

# Structured Content Independent Scalable Meta-formats (SCISM) for Media Type Agnostic Transcoding

*Debargha Mukherjee and Amir Said*

Email: {debargha, said}@hpl.hp.com  
Imaging Systems Laboratory, HP Laboratories  
1501 Page Mill Road, Palo Alto, CA 94304, USA.

## **Abstract**

*This paper develops a universal meta-format (SCISM) and methodology for representation and transcoding of arbitrary scalable content. The abstractions in the meta-format are generic enough to be applicable to any type of media with only a loose restriction on the encoding structure. Scalable bit-streams are naturally organized in a manner such that representation in compliance with the meta-format is straightforward. By interpreting the generic meta-format, a universal transcoder can transcode the content appropriately to suit the needs and preferences of recipients, without knowledge of the specifics of the content, its encoding and/or encryption. The transcoder just needs to be told what the structure of the particular content that goes through it is, and how this content is to be transcoded to achieve the desired transcoding operation. This is meta-data information, which can either be part of the header of the media itself, or can be conveyed to a transcoder separately for an entire class of content. The media meta-data, along with a standardized specification of the capabilities and preferences conveyed by a media destination are all that a transcoder needs to adapt format compliant scalable content appropriately. With universal transcoders, different transcoding infrastructures are no longer needed for different types of scalable media.*

## **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 spread 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 will always be severely 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 too much redundancy leading to wastage of bandwidth and storage. This is especially so, when the media creator wishes to provide a wide range of choices 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.

It is important to realize however, that this case is actually a fully redundant special case of true scalability to be described next, and consequently the framework proposed in the paper still applies.

### **1.2. Scalable Bit-streams**

In order to provide a more elegant solution, 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 and frequency band. In the future, with the evolution of newer, richer and more interactive types of media, there will be newer types of scalability that we do not even know yet.

A scalable bit-stream does not always have a single type of scalability. In fact, different types of scalability often co-exist in a multi-dimensional structure, so as to provide a wide range of adaptation choices. As an example, the new JPEG2000 [1] scalable standard for still images endeavors to combine quality scalability and resolution scalability in a common format, to enable distribution and viewing over a 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 model 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 a seamless ease of use of the concerned media.

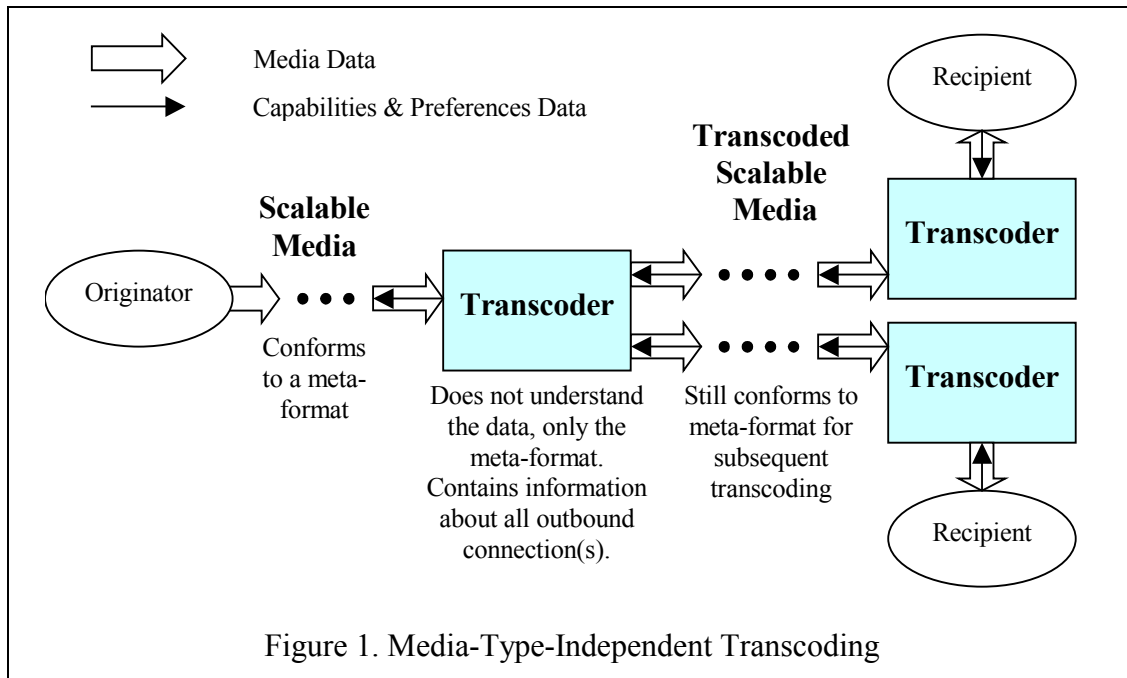
For example, even though the JPEG2000 format itself is very powerful, the lack of a complete infrastructure that supports appropriate transcoding of JPEG2000 content and delivery to a heterogeneous recipient base has severely restricted the usability of its scalable features. In recent years, a great deal of attention has been focused on delivering streaming video over the Internet or wireless [2], [3], [4]. In order to reach heterogeneous recipients in a dynamic transmission environment, video standards of MPEG-X [5] (mostly MPEG-4 [6]) and H.26X [7], [8] families incorporate various forms of scalability. Although rudimentary 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.

### **1.4. Need for Generic Infrastructures**

Any infrastructure, is expensive to deploy, and requires significant financial commitments from the patron companies or patron consortia. In order to guarantee constancy of the format it would also be desirable that the format the content is represented in be standardized. On the other hand, standards take several years to come into effect, typically much longer than is commensurate with the normal pace of change in the multimedia industry. As new types of media beyond traditional images, video and audio evolve; it would become more and more difficult to expect standards to support their representation.

Furthermore, even if efficient scalable formats evolve for every new type of media, the inevitable difference in the structure of the content would necessitate use of different infrastructures 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 agnostic of the type of media being transcoded. Such universal transcoding infrastructures only need to be deployed once to support transcoding 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. Motivation for this work

In order to enable universal content adaptation infrastructures for scalable media, we introduce a new paradigm for scalable media representation that standardizes a common meta-format for all scalable media types rather than a single format for a specific media type (such as JPEG2000 for images). The meta-format is called **SCISM** (Structured Content Independent Scalable Meta-format). Media adaptation and delivery infrastructures based on interpretation of such meta-formats would be truly universal and cost-effective because they would support a wide variety of media types rather than a single type. Such universal transcoding infrastructures can support transcoding and delivery of all types of scalable media, as long as they conform to certain loose restrictions imposed by the meta-format.

## 2. Media-Type-Independent Transcoding

Noting that the eventual purpose of scalable representations is seamless, flexible delivery to a heterogeneous recipient base, it is not only enough to obtain a compact scalable representation. It is also necessary to develop transcoders in the network that convert content suitable for a higher set of capabilities and preferences to a lower one. Because it is impractical to create new transcoders for every type of media that currently exists and would evolve in the future, we need to dissociate the media-type from the transcoding operation. This is referred to as *media-type agnostic transcoding*.

## 2.1. **Universal Meta-format**

In order to enable *media-type agnostic transcoding*, it is necessary to develop a *media-type agnostic meta-format* that compressed scalable bit streams must conform to, and which must be understood by all intermediate transcoding nodes within a delivery architecture. This paper is essentially involved with developing such a universal meta-format for all scalable media, called SCISM (Structured Content Independent Scalable Meta-format), and an associated methodology that allows network transcoders to generate content suitable for a variety of outbound bandwidths, display capabilities, user preferences and so on, from the incoming content. Because transcoders based on SCISM are truly media-type- and content-independent, they can transcode 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.), as long as the content bit-stream conforms to the loose SCISM meta-format. 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.

## 2.2. **SCISM based Delivery Model**

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 transcoders 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 company opened up its technology for another company to create the experiencing system for, as part of a partnership; or the media format is a SCISM-compliant open standard known to all. Irrespective of the actual encoding however, the scalable media data at a higher level is conformant with SCISM, which all intermediate transcoders understand. Transcoders receive SCISM compliant scalable content, and deliver transcoded content over multiple outbound connections. All content after transcoding is also SCISM meta-format compliant so that it can be re-transcoded at a subsequent stage of delivery.

