engine could send the bounding box containing the different versions for different receiver profiles, for both exclusive and incremental tiers, which allows re-adaptation to lower terminal versions downstream. If all atoms within the bounding box including those that are not used by any of the profiles are sent, it is referred to as *mid-stream* adaptation type. Alternatively, atoms unused by all profiles may be dropped from the bit-stream, thereby saving bandwidth. This type of adaptation is referred to as *terminal*. The outbound constraints specification also conveys the type of adaptation desired. Further, when atoms within the bounding box of profiles are dropped as a consequence of *terminal* adaptation, this information is conveyed in the adapted resource description metadata for use by subsequent adaptation engines.

Consider Figure 12 that shows a delivery chain to two recipients through two adaptation engines, along with the example of a SSM component. Both recipients convey their profiles in their respective outbound constraints specifications to adaptation engine B. Engine B aggregates the two profiles into single outbound constraints specification and forwards it upstream to adaptation engine A, requesting either *mid-stream* or *terminal* adaptation types. The engine A actually decides on the adaptation points for each profile, and packs the bit-stream using either mid-stream or terminal adaptation type as requested by engine B, and forwards it to B. On receipt of the bit-stream, B applies the same profiles to the input stream but individually for each of the two output streams for the two recipients, performing *terminal* type adaptation

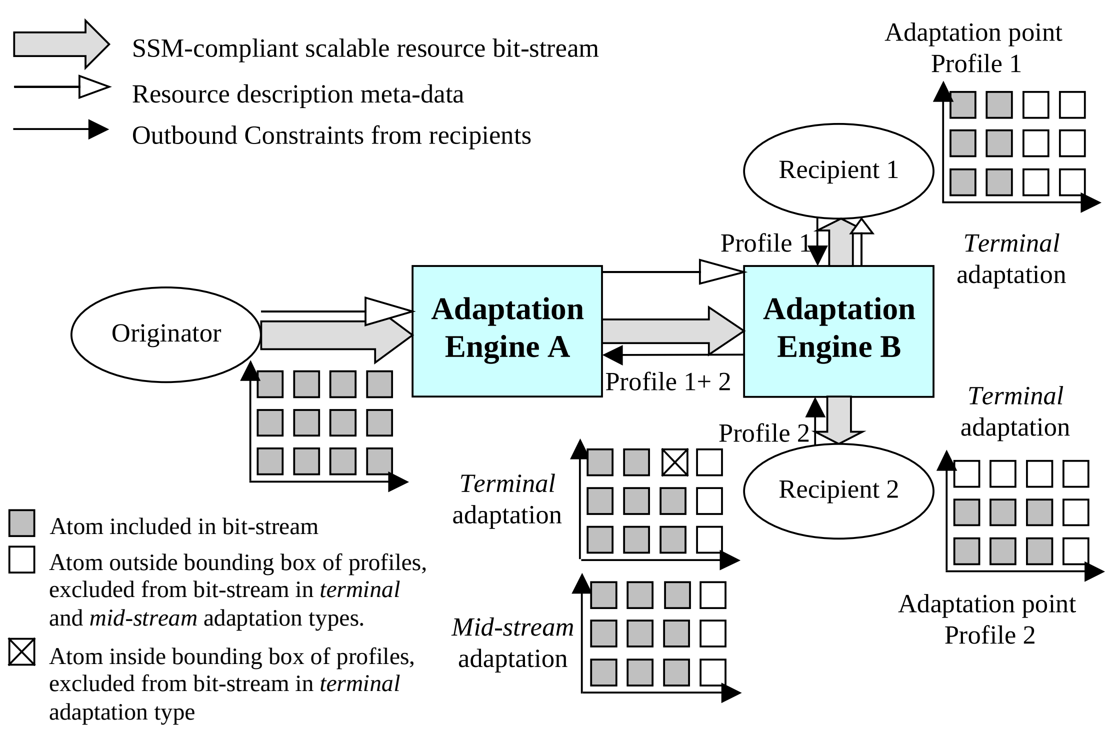

SSM-compliant scalable resource bit-stream
Resource description meta-data
Outbound Constraints from recipients
Adaptation point
Profile 1
Recipient 1
Profile 1
Terminal
adaptation
Originator
Adaptation
Engine A
Adaptation
Engine B
Profile 1+ 2
Profile 2
Terminal
adaptation
Recipient 2
Terminal
adaptation
Mid-stream
adaptation
Adaptation point
Profile 2
Atom included in bit-stream
Atom outside bounding box of profiles, excluded from bit-stream in terminal and mid-stream adaptation types.
Atom inside bounding box of profiles, excluded from bit-stream in terminal adaptation type

