Programme ANR VERSO

Projet VIPEER

Ingénierie du trafic vidéo en intradomaine basée sur les paradigmes du Pair à Pair

Décision n° 2009 VERSO 014 01 à 06

du 22 décembre 2009

T0 administratif = 15 Novembre 2009

T0 technique = 1er Janvier 2010

Livrable 1.2

Definition of a functional architecture and communication of the different identities composing the distributed CDN

Auteurs :

S. Vaton, A. Gravey (Telecom Bretagne), F. Guillemin, P. Philippe (Orange Labs), Y. Hadjadj-Aoul (INRIA), Jean Kypreos (ENVIVIO)

Compilé par :

F. Guillemin (Orange Labs)

Juillet 2011

Telecom Bretagne ; Eurecom ; INRIA ; France Telecom ; NDS ; ENVIVIO

Résumé : *(15 lignes)*

This deliverable presents a first draft of the VIPEER video delivery architecture. VIPEER considers that video is distributed using HTTP streaming. In standard HTTP streaming solutions, a single server downloads all successive temporal segments, while the user adapts the segment's encoding rate to the available throughput. VIPEER enhances this architecture by duplicating and caching the various "chunks" of data in "dCDN nodes" within the operator's network, and by allowing the selection the appropriate dCDN node to download the requested chunk. The major building blocks for this architecture are
(1) an interface with a client CDN
(2) a policy for caching and duplicating chunks in the dCDN nodes
(3) mechanisms for monitoring available storage space, link bandwidth and delivered QoE
(4) protocols for serving customers by selecting for each demand and each customer the appropriate dCDN node where the requested content is cached.

Key words : HTTP streaming, Digital Rights Management, Contend Distribution System, Quality of Service, Quality of Experience, Collector, Tracker

Table des matières

Table des figures

1. Introduction

The global picture of the distributed Content Delivery Network (dCDN) studied in the framework of the VIPEER project has been described in Deliverable 1.1.

The objective of a dCDN is to offer to a classical CDN the possibility of disseminating content within a national network (typically the network of an ISP in a country). In this respect, a dCDN could be seen as a capillarity network for disseminating content as close as possible to the end user.

This objective leads to a neat delineation between the role of a CDN and that of a dCDN: a CDN is in charge of disseminating content at a wide scale (say, at the international level) while a dCDN aims at delivering content within a medium size network (say, regional or national network). This is highly desirable since most international CDN have national servers but QoS in the last miles is not controlled, especially for those services in the best effort Internet channel. The role of a dCDN is precisely to improve the dissemination of content in medium scale areas while controlling the level of QoS. It is worth noting that besides CDN, the operation of a dCDN is also desirable for services distributed from national or regional service platforms, such WebTV, catch-up TV, and IPTV services.

The various actors playing a role in the distribution of content in the case of a dCDN as devised by the VIPEER consortium are displayed in Figure 1. The content provider entrusts the CDN to distribute the content. The CDN is in general connected to a Tier 1 network (transit network) to transfer the content worldwide. Nowadays, CDN usually have local servers in countries connected to local networks. While in the current situation, the QoS of services in the best effort channel is not controlled in the local network, the proposition of the VIPEER project is to deploy a distributed CDN in order to improve the QoS and reduce the load of the local network.

A dCDN may span over various types of equipment. More precisely, a dCDN may be composed of several storage and processing capacities hosted by various pieces of equipment of the local network operator (routers, servers in a PoP, etc.). A dCDN may also be composed of such capacities in the switches and routers of the access network, typically the backhaul network of an IAP, possibly shared by several ISPs. Finally, the end user could also dedicate some storage and processing capacities to the dCDN. This latter possibility already exists in peer-to-peer networks.

In spite of the fact that a dCDN spans over several types of equipment, the option adopted by the VIPEER consortium is that a network operator operates a dCDN, typically the ISP to which the CDN is connected. Very often, an ISP is also an IAP and owns the home gateway and the set top box used by the end user. This is notably the case for Orange and major ISP in France. Nevertheless, when an ISP is not the IAP, then the latter may be constrained to offer resources to the former.

As a matter of fact, as we shall see in the following, a dCDN is based not only on bandwidth but also on storage capacities. Hence, to build a dCDN both types of resources are needed. To implement a dCDN, an IAP may actually have to offer storage resources to third party operators. *This is a groundbreaking requirement with respect to the current situation, where only bandwidth is considered in business relationships between operators.* Hence, the emergence of dCDN may call for a major modification of the business models between operators.