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

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 transcode appropriately. The transcoding operation is based purely on an interpretation of the meta-format, 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 transcoders 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

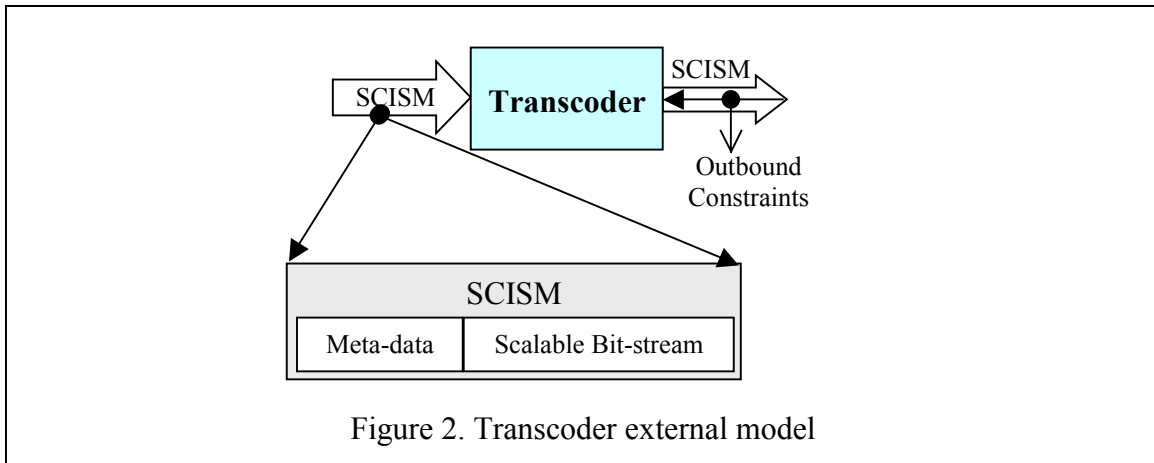


Figure 2. Transcoder external model

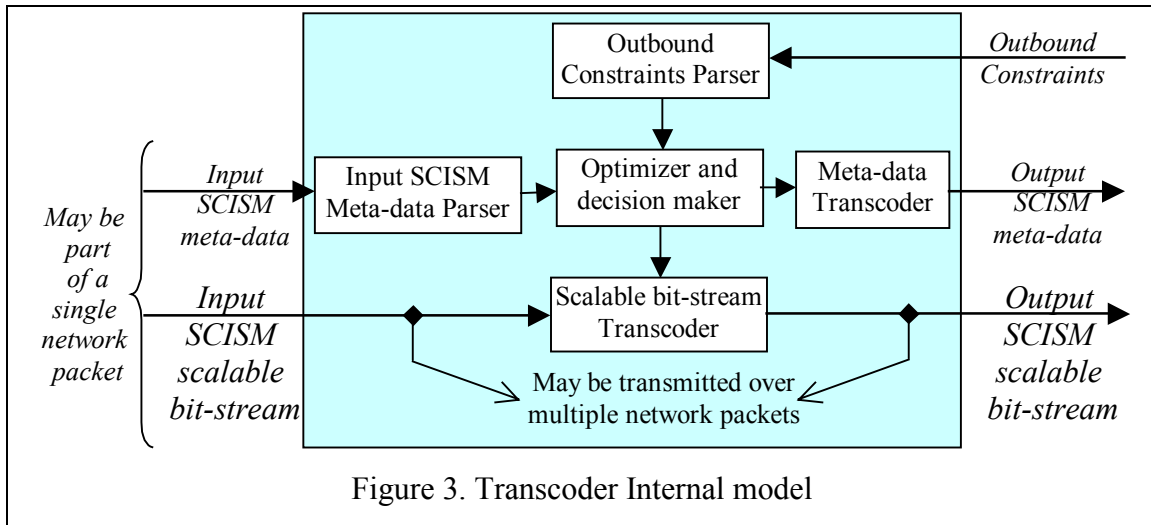
servers through which scalable content is transcoded and routed; or Edge servers that connect directly to eventual recipients. Also the generic delivery model considered can collapse to as simple as a client-server delivery system where a client requests content from a media server with specified capabilities and preferences, and gets appropriately transcoded content directly from it. In this case, the functional transcoder would be part of the Media server itself.

### 2.3. *Isolated Transcoder Model*

From the generic model, we can isolate a single functional transcoder whose external model is shown in Figure 2, to understand the scope of the work. The transcoder receives a SCISM compliant piece of media, which it must transcode appropriately and forward in a SCISM compliant manner to an eventual consumer or another transcoder. It also receives another input, a specification of the capabilities and preferences of its output connection referred to as *outbound constraints* for the transcoder based on which it must adapt the content. The way the capabilities and preferences of an outbound connection, is conveyed to a transcoder, depends on the specifics of the delivery architecture. The transcoder may either be directly connected to a recipient who conveys its capabilities to it, or aggregated capabilities of all downstream clients may be conveyed to a transcoder by some network aggregation mechanism. Based on the information contained in the SCISM meta-format and the *outbound constraints*, a transcoder performs the transcoding operation on the input stream and delivers content to its outbound connections.

The actual SCISM meta-format consists of two parts: the first part contains meta-data that describes the structure and properties of the content in a non-media-type specific manner, and the second part that contains the actual bit-stream for encoded scalable data. The SCISM meta-data in the input media and the outbound constraint (Capabilities and Preferences) specifications together provide all the information a transcoder needs to decide how to transcode the content. For the most part, the transcoder only works with numbers to make its decision.

In addition, the format is derived solely based on transcoding considerations and can be applied at various levels of granularity, based on design choices for transcoding. It can be interpreted as a file-format if transcoding is to be applied to a file as a whole, or as a packet-format if the unit of transcoding is a network packet.



Formalizing, the internal model for the transcoder is shown in Figure 3. In particular, it consists of the following functional blocks: 1) A parser to parse the SCISM meta-data, 2) a parser to parse the outbound constraint specifications, 3) An optimizer to decide on transcoding options, 4) a meta-data transcoder to scale the media meta-data to obtain the outbound media meta-data, 5) a bit-stream transcoder for the scalable bit-stream part.

#### 2.4. Relation to Network Packetization

It is important to realize that while SCISM is about formats and meta-data describing how format-compliant scalable content is to be adapted, in an actual delivery scenario the content would probably need to be packetized and transmitted. While there may be various design choices for usage of the SCISM format, there are two cases that would be of particular interest, one based on using SCISM as a file-format, and another based on using it as a packet-format.

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

In the packet-format case, the entire SCISM compliant content, including the meta-data and the multi-tier scalable bit-stream, comprises one packet, which can be transcoded by a mid-stream transcoder and transmitted. Packet based scalable video transcoding has been considered before in [2], [3].

In the rest of this paper, we will describe the specifics of the SCISM universal meta-format: comprising the scalable bit-stream format, the meta-data that goes with the media, as well as how the capabilities and preferences are conveyed. We will also

describe how the transcoding operation is conducted based on the meta-format, the meta-data and capabilities. It is important to stress that this paper really attempts to understand and specify what information needs to be conveyed in media headers and in outbound constraints, to make media-type-agnostic transcoding possible. The design of the actual bit-stream and/or XML syntax, to describe media and the outbound constraints, has not been covered. Note that from interoperability and portability considerations, it may be found more convenient to use XML based languages for the SCISM meta-data and outbound capabilities and preferences specifications, but we do not discuss these issues in this paper.

### **3. Scalable Bit-streams and SCISM Meta-Bit-Stream-Format**

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) apply to different types of media, and often more than one kind is combined. From an understanding of how a generic scalable bit-stream is naturally organized, we propose a common media-type-agnostic bit-stream-format for all scalable media, referred to as the SCISM meta-bit-stream-format. This corresponds to the scalable bit-stream part of SCISM in Figure 2.

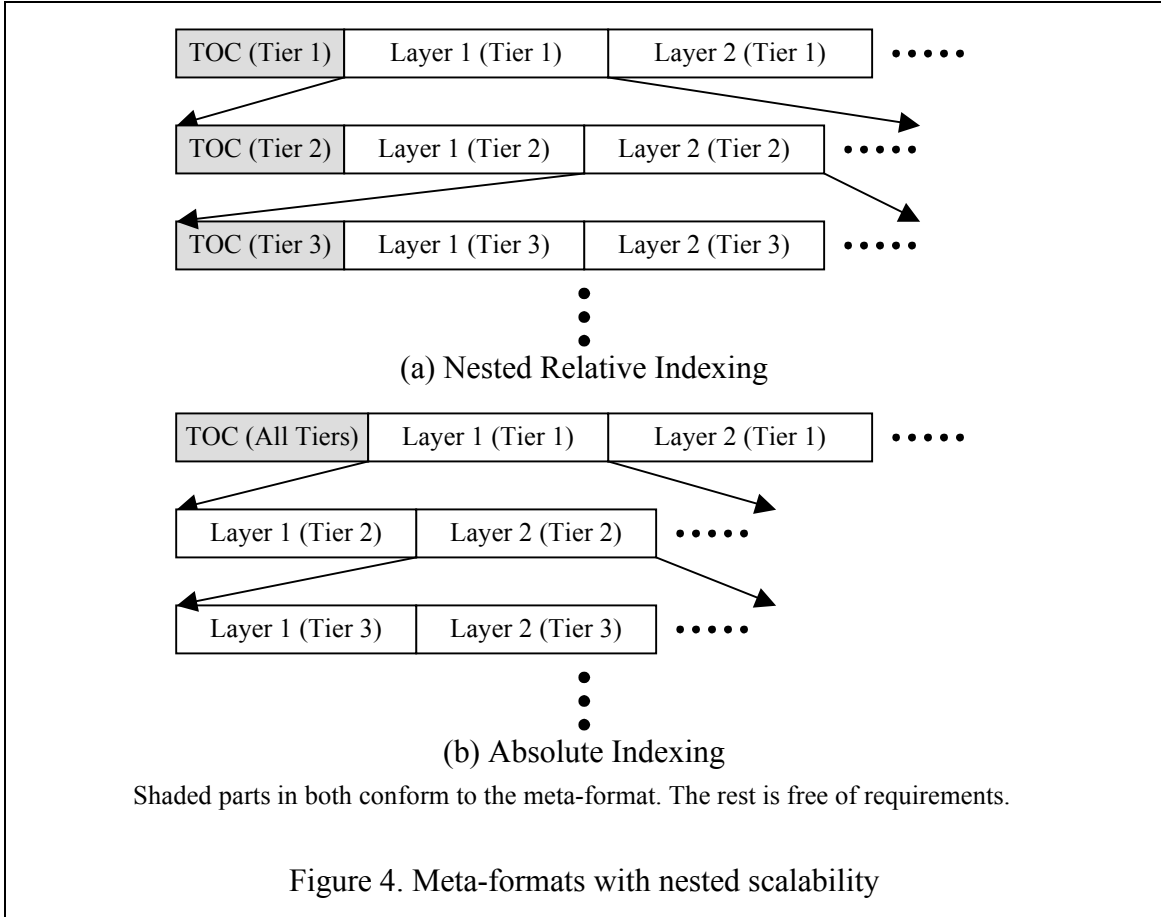
#### **3.1. Nested Scalability Structure**

Any scalable bit-stream inherently contains nested tiers of scalability. The bit-stream is first divided into multiple layers of tier 1 scalability. Here tier 1 is an abstraction, and depending on the actual content it may mean any one of resolution, temporal, SNR and so on. Each data chunk in each tier 1 layer, is further divided into layers of tier 2 scalability, and so on. Again, tier 2 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 meta-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.