Figure 12. Multi-step multi-profile adaptation minimizing end-to-end redundancy

in this case. Since the bit-stream transmitted by A already contains the optimal decisions for both recipients, the same optimization over a reduced set of possibilities would yield the same solutions individually. Hence, engine B is able to extract the bit-stream needed for each recipient, irrespective of the adaptation type *mid-stream* or *terminal*, from its input bit-stream, and deliver to the recipients. The only difference between the two adaptation types used by engine A is that the *mid-stream* type leaves more flexibility for engine B to create further versions other than those needed by the two recipients, while the *terminal* type attempts to minimize the end-to-end redundancy of transmission bandwidth.

An alternative architecture is based on the assumption that the resource descriptor is available at both adaptation engines before the resource is transmitted through either engine. In this case, the engine B could do the decision making itself for the two recipients, and send only a *structure-driven* adaptation request to engine A. The required bit-streams are packaged in the same way as in the previous case by engine A and sent back to B, which then uses single-profile *structure-driven terminal* adaptations to extract the appropriate bit-stream for each for delivery to each recipient.

Note that the SSM-enabled multi-step multi-profile architecture above, supports end-to-end adaptation and delivery of scalable content in a fully codec non-specific manner, while minimizing overall redundancy in bandwidth.

8. CONCLUSION

Use of scalable media for content-agnostic adaptation is well known in the literature. These adaptation engines do not need to decrypt or decode compressed content in order to adapt it into a form appropriate for lower bandwidth/resolution etc. The underlying assumption behind the adaptation operation is that an engine understands the format in which the data is represented in, even though it does not need to know what the data actually is. However, the requirement on the structure of the content is still rigid in these approaches, because codec specific components are still needed for different types of media content. That is, an adaptation engine for images compressed in a particular way, say JPEG2000, would still be different from an adaptation engine for a certain kind of interactive content encoded in an entirely different way.

The SSM proposal for MPEG-21 DIA[25] described in this paper advances the level of abstraction to develop a flexible methodology for universal adaptation of scalable content, where the adaptation operation is generic enough to be applicable to any type of media having any type of encoding, and does not use any codec-specific code or stylesheets at the adaptation engine. The adaptation engine just needs to be told what the structure of the particular content that goes through it is, and how this content is to be adapted to achieve the desired transcoding operation. This meta-data information is conveyed by the media creator/originator to the engine. The specific requirements for a recipient are conveyed to the adaptation engine through the outbound constraints specification. With this framework, different adaptation infrastructures are no longer needed for different types of scalable media. For media that is non-standard or for media that do not exist today but would evolve in the future, as long as they conform to the lose bit-stream restrictions that the universal adaptation engine understands, it still becomes possible to adapt it appropriately using SSM adaptation engines.

The main features of the SSM framework are:

- Modeling of scalability in a very generic way that is not too restrictive, yet not so unrestrictive that the number of adaptation choices becomes unacceptably large from the perspective of metadata compactness. (Note that unrestricted adaptation is still supported by *structure driven* adaptation).
- Fully content independent operation, making it easy for new content providers to use. The adaptation infrastructure does not need to change in any way.
- Provides a complete end-to-end delivery solution to multiple recipients minimizing end-to-end redundancy, possibly in encrypted domain.
- Extensible to new types of content with new types of scalability.
- Provides enormous flexibility for adaptation both from the recipient and content creator points of view, for a variety of delivery scenarios.
- Caters to both incremental scalable, exclusive scalable as well as hybrid scalable bit-streams under a common framework.