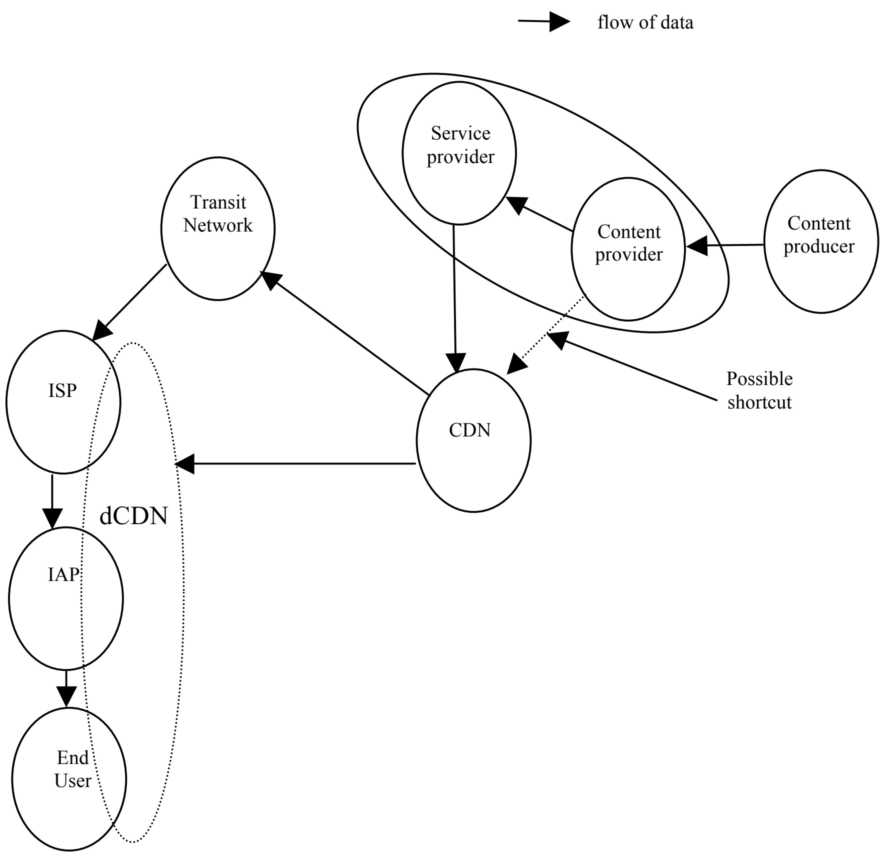

Figure 1 : Various actors of the content distribution chain in the case of a dCDN

The case when an ISP offers resources in terms of bandwidth and storage capacities to third party operators is examined by the FP7 OCEAN project. The technical options of the VIPEER project thus differ from those taken in OCEAN in the sense that in VIPEER, the ISP, who has the knowledge of traffic conditions and resources available within its own network, directly operates the dCDN.

The fact that the ISP directly operates a dCDN also raises issues with regard to Intellectual Property Rights (IPR). As a matter of fact, content providers are reluctant to authorize operators to store content in their networks, in order to avoid abusive replication. This is often an issue when an operator wants to use peer-to-peer networks for disseminating copyrighted material. The fact that content is stored or modified (for instance via transcoding) by a third party is considered as an obstacle by content providers or content producers. In the usual case, content providers pay CDN operators for disseminating their content. When there is a dCDN, new business agreements have to be set up between the dCDN operator,the CDN and maybe with the content provider because content is no more stored by a single entity. *Specific IPR rules should be set up between ISPs operating a dCDN, CDN operators and content provider.*

Finally, the option adopted by the VIPEER project implies that the dCDN does not simply cooperate with the CDN to improve the QoS of the delivered content. The dCDN is an active actor in the content delivery chain and is fully responsible of the content delivery within its own network. In particular, the dCDN is not requested to provide information on the traffic conditions or the topology of the underlying network to a third party (typically an overlay network), which then controls the global delivery of content. This is a fundamental difference with the FP7 ENVISION project, which studies the relationships between the infrastructure network and the overlay network to improve the delivery of content.