#### **3.2. SCISM meta-bit-stream format**

The proposed SCISM meta-bit-stream-format is based on this inherent nature of scalable bit-streams, and comprises nested tiers of scalability indexed by Tables of Contents (TOCs), as shown in either variant in Figure 4. The only difference between the two variants is in the organization of the Table of Contents (TOC), which we will describe soon. But the point to note first is that the essential data part in both is organized in multiple nested scalability layers. While the actual content may vary from media to media, the only requirement for transcoding is that the Header and the TOCs conform to the meta-format exactly.

The above-described meta-bit-stream-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.

The purpose of the TOCs is to provide easy access to chunks of the bit-stream for dropping, or truncating during the transcoding operation. Depending on the way the Table of Contents (TOC) is specified there can be two formats, shown in Figure 4(a) and Figure 4(b) respectively. In the (a) Nested Relative Indexing case, there are multiple small one-dimensional TOCs, each specifying the offsets relative to itself for its constituent layers at the same tier. If the constituent layers have further nesting, at the offsets specified there would be the next tier TOCs to provide the relative offsets to find their constituent layers, and so on. In the (b) Absolute Indexing case, there is one big multi-dimensional TOC at the beginning, which provides the offsets relative to itself to each layer at the deepest nesting tier.

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  $l_0 \times l_1 \times \dots \times l_{L-1}$  data chunks  $B(j_0, j_1, \dots, j_{L-1})$ , where  $j_0=0, 1, \dots, l_0-1$ ;  $j_1=0, 1, \dots, l_1-1$ ;  $\dots$ ;  $j_i=0, 1, \dots, l_i-1$ ;  $\dots$ ;  $j_{L-1}=0, 1, \dots, l_{L-1}-1$ . A way to visualize this data is to consider a  $L$ -dimensional *data cube* of size  $l_0 \times l_1 \times \dots \times l_{L-1}$ , the  $(j_0, j_1, \dots, j_{L-1})^{\text{th}}$  element of which is the data chunk  $B(j_0, j_1, \dots, j_{L-1})$ , 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  $j_0$ .

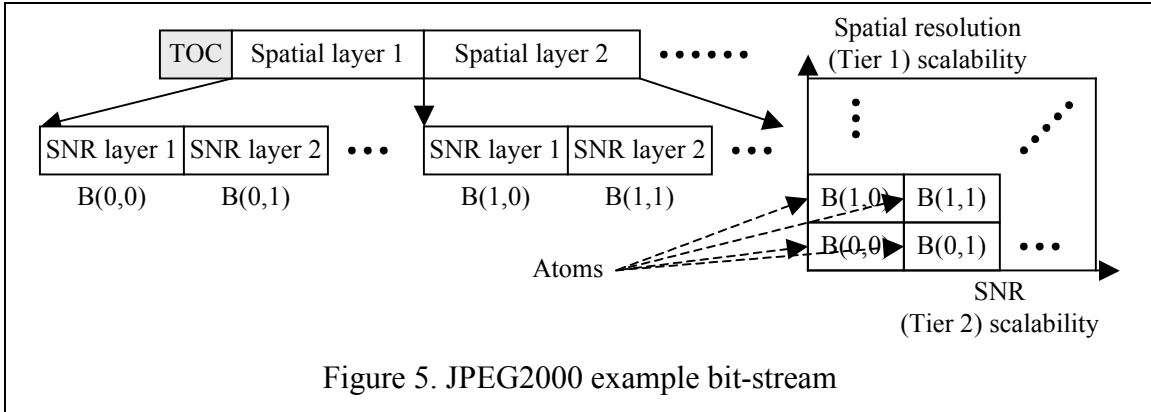


Figure 5. JPEG2000 example bit-stream

Using an example of the first two tiers of JPEG2000 RLCP progression mode, we can visualize the data as organized in a 2-dim cube ( $L=2$ ) as shown in Figure 5. The full bit-stream apart from the header and the TOC can be visualized as being obtained by scanning the atoms in the data cube in row-by-row order, starting from the bottom and moving up. 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.

In the Absolute indexing case, it is also possible to change the order of the data atoms so that the bit-stream is obtained by scanning the data cube in any order other than row-by-row, such as zigzag etc.

While the meta-bit-stream format and the data cube representation has been defined above for true scalable bit-streams where successive layers in each tier are handled incrementally by an eventual recipient, the same 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

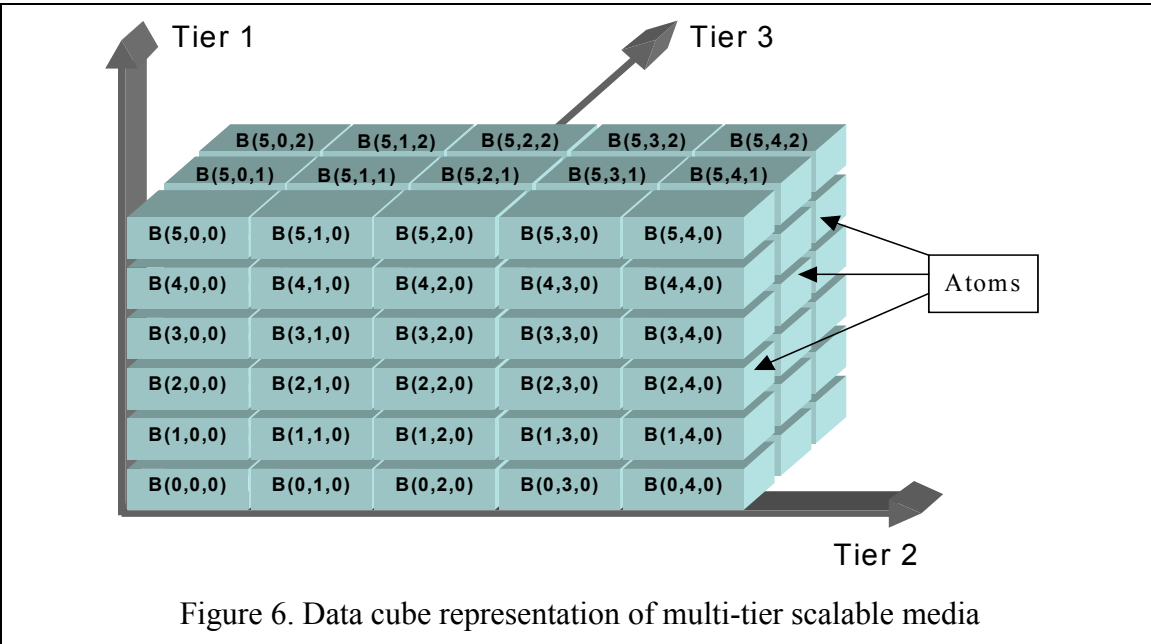
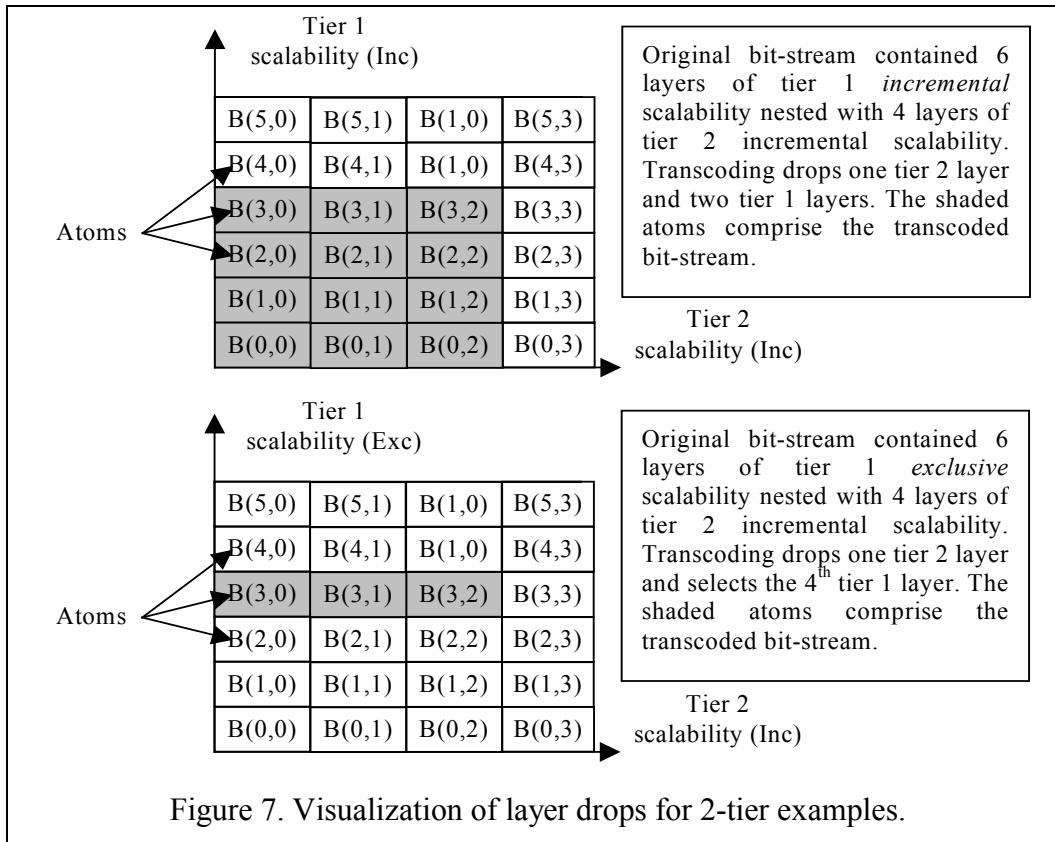