REFERENCES

1. David Taubman, “High Performance scalable image compression with EBCOT,” *IEEE Transactions on Image Processing*, vol. 9, no. 7, July 2000, pp. 1158-70.
2. Amir Said and William A. Pearlman, “A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 243-250, June 1996.
3. Debargha Mukherjee, “Vector set partitioning and successive refinement VQ for wavelet image and video compression,” *PhD thesis*, University of California, Santa Barbara, Aug 1999.
4. J. M. Shapiro, “Embedded image coding using zerotrees of wavelet coefficients,” *IEEE Trans. Signal Processing*, vol. 41, no. 12, Dec. 1993.
5. David S. Taubman and M. W. Marcellin, “JPEG2000: Image Compression Fundamentals, Standards and Practice,” Kluwer Academic Publishers, 2002.
6. B. G. Haskell, A. Puri, A. N. Netravali, “Digital Video: An Introduction to MPEG-2,” New York: Chapman & Hall, Sept 1996.
7. Weiping Li, “Overview of Fine Granularity Scalability in MPEG-4 Video Standard,” IEEE Trans. Circuits and Systems for Video Technology, March 2001, vol. 11, No. 3, pp. 301-317.
8. (MPEG-4) Information technology – Coding of audio-visual objects – Part 2: Visual, ISO/IEC 14496-2-2001.
9. (MPEG-4) Information technology – Coding of audio-visual objects – Part 3: Audio, ISO/IEC 14496-3-2001.
10. Video Coding for Low Bitrate Communication, ITU-T Recommendation H.263, Nov. 1995.
11. Video Coding for Low Bitrate Communication, ITU-T SG16/Q.15 H.26L Project, Feb. 2000.
12. J. Xu, Z. Xiong, S. Li, and Y.-Q. Zhang, “3-D embedded subband coding with optimal truncation (3-D ESCOT),” J. Applied and Computational Harmonic Analysis: Special Issue on Wavelet Applications in Engineering, vol. 10, pp. 290-315, May 2001.
13. Shih-Ta Hsiang and John W. Woods, "Embedded video coding using motion compensated 3-D subband/wavelet filter bank", Packet Video Workshop, Sardinia, Italy, May 2000.
14. Shih-Ta Hsiang and John W. Woods, "Embedded video coding using invertible motion compensated 3-D subband/wavelet filter bank," Signal Processing: Image Communications, Vol, pp. 705-724, May 2001.
15. Shih-Ta Hsiang, “Highly Scalable Subband/Wavelet Image and Video Coding,” Ph.D. Thesis, Rensselaer Polytechnic Institute, Troy, New York, May 2002.

16. J. W. Woods and Peisong Chen, “Improved MC-EZBC with quarter-pixel motion vectors,” ISO/IEC JTC1/SC29/WG11, MPEG2002/M8366.
17. J. W. Woods, Peisong Chen, and Shih-Ta Hsiang, “Exploration experimental results and software,” ISO/IEC JTC1/SC29/WG11, MPEG2002/M8524.
18. MPEG-21 Digital Item Adaptation WD (v3.0), ISO/IEC JTC1/SC29/WG11/N5178.
19. MPEG-21 Requirements on Digital Item Adaptation, ISO/IEC JTC1/SC29/WG11/N4684.
20. Sylvain Devillers, Myriam Amielh, Thierry Planterose, **“**Bitstream Syntax Description Language (BSDL), Response to the Call for Proposals on MPEG-21 DIA”, ISO/IEC JTC1/SC29/WG11, MPEG2002/M8273.
21. Sylvain Devillers, “BSDL architecture for multi-step adaptation”, ISO/IEC JTC1/SC29/WG11, MPEG2002/M8523.
22. Hermann Hellwagner, Jörg Heuer, Andreas Hutter, Harald Kosch, Gabriel Panis, Christian Timmerer, “Adaptation architecture using BSDL” ISO/IEC JTC1/SC29/WG11, MPEG2002/M8290.
23. Jörg Heuer, Andreas Hutter, Gabriel Panis, Hermann Hellwagner, Harald Kosch, Christian Timmerer (Univ. Klagenfurt), Proposal of a Generic Bitstream Syntax Description Language (g-BSDL), ISO/IEC JTC1/SC29/WG11, MPEG2002/M8291.
24. Debargha Mukherjee, Amir Said, “Structured Content Independent Scalable Meta-formats (SCISM) for Media Type Agnostic Transcoding: Response to CfP on DIA / MPEG-21,” ISO/IEC JTC1/SC29/WG11, MPEG2002/M8689.
25. Debargha Mukherjee, Geraldine Kuo, Amir Said, Giordano Beretta, Sam Liu, Shih-ta Hsiang, “Proposals for end-to-end Digital Item Adaptation using Structured Scalable Meta-formats (SSM),” ISO/IEC JTC1/SC29/WG11, MPEG2002/M8898.
26. S. J. Wee and J. G. Apostolopoulos, “Secure scalable video streaming for wireless networks,” Proc. IEEE Int. Conference on Acoustics, Speech and Signal Processing, Salt Lake City, Utah, May 2001.
27. S. J. Wee and J. G. Apostolopoulos, “Secure scalable streaming enabling transcoding without decryption,” Proc. IEEE Int. Conference on Image Processing, Thessaloniki, Greece, October 2001, vol. 1, pp. 437-40.
28. D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang, J. M. Peha, “Streaming Media over the Internet: Approaches and Directions,” IEEE Trans. Circuits and Systems for Video Technology, March 2001, vol. 11, No. 3, pp. 282-300.