The last prerequisite for the VIPEER dCDN is the assumption that content is coded in different formats (coding schemes, bit rates, etc.) and decomposed into chunks of data with sizes of a few tens of Megabytes. The data entity manipulated by a VIPEER dCDN is thus the elementary chunk. This assumption implies that a dCDN is very different from a Content Centric Network (CCN) where information is decomposed into packets and retrieved by end users one by one. This latter operating mode is motivated by real time constraints of interactive services. As a matter of fact, if information is divided into packets, the latency is reduced when compared with a chunk-based operation where the assembly of chunks may lead to large delays. In a CCN, the only source of latency is due to the mechanism in charge of locating individual packets and to retrieve them. The issue of interactive services has to be carefully examined by the VIPEER project, especially when end users are mobile.

Goal of VIPEER: Specify a dCDN composed of a set of network resources operated by an ISP which is also an IAP and possibly controls some pieces of the end user equipment (set top boxes and/or Internet access boxes) to assist a global CDN in the delivery of content in the last miles. The dCDN interfaces with the CDN can store the content to be delivered. Content is available in different formats, and is decomposed into chunks that are individually stored and distributed by the dCDN.

The objective of the present deliverable is to identify the various functions to be implemented in order to achieve this goal.

The organization of this deliverable is as follows. Since video distribution is a central topic in the VIPEER project, we present in Section 2 the video distribution architecture that was selected by WP3, and outline the elements needed to support dynamicity. In Section 3, we identify the various elements of the data plane of a dCDN. In Section 4, we present the control plane of a dCDN. In Section 5, we explain how the VIPEER data plane and control plane are mapped on the dynamic video distribution architecture presented in Section 2. Lastly, Section 6 lists each individual function identified in Section 4.

2. Adaptive video distribution

The VIPEER project considers adaptive video distribution relying on HTTP streaming.

In current HTTP streaming solutions, video is distributed from one origin point to clients through a CDN and some unmanaged network resources where resources are shared among users. Bottlenecks may occur in the distribution chain and customers can use various types of terminals or devices. An encoder creates the various chunks (segments) of a given content. It encodes video and audio sources and creates segment of interleaved and synchronized video and audio within a given system layer syntax. Segments are temporally sized and are independent so as to be compatible with the HTTP streaming technique. Those streams organized in chunks are stored on an HTTP "storage" server. A given client typically gets successive temporal chunks from a single server. HTTP streaming is by nature adaptive in the sense that several representations of a given content are adapted to each type of device in terms of resolution, encoding format and bit rate range shall be available. A client dynamically selects the resolution that corresponds to his viewing device and current available throughput, which is locally estimated (e.g. by estimating available throughput when receiving successive chunks).

However, the VIPEER project attempts to improve the dynamicity of HTTP streaming by allowing a given client to download successive chunks from different servers, and by providing methods for optimizing the selection of the server to use for each chunk based on network performance and QoE monitoring, and by chunk replication and caching optimization.

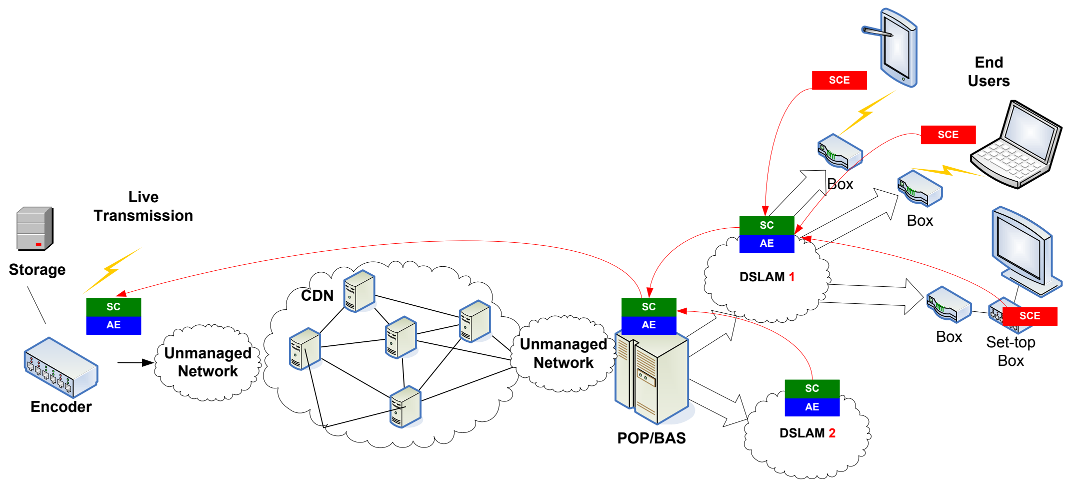

Figure 2 : A video distribution architecture.