Figure 6. Data cube representation of multi-tier scalable media



is fully multi-version where each atom is an independent version. If all tiers are *incremental*, the bit-stream is truly scalable. In the most general case, tiers could be mixed between *incremental* and *exclusive* scalability. In all cases however, the same meta-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. Transcoding

The above meta-bit-stream-format allows multiple tiers of scalability to co-exist in a bit-stream, and allows simple transcoding tasks like truncation, bit-stream skips and rearrangement to produce bit-streams at different scales in a variety of ways, without knowledge of the actual content or encoding scheme. If all media routed through a transcoder abide by this meta-bit-stream-format packaged as part of SCISM, the transcoder can transcode content without needing to decompress or decrypt. Furthermore, since only the structure of the data is important, the same infrastructure can be used for all types of media, both present and future, as long as they comply with SCISM.

Having understood the meta-bit-stream format and the data cube representation, we next define formally a generic transcoding operation on it.

In particular, with a scalable bit-stream conformant with the above meta-bit-stream-format, all transcoding is implemented as dropping layers, repacking the bit-stream and updating the TOCs appropriately, while preserving the same 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, for the  $i$ th tier, either up to  $d_i$  layers are included if incremental, or only the  $d_i^{\text{th}}$  layer is included if exclusive. The transcoded subset bit stream would then be given by the concatenation of the atoms  $B(j_0, j_1, \dots, j_{L-1})$ , where for tier  $i=0, 1, \dots, L-1$  either  $j_i=0, 1, \dots, d_i-1$  for incremental tier, or  $j_i=d_i-1$  for exclusive tier. 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-null transcoded bit-streams must contain at least the layer  $B(z_0, z_1, \dots, z_{L-1})$ , where  $z_i=0$  for incremental tiers and  $z_i=d_i$  for exclusive tiers. 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 transcoding. A couple of examples for the 2 nested tiers case are shown in Figure 7.

### 3.4. Causality Requirement

Because transcoding can be implemented as simple dropping of layers, a transcoder does not need to *decode* or *decrypt* content in order to transcode. However, an encoder or an encrypter must maintain causality in data atoms, so that a decoder or decrypter can still handle transcoded content. In general, it is necessary to ensure that there are no dependencies across layers in exclusive 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  $B(j_0, j_1, \dots, j_{L-1})$ , the encoder only uses information from atoms  $B(k_0, k_1, \dots, k_{L-1})$ , where for incremental tiers  $i$ ,  $k_i \leq j_i$ , and at least one  $k_i \neq j_i$ ; and for each exclusive tier  $i$ ,  $k_i=j_i$ ; within the usual limits  $0 \leq j_i, k_i \leq l_i - 1$ . This ensures that for any usable transcoding, the decoder at the consumer end can decode the content unambiguously.

The causality constraint for encryption is that the starting state of the encryption engine for atom  $B(j_0, j_1, \dots, j_{L-1})$ , is derived from the ending states of the encrypter for adjacent causal atoms of incremental tiers  $B(k_0, k_1, \dots, k_{L-1})$ , where for incremental tiers  $i$ ,  $0 \leq j_i - k_i \leq 1$  and at least one  $k_i \neq j_i$ ; and for exclusive tiers  $k_i=j_i$ ; within the usual limits  $0 \leq j_i, k_i \leq l_i - 1$ . Progressive encryption enabling transcoding without decryption has been considered in [2], [3].

Finally note that even if the structure of the encoded bit-stream format is exclusive in certain tiers, the type of encryption applied may modify the exclusivity. For example, a fully multi-version bit-stream with all tiers exclusive can actually be converted to a fully incremental bit-stream for all practical purposes if the encryption applied uses information across boundaries of exclusive tiers.

### 3.5. Mid-stream transcoding for combinations

While so far, what we have considered is the generic model for transcoding that generates a single lower version that allows either decryption/decoding for eventual experience, or re-transcoding to other lower versions, there are other scenarios where different things may be done. This particularly applies to the case where a mid-stream transcoder must deliver a combination of several versions of a piece of media, to be eventually extracted by other downstream transcoders. In this situation, a mid-stream

transcoder could send the bounding box containing the different versions, which though wasteful allows re-transcoding to unknown lower versions downstream. Exclusivity of tiers is not considered so as to provide the option for downstream transcoders

Alternatively, if the versions needed are known exactly, it can save bandwidth by converting the unused atoms in the non-intersecting region into empty ones while preserving the same structure of the bit-stream corresponding to the bounding box. Atoms can be made empty by dropping the corresponding bit-stream component, while pointing the corresponding TOC entries to empty chunks.

## **4. Attributes**

### **4.1. Definition**

Now that we have seen the structure of the meta-bit-stream-format, and what a transcoding operation involves, we next need to talk about ways a transcoder can decide which layers to drop, without knowing what the media is all about. A transcoder is expected to have knowledge of the capabilities and preferences of its outbound connection(s). At the same time, headers in the input scalable media contain descriptions pertaining to certain scalability properties based on which the content may be transcoded. The bridge between the two sides is provided by *attributes*. The capabilities and preferences as well as the media meta-data speak the same language through attributes, so that a transcoder can decide how to drop layers to match the two sides. If a transcoder finds that the capabilities of an outbound connection cannot support the full media data, then layers are dropped until they can.

*Attributes* are nothing but certain quantifiable properties relevant to media experience. However, they have different interpretations for different entities in the delivery model. To the media creator/originator, they are quantifiable properties based on which a content may be transcoded. To a media consumer they are quantifiable properties to indicate its limitations and preferences. To a transcoder, they are simply numbers based on which it must decide how to drop layers and transcode an input bit-stream.

In particular, the *attributes* and their quantified values are used to describe both the capabilities and preferences of a transcoder's outbound connection, referred to as *outbound constraints*, as well as to describe a received media in terms of the minimum capabilities that a client should have to experience it, referred to as *media description* that occurs in the SCISM meta-data.

However, note that not all attributes are relevant to all types of media, and not all attributes may be specified for all media content. Likewise, not all attributes for an outbound connection may be known to a transcoder. The transcoder only transcodes content when attribute codes found in the media description are also involved in certain outbound constraints.

Some examples of attributes are: *size*, *display\_resolution*, *processing\_power*, *number\_of\_speakers* etc. Note that while *bandwidth* is a more commonly used term than *size*, it is not strictly a property of the media, because it depends on the desired latency of transmission. Bandwidth however could be an attribute under a standardized assumption about the latency.

## 4.2. Attribute types and code space

Attributes can be either *reserved* or *custom*. Reserved attributes, like the ones mentioned above (*size*, *display\_resolution*, etc.), have the same standardized meaning across different media-types. Custom attributes are relevant only to one or a few specific types of media.

In addition, each attribute is associated with a 4- or 8-byte code called the *Attribute\_code* that uniquely identifies the attribute. Thus, *size* would have a code that is different from *display\_resolution*, and so on. While codes for reserved attributes may be standardized, other media specific custom attributes can be defined later by allocation of the attribute code space to different companies creating media. Enough reserved and custom code space is left free to allow extensions to denote properties of new types of scalable media, as they evolve. If both the media creator and media experiencing systems are owned by the same company or are owned by different companies in a partnership or agreement, they can define their own attributes and what they mean in the way they want. The only requirement they need to follow is to use a pre-allocated range of attribute codes for their custom media.

## 4.3. Attribute values

The most important feature of all attributes is that they are expressed quantitatively in terms of non-negative numbers, referred to as *attribute values*. For reserved attributes, the quantification is also standardized along with the code. For example, *size* can be expressed in KBytes, *display\_resolution* may be expressed as the diagonal width of the screen in number of pixels, *processing\_power* may be denoted by  $CPU\_speed \times Number\_of\_processors$ , and so on. Whatever method is used to quantify the reserved attributes must be standardized so that uniformity across different types of media and how capabilities are conveyed is preserved. However, the transcoder itself does not need to know what these attributes mean.