In order to better manage HTTP streaming, some intelligence has to be added within the distribution chain. The following elements are part of the architecture displayed in Figure 2:

- Statistical and Contextual Elements (SCE). The role of the Statistical and Contextual Elements (SCE) is to collect static as well as dynamic information at different locations. Examples of relevant dynamic information are the current audience, the bandwidth usage, the cache occupancy, etc. They are completed with static information such as the types of supported codecs, the network maximum bandwidth, users' eligibility to various offers, etc. These elements will drive the initialization of media distribution and also, in a second phase, the dynamics of this distribution. SCE are knowledge elements having a potential influence on the adaptation. Information is collected in numerous locations, e.g. sent back by users or network elements.

- Statistical Control (SC). Based on the information collected by the SCE, a Statistical Control (SC) controls the adaptation through dedicated control signals. A SC may be connected to multiple SCE in different network locations and thus gathers information in order to decide on the potential actions required at the cache and adaptation levels. A statistical control is coupled to an adaptation engine performing the actual content adaptation.
- Adaptation Engine (AE). The Adaptation Engine (AE) performs the adaptation of the content. This adaptation can be dynamic or static. In the present context, "dynamic" means that the content is transcoded in real-time, and "static" means that the adaptation engine provides a set if different versions of the same content.

The relationships between SCE, SC and AE are illustrated in Figure 3. Adaptation may consist of transforming the video content. Examples of transformation can be transcoding, i.e. change of codec, of bit rate or of resolution, but also the modification of the random access point (e.g., to enable fast content browsing) and digital right management (DRM). Different transformations can be operated during an adaptation stage, such as the cascade of DRM decrypting, transcoding and finally re-application of the DRM. The AE performs those transformations and their organization.

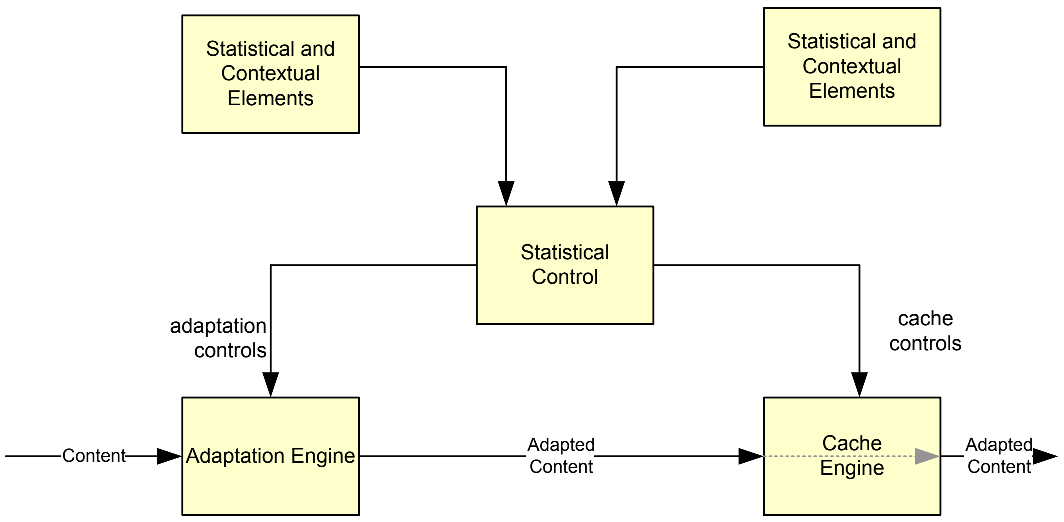

Figure 3: Relationships between adaptation elements

Note that an adaptation engine potentially requires a large quantity of processing resources; if transcoding is considered, the adaptation element may require intensive computational capacity.

It is also worth mentioning that if DRM is in force, then it may be required to adapt the DRM in private areas for IPR (Intellectual Property Rights) reasons. Management of DRM with decrypting and re-encrypting rules should not be done inside the adaptation engine. For those reasons, the location of the adaptation engine within the content delivery architecture should be carefully chosen, and it may not be appropriate to locate them within the dCDN.

Nonetheless, for DRM-free contents (YouTube, Dailymotion, etc), adaptation engines may be located within the dCDN in order to manage only specific subset of chunk streams.

The above description is valid for video distribution. Only a subset of the various elements appearing in the above architecture will be integrated in the VIPEER dCDN architecture (see Section 5). But it is worth noting that monitoring available bandwidth as well as bit rate of delivered chunks is a central concept in the video distribution architecture.

In the remainder of this document, we shall consider that transcoding is not implemented in the dCDN. If some transcoding functions (such as IPR management) have to be implemented, then the CDN should be in charge of them, possibly upon request by the dCDN. This assumption shall be revisited in Deliverable D1.3, which shall consider content protected by DRM, and DRM-free content.

3. Data plane of a dCDN

The functional architecture of a dCDN will be described by using the classical dichotomy based on the data and control plane differentiation:

- The data plane is composed of those functional elements in charge of transferring information (data forwarding, data storage, packet dropping, etc.).
- The various functions in charge of controlling the elements of the data plane and the interaction with external entities (end users, client CDN, etc.) via signaling functions compose the control plane.

In general, one may distinguish a third plane, referred to as management plane. This last functional plane consists of all configuration functions for the data and control planes and the relationships with third party players, end users and the client CDN in the present case (service subscription, hot lines, etc.). *The management plane will not be considered in this document*.

In this section, we consider the functional architecture by specifying the various elements that implement functions of a dCDN. We do not consider the actual architecture of the network in terms of links and nodes where functions are located. The various elements of a dCDN could be implemented in different types of equipment (routers, DSLAM, set top boxes, home gateways, etc.).

The basic components of the dCDN are storage capacities managed by some processing facilities and interconnected by links. In the following, the individual storage capacities and their associated processing facilities will be referred to as dCDN nodes. A dCDN node is in charge of storing the chunks of data requested by end users.

These chunks are provided by the client CDN and *are not modified* by the dCDN (in order to prevent IPR issues – see Section 2).

If a given chunk is coded in different formats, then the dCDN will have to store as many versions as necessary or request the missing versions in the ad-hoc format to the client CDN, possibly by requesting some transcoding functions to be performed by the CDN.

The dCDN nodes are interconnected by (virtual) links. These links are virtual in the sense that they are built on top of transmission links, as an overlay, and do not necessarily correspond to real transmission links. The traffic exchanged by dCDN nodes is merged with the rest of Internet traffic. With regard to the bandwidth associated with these links, various options can be envisaged:

1. The dCDN links can be purely best-effort and the dCDN traffic is merged with the rest of best-effort Internet traffic.

2. The dCDN links can have guaranteed bandwidth that cannot be exceeded (such as a TDM circuit, an ATM connection, a lambda in a WDM network or a link of a virtual network set up by means of virtualization techniques).

3. The dCDN links can have a minimum bandwidth guaranteed via scheduling disciplines in the underlying networks (e.g., weighted fair queuing or similar disciplines).

4. The dCDN links may be in a class of differentiated quality of service offered by the underlying network, while having no specific dedicated bandwidth.

In the following, we shall group all the functions related to multiplexing and flow control (e.g., via TCP) into a functional block referred to as FEC for “Flow and Error Control”. The FEC functional block is in charge of multiplexing flows by respecting some bandwidth requirements and/or priority hierarchy. All scheduling mechanisms (priority management, bandwidth sharing, etc.) are contained in this functional block, which also includes replication and storage management.

From a traffic management perspective, it is worth noting that a dCDN raises new issues with regard to resource sharing. As a matter of fact, in classical networks, the most limiting resource is the bandwidth and specific actions have to be taken in order to share bandwidth between the various flows traversing a network. For a dCDN, beyond the bandwidth, the storage capacities also have to be carefully managed. Since a dCDN is highly distributed, it is quite clear that the storage capacities of the various nodes may be quite limited and hence have to be efficiently managed. This is a big difference with CDN servers that are equipped with huge storage capacities. The storage capacities of dCDN nodes are managed thanks to information provided by control plane functions, detailed in the next section. A dCDN node should as a consequence contain a specific functional block dedicated to storage control.

A dCDN should implement a chunk replication policy. This policy can be centralized or distributed. In the latter case, some well known policies such as LRU (least recently used), LFU (least frequently used), random, etc. could be implemented.