For most known attributes, the value is either *non-decreasing* or *non-increasing* with layers. Thus, as more layers are added to a scalable media, the attribute values usually change monotonically.

We will next describe the overall media format in detail and show how media descriptions are conveyed in the headers, so that the transcoder has all the information it needs to decide which layers to drop.

# 5. SCISM Format

## 5.1. Parcels and Components

The content passed around in each transmission instance is called a *parcel*, defined as the basic unit of transcoding. The size of a parcel is really a design choice, and may range from an entire scalable compressed file to a network transmission packet. Each parcel in the generic case may be comprised by multiple media components to provide a composite experience. For example, one component may be an image and a second component may be audio annotation that goes with it; both components are packaged together in a single parcel to provide an experience of image viewing with audio annotation; when parcels like that are transmitted at a high enough rate, we have video. Each media component in a

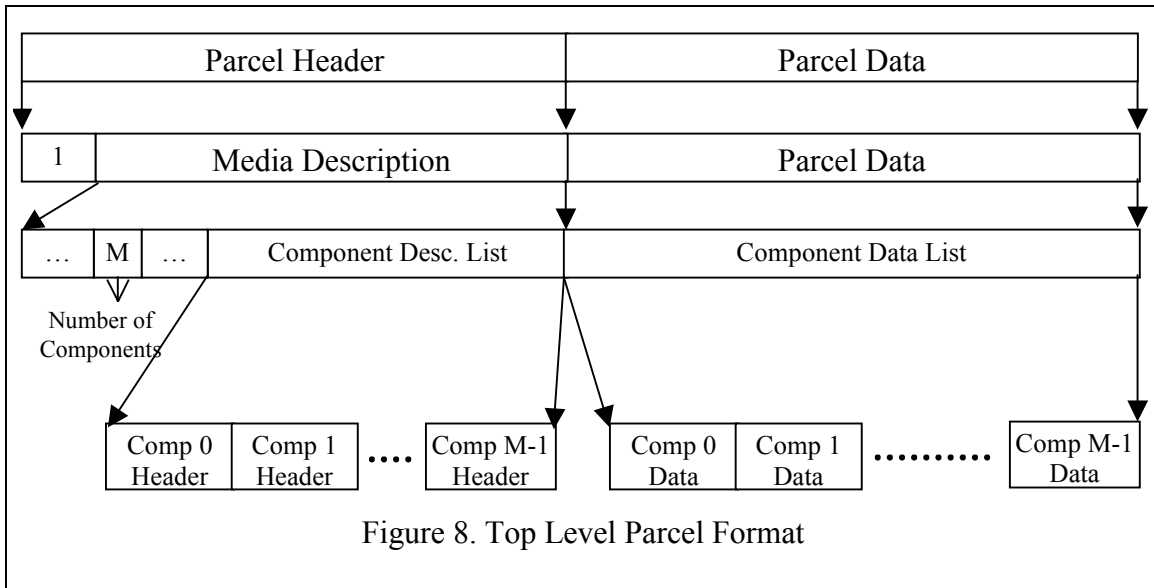


Figure 8. Top Level Parcel Format

parcel is a coded unit of data that may be represented in the scalable meta-bit-stream-format of Figure 4, along with a header containing its description. The overall media description for a parcel consists of the descriptions for the individual components in its header, while the overall parcel data consists of (scalable) coded data for the individual components.

The top-level parcel construct is roughly shown in Figure 8. The parcel consists of two parts: the parcel header and the parcel data. Without going into the details immediately, the parcel header part contains among other things the number of media components, as well as the individual headers for each constituent component. The parcel data part contains the encoded data for the individual components.

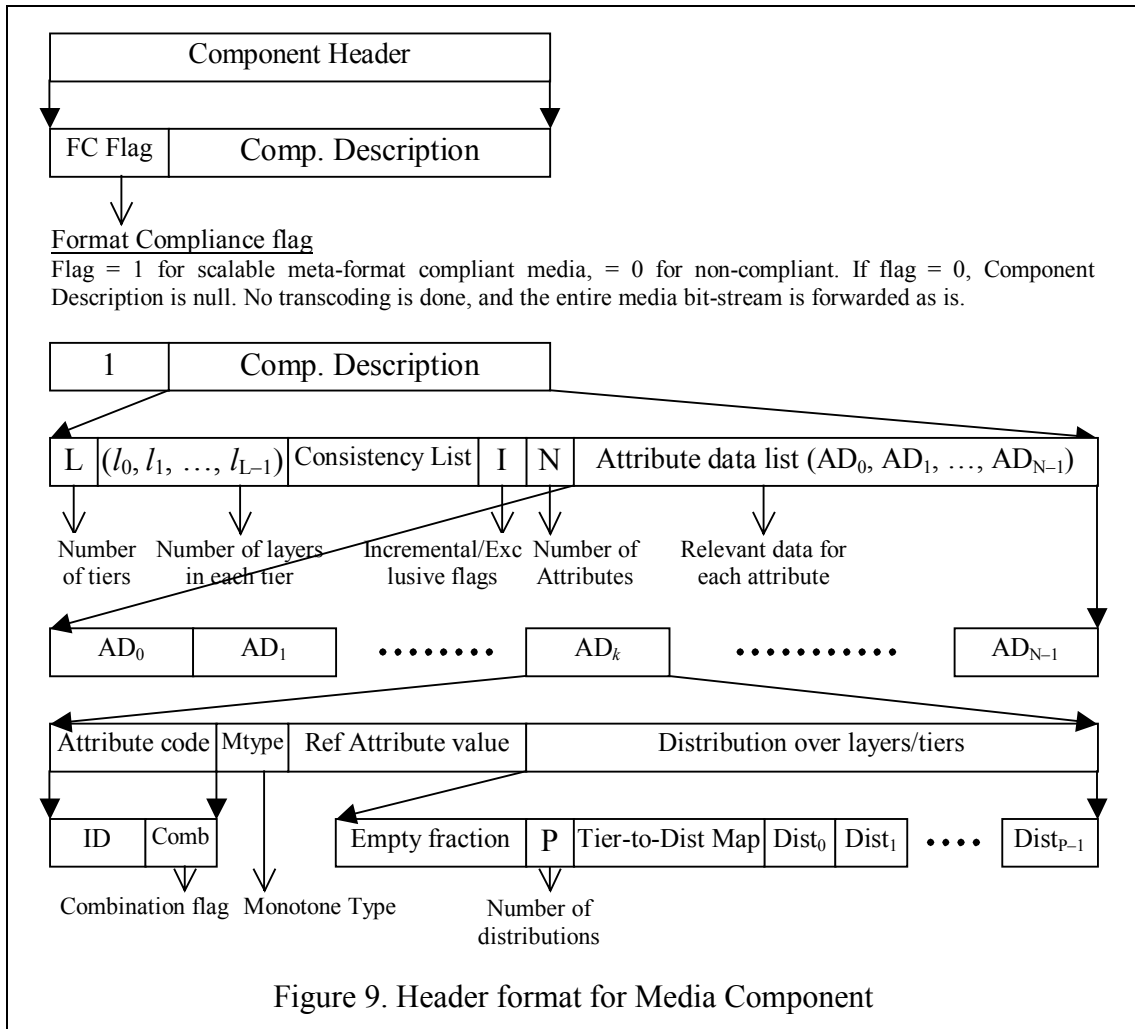
We first describe the format of the component header containing the component description, and then show in greater detail how multiple components are combined in a single media parcel.

## 5.2. Component header format

The format for each media component header is shown in Figure 9. The header starts with a flag specifying whether the media component is a SCISM meta-format compliant scalable media or not. If not, no transcoding is done, and the entire media parcel is forwarded as is to the outbound connection(s). There is no component description in the header in this case. If however the flag indicates that the parcel is scalable and SCISM compliant, then the description follows in the header.

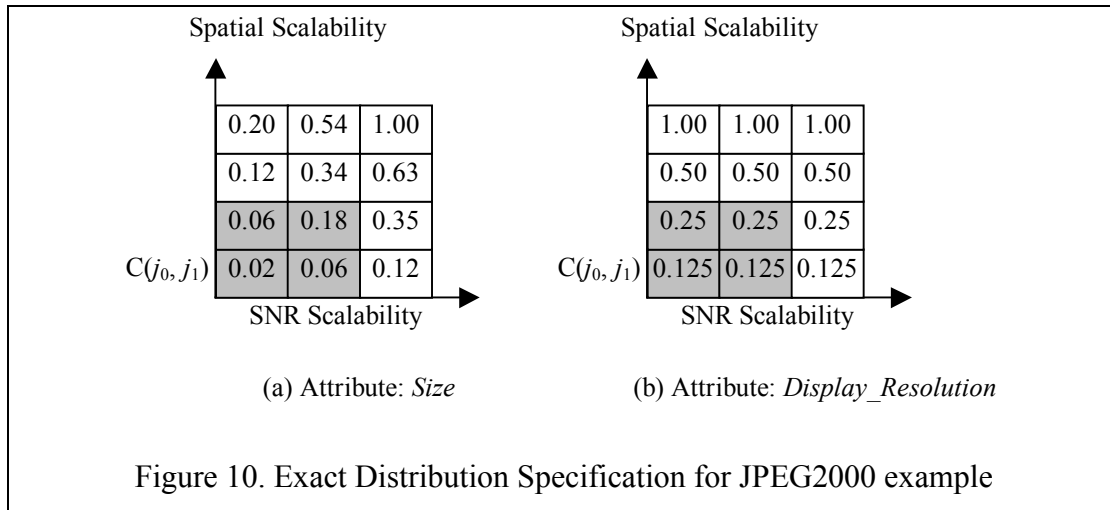
The component description contains  $L$ , the number of nested scalability tiers, followed by  $l_i$ , a list containing the number of layers in each tier  $i$ . Next comes a list called the consistency list, consisting of a subset of tiers that are important for preserving consistency across parcels of the same type. We will explain this further in the section on consistency (Section 7).