Bandwidth sharing within a dCDN has to be performed at different levels. The bandwidth of the dCDN has to be shared with other Internet flows on the links of the network but also inside the dCDN. For this last point, several parameters can be taken into account. First, a chunk can *a priori* be retrieved from several nodes and hence, the source of the chunk has to be cleverly chosen so as to avoid congested links. Second, chunks may be coded according to various bit rates and some instances, it is better to transmit a chunk at lower bit rate than causing congestion, even if the quality of experience of the end user is temporally degraded, which always better to have a frozen picture. Finally, the replication policy of chunks inside the dCDN has a great impact on the congestion of links. There is clearly a need for coupling the supervision of traffic within the network and the replication policy.

Upon request of a chunk at a given bit rate which has to be streamed to the end user, it may happen that the resources locally available at a dCDN node are not sufficient. In that case, the request to the dCDN node should be blocked, and possibly redirected to another dCDN node. We then see that the dCDN should implement an Admission Control (AC) function in order to block some request when resources locally available at the dCDN node are not sufficient to accommodate the request, and more generally to route the request to a sufficiently provisioned dCDN node.

Hence, we see that resource sharing within the dCDN relies on various actions :

- Share the network bandwidth between the dCDN and other Internet flows.
- Share the bandwidth within the dCDN by taking into account the location of chunks, the congestion level of the network and the bit rates of chunks available in the dCDN.
- Perform Admission control to deny some requests if the bandwidth of ongoing chunks transmission becomes too small and thus induces too much latency.

- Implement a replication policy within the dCDN that should take into account the demand of end user (traffic matrix in terms of chunks), the storage capacities, the bit rates of chunks, and the level of congestion of the network (including network topology).

Resource sharing techniques shall be implemented within the FEC functional blocks.

Resource management in a dCDN has to take account of detailed information on the state of the network. This is possible because the dCDN is under the responsibility of the ISP. Such a process would be very difficult if the dCDN were operated by a third party. In general, operators are very reluctant to provide detailed information about the level of congestion and about the topology of their network, as is promoted by the ALTO WG of the IEFT. This is because in an adverse environment, such information could be used to orchestrate massive attacks against the network.

To summarize, a VIPEER dCDN node should contain various functional blocks (FB) in the data plane:

- Storage FB: chunks storage.
- An admission control (AC) FB.
- FEC (Forwarding and Error Control) FB: forwarding and multiplexing of flows with other Internet flows and inside a dCDN.

A dCDN node can thus be decomposed into various functional blocks. There is some processing capacity dedicated to running the various processes related to the various functions.

To run the iterative process described above, the control plane of the dCDN has to provide information about the network and to perform some specific tasks, which are described in the next section.

4 Control plane of a dCDN

4.1 Control plane functions

In the following, we shall assume that a user requests a specific content from the CDN, which then redirects the request to the dCDN. The dCDN does not directly interpret the requests issued by the end user. Instead, the dCDN receives the requests redirected by the CDN and can then choses the chunks that minimize resource consumption under the actual network's condition, e.g., those which are the closest to the end user.

The control plane of the dCDN has to perform several tasks:

1. Offer interfaces to the client CDN:
    1.1. Feed the dCDN with chunks. Fresh chunks are injected by the CDN into the dCDN.
    1.2. Handle end-users requests, and deliver the requested contents to the end-users.
2. Implement a chunk replication policy taking account of the network's congestion level, the bandwidth available for the dCDN, the bit rates of chunks, the popularity of chunks and the storage capacities of the dCDN.
3. Monitor network and user's satisfaction
    3.1. By assessing congestion level and by monitoring link and dCDN nodes.
    3.2. By allowing the users to give feedback on their quality of experience.

In this section, we describe the functions that should be performed by the control plane. The exact implementation of the functions will depend on the results of the various WP of the VIPEER project.

4.2 Interface with the client CDN

This section briefly describes the functions related to the interface between the dCDN and the client CDN. As previously pointed out, there are two sets of functions:

- the first set of functions is related to the acquisition of content by the dCDN,
- the second set of functions is related to the treatment of users' requests.

Of course, those two sets are not independent since the objective of the dCDN is to fulfill users requests thanks to content provided by the client CDN.

The VIPEER architecture identifies a special node, referred to as "super node" or dTracker, which is in charge of offering an interface with the client CDN. There is thus some kind of peering link between the dTracker and the client CDN.

4.2.1 Content acquisition by the dCDN

The peering link between the dTracker and the client CDN is used to control the injection of content into the dCDN either via this link or via other links under the control of the dTracker.

The role of the dTracker is to control the reception of content pushed by the CDN and to pull new content from the CDN into the dCDN.

For instance, if a new content becomes popular and is missing in the dCDN (because it has been discarded via the replication policy in force in the dCDN), the role of the dTracker is to request the corresponding chunks.

To some extent, the management of content in the CDN and the dCDN are asynchronous. Since the storage capacity of a dCDN is much lower than that of a CDN, it is very likely that some content could be discarded in the dCDN and hence, the CDN could believe that the content is still available in the dCDN while it has been discarded. Hence, upon the redirection of a request, the dCDN could request again the desired content. It may also possible to request chunks with other bit rates, either because the network becomes congested or because higher bit rates are requested by the end user (HD videos for instance).

The global picture of a dCDN is represented in Figure 4. In the early development of the VIPEER demonstrator, the tracking function shall be achieved by a single, centralized dTracker and content shall be injected into the dCDN via the peering link. The dTracker thus receives fresh content that is then dispatched in the dCDN via control functions detailed later in this document. In the general case, several trackers may be implemented in order to distribute the tracking function and to avoid specific bottlenecks in the networks.

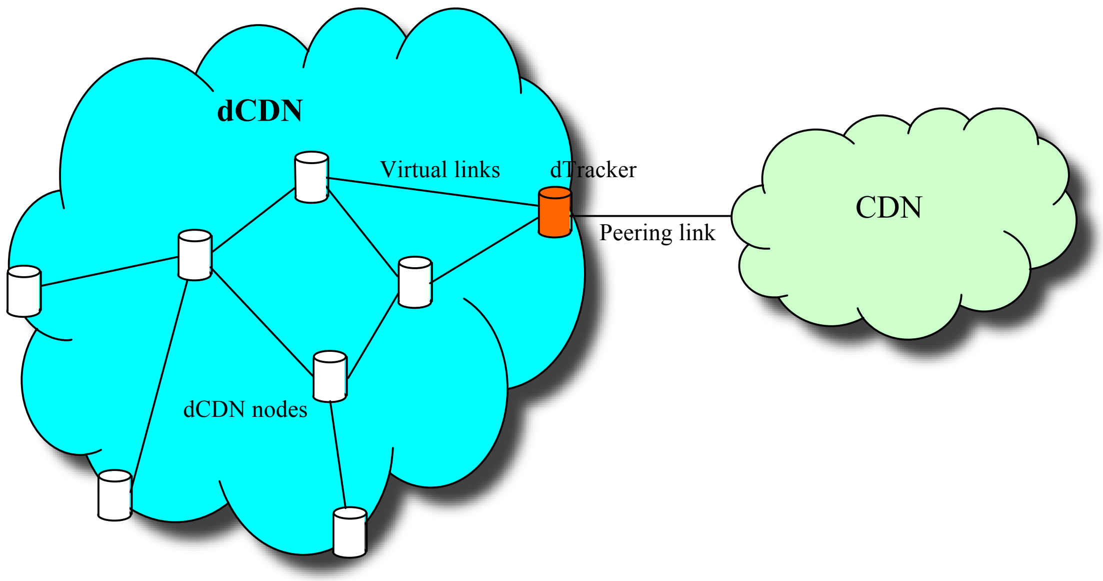

Figure 4 : illustration of a dCDN with a single dTracker

4.2.2 Handling users' requests

In order to allow a user to request a specific content, some kind of signaling has to be supported by the control plane.

Classical caching policies rely on nodes intercepting user requests (e.g., HTTP request) and then searching for the desired content in caches. In this case, there is no explicit signaling between the user and the caching engine. This is very close to a classical caching solution except that the ingress node performs the content search in place of the end user.

In the framework of the VIPEER project, we shall not consider the case based on request interception. This choice is motivated by several reasons.

- The dCDN (and the operator) should not stand between the end user and the content provider (and the CDN).
- While interception was quite easy to perform for web pages at the beginning of the Internet, the proliferation of various types of content and encryption techniques make this operating mode today much harder to implement.
- For a proper implementation, the dCDN should be aware of the naming scheme, which could be proprietary to the content provider.
- The situation is even trickier when encryption techniques are used between the user and the content provider.