Following the consistency list is an  $L$ -bit field, called *Incremental\_Exclusive\_Flags* that describes by a single bit for each tier, whether the tier is in an incremental scalable format, or whether there are multiple independent exclusive layers packaged within the tier. Recall that the same bit-stream format and consequently the media component



header can describe incremental, exclusive or mixed bit-streams. While exclusive tiers are really a special case of incremental tiers, these flags are needed so that a transcoder may increase the transcoding efficiency by knowing that some causal atoms will not be used (see section 3).

The next field is  $N$ , the number of attributes relevant to the media, followed by a list of required data for each of them. The data for each attribute first contains the unique *Attribute\_code* code that identifies this attribute. The *Attribute\_code* actually consists of two fields, *Attribute\_ID* and *Attribute\_combination*. The *Attribute\_ID* is a unique identifier, and *Attribute\_combination* is a field that describes how the attribute value changes when combined with another media component having the same attribute. Possible values are *additive*, *maximum*, *minimum* and so on. For example, *size* is always *additive* in combination, but *display\_resolution* is the maximum of individual components after combination. That is, when two or more media components are combined, the *size* required is the sum of the sizes required for all of them. On the other hand, the *display\_resolution* required is the maximum of all of them. The relevance of this field will become clearer when we describe combination of media components in the next paragraph. Overall, the unique *Attribute\_code* not only identifies the attribute, but also defines its behavior when combined with another component.



The next field is the *Attribute\_Monotone\_Type*, which indicates how the attribute value changes with increase in layers. Possible types are monotonic non-decreasing, monotonic non-increasing, non-monotonic with the number of layers.

The next field in attribute data is the *Reference\_Attribute\_value*. This is the numeric reference value of the attribute, which when multiplied with distribution values that follow later, yield the attribute value for various layer drop options.

The *Reference\_Attribute\_value* field is followed by a specification of how the attribute value changes when layers are dropped. This specification is called the distribution because of its parallels with the cumulative distribution of a random vector. The specified distributions can either be *exact* or *approximate*.

The distribution is similar to a multidimensional cumulative distribution. If there are  $L$  nested tiers with  $l_i$  layers in the  $i$ th tier, we need to transmit a  $L$ -dimensional matrix of size  $l_0 \times l_1 \times \dots \times l_{L-1}$ , whose  $(j_0, j_1, \dots, j_{L-1})^{\text{th}}$  element denoted  $C(j_0, j_1, \dots, j_{L-1})$ , for  $j_0 = 0, 1, \dots, l_0-1; j_1 = 0, 1, \dots, l_1-1; \dots; j_i = 0, 1, \dots, l_i-1; \dots; j_{L-1} = 0, 1, \dots, l_{L-1}-1$ , is a number in  $[0, 1]$  specifying a fraction of the reference attribute value, the component would have if only up to  $(j_0, j_1, \dots, j_{L-1})$  layers were transmitted, along with an optional *empty* multiplier  $C_\emptyset$  in  $[0, 1]$  specifying the fraction of the reference attribute value the component would have when the entire component is dropped, *i.e.* none of the layers are transmitted. The default empty multiplier is 0. The total number of fractions that need to be sent is therefore  $1 + l_0 \times l_1 \times \dots \times l_{L-1}$ . Note that for a monotonic non-decreasing type attribute, the fraction  $C(j_0, j_1, \dots, j_{L-1})$  would be analogous to the cumulative distribution of a multi-dimensional discrete random vector, if the *Reference\_Attribute\_value* were the attribute value corresponding to the full media with no layer drops. In any case, the *Reference\_Attribute\_value* multiplied by the last fraction  $C(l_0-1, l_1-1, \dots, l_{L-1}-1)$  yields the *full attribute value*, or the value of the attribute the media would have if it were transmitted as is without any layer-drop transcoding.

For JPEG2000 RLCP progression mode, the *size* and *display\_resolution* attribute distribution specifications may look as in Figure 10. Both are non-decreasing monotonic. Here we have four spatial scalability layers nested with three SNR scalable layers each. Note that in Figure 10(b), the display resolution attribute does not change with SNR scalable layers. As a result of transcoding, if a SNR layer and two Spatial layers are dropped, the *size* attribute of the transcoded bit-stream shown shaded in Figure 10 would

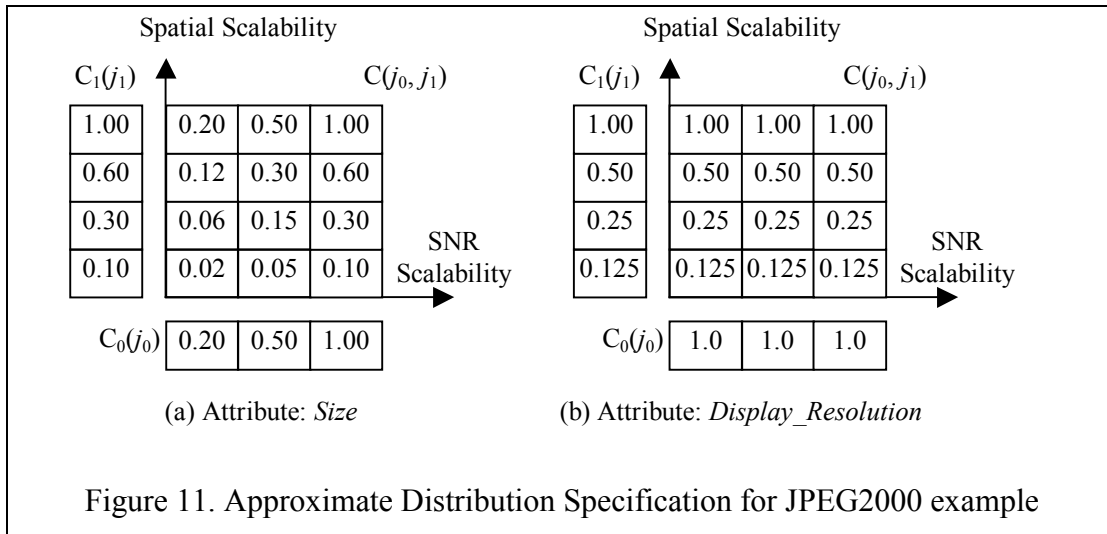


Figure 11. Approximate Distribution Specification for JPEG2000 example

be 0.18 times the reference *size* value, while the *display\_resolution* attribute would be 0.25 times the reference *display\_resolution* value.

Oftentimes, it is 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  $C(j_0, j_1, \dots, j_{L-1})$  is obtained approximately as  $\hat{C}(j_0, j_1, \dots, j_{L-1})$  using a product combination of marginal distributions. That is, the specification involves  $P$  lower dimensional cumulative distributions  $C_i(\cdot)$  that cover  $L$  dimensions together:  $\hat{C}(j_0, j_1, \dots, j_{L-1}) = C_0(\cdot) \times C_1(\cdot) \times \dots \times C_{P-1}(\cdot)$ . The empty fraction  $C_\emptyset$  is transmitted separately.

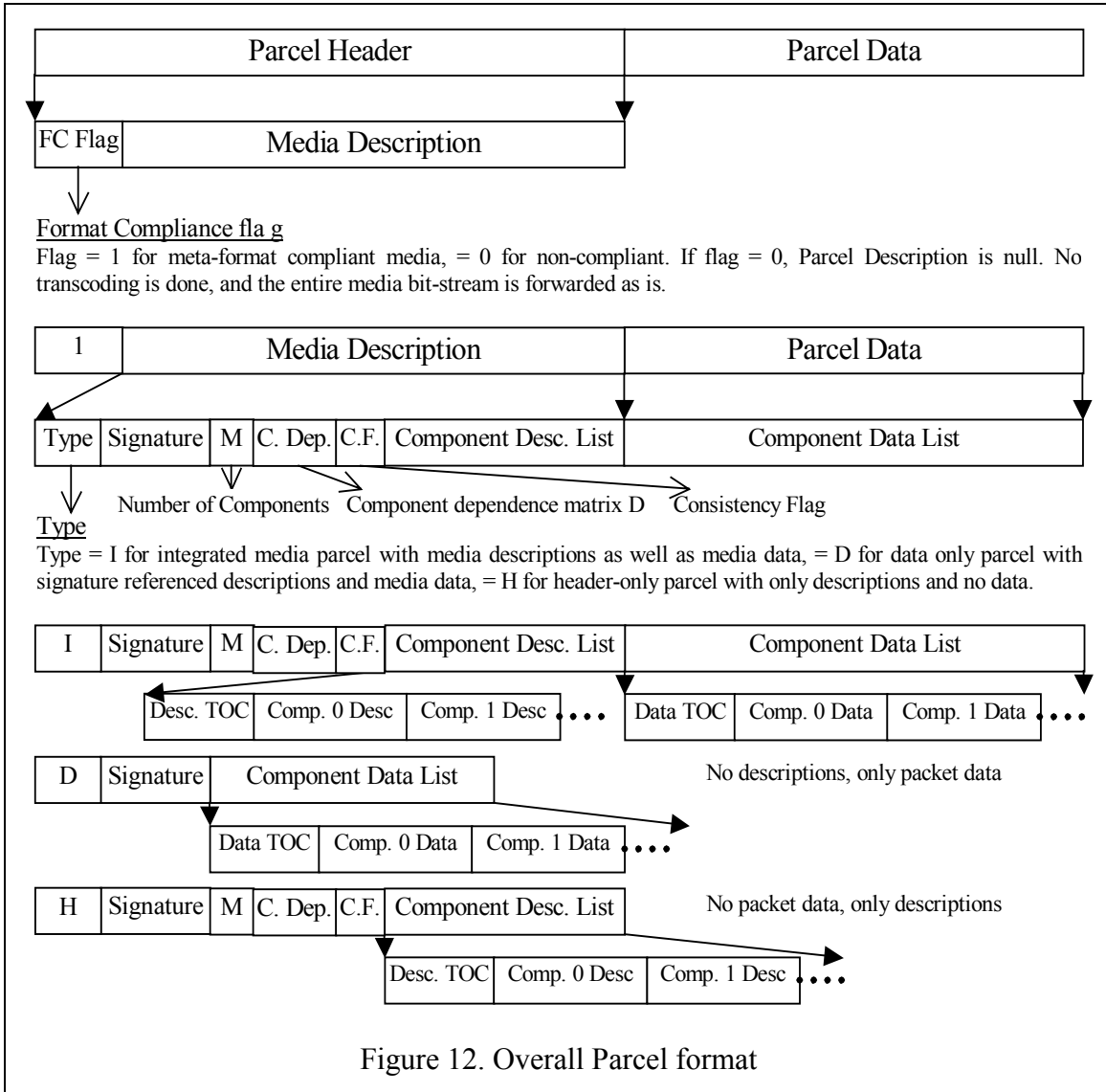
In order to cover all manners of distribution specifications, exact or approximate, the following specification methodology is used. The meta-data contains first the empty fraction  $C_\emptyset$ , followed by the number  $P$  indicating the number of product distributions specified, followed by a list of  $L$   $P$ -ary elements, one for each of  $L$  tiers, indicating which tier map to which distribution. Following this are the actual specifications of the  $P$  distributions in order. The actual fractions in the distribution may be quantized to save bits. One possibility is to divide the range  $[0,1]$  into 256 uniformly or logarithmically spaced levels and use an 8-bit representation for each fraction.

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 11. As seen in Figure 11(b), the *display\_resolution* has been represented exactly using the approximate approach, while the *size* is represented only approximately.

### 5.3. Parcel header format

Now, let us consider in detail how individual media components are combined in a parcel structure. The top-level parcel format is shown in Figure 12. Here again, the first Format Compliance (FC) flag denotes if the parcel is SCISM meta-format compliant or not. If not, the entire parcel contents is forwarded or transmitted without any kind of transcoding.

If the flag indicates that the parcel conforms to SCISM then it can be one of three types, defined by the value of the Type field. Type = I (Integrated) indicates an integrated parcel with media descriptions and data, Type = D (Data only) indicates a parcel with no



descriptions only data, and Type = H (Header only) indicates a parcel with only descriptions and no data. A *signature* field that uniquely identifies the parcel *class* (type) follows the type field. The transcoder stores for future reference in its internal memory, all the header information as well as the layer drop decisions made for a parcel, indexed by its signature. The signature may be derived in part from the network session id. Once a signature has been registered in the transcoder, Type D parcels can be sent, in which case, the media description (header information) corresponding to the signature in the parcel is looked up in the transcoder's internal memory. The description and decision information stored for each signature is updated every time a new parcel with the same signature (class) is routed. For Type I and H parcels, the new media description in the current parcel replaces the transcoder's internal stored description, while for Type I and D parcels, the transcoding decisions made for the current parcel replaces the transcoder's internal stored decision for the class. The stored information enables use of Type D parcels, as well as allows maintaining consistency of transcoding, which will be covered in Section 0.

For a Type I or H parcel with header data, the signature field in the parcel header is followed by a specification of the number of media components, followed by dependency data for the components referred to as Component Dependency, followed by a flag called the Consistency Flag, followed by the list of individual media component headers each in the format of Figure 9. For a Type I parcel, this parcel header is followed by the list of the actual coded scalable data for the components each in the meta-bit-stream-format of Figure 4. For a Type H parcel, the parcel ends at the end of the header. For a Type D parcel there are no headers, but it only contains the list of scalable data components each in the format of Figure 4. We next describe the Component Dependency specification for Type I and H parcels.

When different media components are combined, it is necessary to include a specification for dependency between different components. Certain components in the media must be included after transcoding even if it is only the lowest scalability layer B(0,0,..), while certain others may be dropped entirely. Furthermore, depending on the media, if one component is included, certain other(s) must be included too. All this information at the component level is conveyed in terms of a Component Dependency specification.

If there are  $M$  components in a media parcel, the component dependency rules are specified in terms of an  $M \times M$  matrix  $D$ , where each element  $d_{ij}$  holds a special meaning. The diagonal elements  $d_{ii}$  are binary and specify whether the  $i$ th component must be included, even though it is only the lowest layer after transcoding.  $d_{ii} = 1$  indicates that the  $i$ th component must be included, while  $d_{ii} = 0$  indicates that the  $i$ th component may be dropped if needed. The non-diagonal elements  $d_{ij}$ ,  $i \neq j$ , are 5-ary and specify whether the  $j$ th component must be included or excluded if the  $i$ th component is included or excluded.  $d_{ij} = 0$  indicates that there are no dependencies between the  $i$ th component and the  $j$ th component;  $d_{ij} = 1$  indicates that if the  $i$ th component is included the  $j$ th component must also be included;  $d_{ij} = 2$  indicates that if the  $i$ th component is included the  $j$ th component must be excluded;  $d_{ij} = 3$  indicates that if the  $i$ th component is excluded the  $j$ th component must be included; and  $d_{ij} = 4$  indicates that if the  $i$ th component is excluded the  $j$ th component must also be excluded. With this simple specification methodology a wide variety of dependencies can be readily conveyed.

There is one assumption that is used to resolve contention between different components for inclusion in the transcoded parcel. That is, a component whose description in the header, and data in the scalable bit stream part, occurs earlier than another component usually gets a higher priority for inclusion. In other words, components in a parcel occur in order of importance to the overall media experience.

In addition to the dependency information, there is a flag called the Consistency flag that is also conveyed as part of the header. This flag indicates if the component inclusion should be maintained consistent with the decisions made for the previous parcel of the same type. We will defer description of this flag to the section on Consistency.

#### **5.4. Parcel attributes**

Given the attributes and their values for the individual components, the attribute values for the overall parcel are obtained as follows. The attribute list for the overall parcel contains the union of all the attributes specified for all its components together. Furthermore, when the same attribute occur in one or more components, the combination

type defined in the *Attribute\_combination* field of *Attribute\_code* determines the overall value. For example, if *Attribute\_combination* = *additive*, the overall attribute value is the sum of attribute values of individual components; if *Attribute\_combination* = *maximum*, the overall attribute value is the maximum of the attribute values of individual components. The overall attribute values of the transcoded parcel are used in the transcoding operation to decide which layers from which components to drop in order to satisfy the imposed by the outbound constraints.

## 6. Outbound Constraints

Universal transcoders may reside mid-stream in a delivery network or at an edge server, or these may be integrated in media servers to which clients connect directly. While in the former case, the overall delivery architecture is responsible for conveying to a transcoder the aggregated capabilities and preferences of its outbound connection(s), in the latter case clients convey their capabilities and preferences directly to the server when they make a request. Certain reserved attributes, can be sensed by transcoders themselves (or other agents) from the outbound link. In general, the capabilities and preferences received by the transcoder from a variety of sources with regard to a single recipient yield a set of *outbound constraints*, expressed in terms of attributes and requirements on their values.

Along with the SCISM meta-format, the specification of the capabilities and preferences of the receiving clients and links must also be standardized so that these can be conveyed to a transcoder unambiguously. The specifications are based on imposition of constraints on definable multivariate functions called *measures* of the attributes. Definable measures are essentially linear combinations of products of simple univariate functions of attribute values. The definition comprises: (i) the number of product terms  $N$  in the combination, (ii) the number of elements  $n_i$  in each product term, (iii) the attribute codes for the attributes  $a_{ij}$  in each product term, (iv) the function codes for certain simple univariate functions  $f_{ij}(\cdot)$  on the attribute values, and (v) multipliers  $\lambda_i$  for the linear combination, so that the overall measure is:

$$\sum_{i=1}^N \lambda_i \prod_{j=1}^{n_i} f_{ij}(a_{ij})$$

$f_{ij}(x)$  are simple univariate functions like  $x$ ,  $x^2$ ,  $x^{-1}$ ,  $\log(x)$ ,  $e^x$ , etc., codes corresponding to which are to be included in the standard specification.

The constraints to be imposed on the above-defined measures are of two types, as explained below:

**Limit Constraints:** The outbound constraints most often consists of specific limiting values for attribute measures, known as *limit constraints*. These constraints are specified as *maximum* and/or *minimum* supportable values for outbound connections for the measure. When both the maximum and the minimum are specified for an attribute measure we have a *range* of supportable values for it. An example of a limit constraint is: *size/latency* < 300 KB/s. Here *size* 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.

**Optimization Constraints:** It is also possible to specify the outbound constraints in terms of a requested *minimization* or *maximization* of an attribute measures. In this case, the description consists of whether minimization or maximization of the measure is desired. The most important example of such a constraint occurs in rate-distortion optimization, where a measure like  $mean\_squared\_error + \lambda \cdot size$  is minimized. Here the *size* attribute corresponds to rate (R), while the *mean\_squared\_error* attribute corresponds to distortion (D). Encrypted domain transcoding based on minimizing  $D + \lambda \cdot R$  has been covered in [2], [3].

Note that one outbound constraint specification may consist of several limit constraints but only one optimization constraint.

A mid-stream transcoder may receive sets of several outbound constraint specifications from multiple recipients. In this case, it needs to make its decision based on each, and send a combined bit-stream containing the union of the atoms needed for each. Alternatively, it can receive a single specification, which is in some sense a union of the constraints for all downstream receivers.

## **7. Consistency across parcels of same type**

Often it may happen that multiple parcels of the same type would need to be sent through the transcoder to the same recipient. This may happen for example, when each parcel is a network packet. In such circumstances, it is not practical to include the media descriptions in each parcel, and expect the transcoder to drop layers as appropriate. While it is wasteful of bandwidth and processing power, it may also lead to lack of consistency at the receiver. For example, if a consumer receives one presentation slide at a different resolution than the next, it would not be a very pleasant experience for him.

The way to get around this problem is to use a common media description for a *class* of parcels, typically of the same type. The transcoder remembers the media description data as well as the transcoding decisions, for a class registered in it indexed by an identifying signature. When a transcoder receives a parcel containing description data (Type I or a Type H parcel) for a class for the first time, it creates an entry in its internal buffer corresponding to the given signature. If the given signature already exists in memory, it is overwritten. Next, if a Type D parcel belonging to the same class is sent, with only the signature in lieu of the media descriptions, the transcoder looks up the descriptions from its own memory, makes the component and layer drop decisions, and stores the new decisions in memory for the class. If a Type H parcel is sent, the descriptions stored for the class are simply updated. If a Type I parcel is sent, first, the parcel description in memory corresponding to the given signature is updated; next, the layer drop decisions are made using the new descriptions; finally, the new decisions are stored in memory for the class. For Type D and Type I parcels of a class, the transcoder remembers its decision for future consistency.

Consistency refers to a constraint as per which, the component drop profile for each parcel as well as the layer drop profile for each component is left unchanged from one parcel to the next for the list of tiers mentioned in the consistency list of the component's header (see Figure 9). The consistency flag in the parcel header simply indicates if the component inclusion would have to be maintained the same as the component inclusion in the previous parcel of the same class or not. The consistency list in the component

would typically contain a subset of all tiers; and for the consistent tiers of a component, the number of layers dropped would have to be the same as the decision made for the previous parcel, stored in memory for the class. These are additional constraints that the layer drop decision mechanism has to adhere to. In the decision making phase of transcoding, the component inclusions are either maintained the same as the pre-stored inclusions for the class or not, depending on the current consistency flag corresponding to a class. Additionally, the tiers in the current (stored) component consistency list for a class are maintained the same as the pre-stored decisions for the class. Thus, for a Type I parcel, based on the order of operation as mentioned in the previous paragraph, the new consistency flag and component consistency lists are used in the decision making phase instead of the old ones, because the description is updated before the decisions are made, even though the previous parcel's decisions are still used as reference.

The consistency mechanism ensures consistency in delivery of parcels belonging to the same class, while still allowing adaptation based on changing descriptions for same type parcels and changing outbound characteristics (such as bandwidth), by permitting change in layer drops for tiers not included in the consistency list.

Each signature persists in memory of the transcoder until it is dropped as a result of not being used. A circular buffer in the transcoder maintains an ordered list of most recently used signatures. When a certain signature has not been used for while a new signature would replace it eventually.

## **8. Constraint based Transcoding**

When a parcel compliant with the Parcel format of Figure 12, is received by a transcoder that knows its outbound constraints, it immediately gets all the information it needs to transcode the content automatically, irrespective of the type of media and content it represents.

For each outbound measure specified with *constraints*, the transcoder first checks to see if all the attributes in the measure occur among the media components in the parcel. If one of the attributes does not occur in the descriptions of any of the media components, the outbound measure is simply discarded as invalid because no transcoding is possible.

For each valid outbound measure specified with *limit constraints* the transcoder checks if the full measure value of the overall parcel satisfies the limit constraints. The full measure value of a parcel is derived from relevant full attribute values for the parcel, which in turn are obtained by combining attributes for media components using the *Attribute\_combination* type field of the *Attribute\_code*. If none of the full measure values violate the outbound limit constraints no transcoding needs to be done to satisfy the limits. The parcel is forwarded or transmitted as is. If at least one of the measures is in violation of the constraints, layers need to be dropped from one or more media components.

Given a list of measures that violate the outbound restrictions, determination of which layers to drop from which components can be implemented in a variety of ways, ranging from simplistic ones to ones involving complex optimizations. If the *Attribute\_Monotone\_Type* field included in the component headers indicates the attribute is monotonic (non-decreasing or non-increasing), it simplifies the task of finding the layer drops. The actual implementation of a decision rule is beyond the scope of this paper. But a requisite bias should be not to drop more than what we need to do. Every

time layers are dropped, the attributes that already satisfy the constraints are further devalued, thereby degrading the overall experience of the media. It is also not necessary always to satisfy all the *limit constraints*. If it is found that too much may be lost in satisfying the constraints, then certain constraints can just be relaxed.

The *optimization* request, if specified, is a lower priority than limit constraints. Among the choices that do not violate the *limit constraints*, the transcoder chooses the one that maximizes or minimizes the measure value of the optimization constraint. This mode will be particularly useful for selecting optimum layers based on a rate-distortion criterion (i.e. the traditional  $D + \lambda R$ ), or selecting optimum layers based on user's relative preferences of one attribute over the other.

In addition to satisfying the limit constraints, and optimizing based on the optimization constraint, the transcoder needs to maintain consistency with transcoding of the previous parcel with the same signature, as well as satisfy the component dependencies. Note however, that for a mid-stream transcoder, the dependency or consistency considerations may be ignored in the actual bit-stream, though not in the decision making process, since there are multiple recipients downstream. These are enforced only for terminal transcoders that connect directly to media consumers.

Once the decision has been made which layers to drop from which components, the transcoder drops the atoms in the scalable bit-stream, repacks it, updates the appropriate TOCs, and truncates the distribution specifications in the meta-data, before sending out the transcoded parcel. If the transcoder is the last in the chain before it reaches the eventual recipient, then the transcoding operation may comprise extracting only the desired atoms, and discarding the rest.

In general, a mid-stream transcoder may receive several sets of outbound constraint specifications from multiple recipients. It can then make the best decision for each specification, and transmit the bit-stream structure corresponding to the bounding box encompassing the decisions made for each of them. The unused atoms in a bounding box may be emptied if a mid-stream transcoder knows exactly the versions that would be needed downstream. Alternatively, a mid-stream transcoder may receive a single set of constraints, which is the union of individual limit constraints, from a downstream transcoder. In this case, it just makes one decision, and transmits all atoms up to the transcoding point. The exact protocol used for upstream constraint communication between transcoders in a chain has not been covered in this paper, but is a straightforward derivative of the general principles covered here.

## **9. Conclusion**

Use of scalable media for content-agnostic transcoding is well known in the literature. These transcoders do not need to decrypt or decode compressed content in order to transcode it into a form appropriate for lower bandwidth/resolution etc. The underlying assumption behind the transcoding operation is that a transcoder 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 different transcoders are still needed for different types of media content. That is, a transcoder for images compressed in a particular way, say JPEG2000, would still be different from a transcoder for a certain kind of interactive content encoded in an entirely different way.

This paper advances the level of abstraction to develop a flexible methodology for universal transcoding of scalable content, where the transcoding operation is generic enough to be applicable to any type of media having any type of encoding. The transcoder just needs to be told what the structure of the particular content that goes through it is, and how this content is to be transcoded to achieve the desired transcoding operation. This meta-data information can either be part of the header of the media itself, or can be conveyed to a transcoder separately for an entire class of content. Different transcoding 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 loose meta-format (SCISM) that the universal transcoder understands, it still becomes possible to transcode it appropriately.

## **10. References**

- [1]** David S. Taubman and M. W. Marcellin, "JPEG2000: Image Compression Fundamentals, Standards and Practice," *Kluwer Academic Publishers, 2002*.
- [2]** 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.
- [3]** 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.
- [4]** 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.
- [5]** B. G. Haskell, A. Puri, A. N. Netravali, "Digital Video: An Introduction to MPEG-2," New York: Chapman & Hall, Sept 1996.
- [6]** 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.
- [7]** Video Coding for Low Bitrate Communication, ITU-T Recommendation H.263, Nov. 1995.
- [8]** Video Coding for Low Bitrate Communication, ITU-T SG16/Q.15 H.26L Project, Feb. 2000.