This is why VIPEER proposes to facilitate business relationships by ensuring that the content provider remains the unique partner of the end user. The content provider then redirects the request to the CDN, which may ultimately forward the request to the dCDN.

As we have chosen to base video distribution on HTTP streaming, we assume in the following that the content provider processes users' requests by translating each request into a "manifest" identifying successive chunks composing the video, where several chunks (corresponding to different codecs or bitrates) may correspond to a single temporal segment. The manifest is then forwarded to the dTracker.

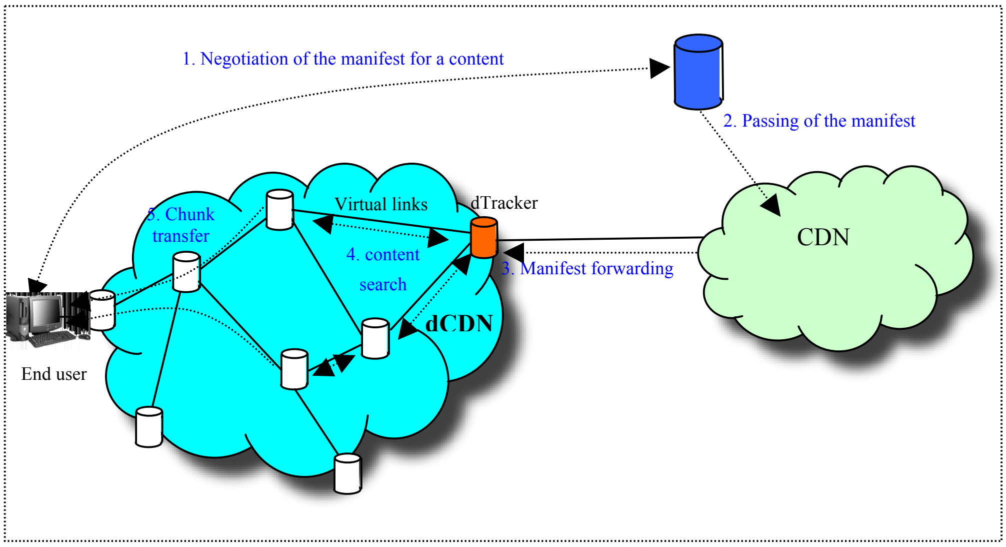

Figure 5 : Manifest negotiation and content search

Once the manifest is received by the dTracker, several options can be envisaged to associate chunk identifiers with actual storage locations:

1. The dCDN nodes can share a distributed hash table (DHT), indicating which dCDN node stores a specific chunk in a given format. The dCDN nodes and the user share this DHT, and the user can download the successive chunks thanks to the manifest. In the terminology specific to peer-to-peer networks, end users thus appear as "free riders" since they are not requested to store data in their own premises, but can download data from the dCDN nodes that store the chunks.
2. A set of nodes can collectively implement a tracking function, as e.g. in BitTorrent. The role of the tracking function is to forward the manifest to the end user with the

URL or the IP addresses of those dCDN nodes storing the desired chunks as shown in Figure 5. For example, in a completely centralized architecture, the tracking function could be realized by the dTracker. However, it is also possible to distribute the tracking function on a larger set of servers. The global dCDN could be divided into areas, each area being controlled by a server, collaborating with the dTracker to identify the appropriate url/IP addresses.

4.3 Chunk replication control

This functional block is central in the performance of the VIPEER architecture.

Once a chunk is acquired from the client CDN, it is potentially replicated and stored in one or several dCDN nodes. For example, it seems natural to replicate the chunks of a popular video in several dCDN nodes in order to efficiently serve all geographical areas. Bandwidth cost shall then be limited, but counterbalanced by storage costs. A less popular video could be stored in a smaller number of dCDN nodes, thus limiting the storage costs, but potentially increasing bandwidth costs.

Clearly, the design of optimal or at least efficient chunk replication policies has to take account of both bandwidth and storage costs.

The selection of dCDN to store chunks should take account of available resources (in terms of bandwidth and storage space).

Furthermore, chunk replication control has to specify how chunks are erased from the dCDN. A popular chunk should not be erased, whereas a chunk that has not been requested for a long time could be erased to make place for more popular chunks.

At this stage of the VIPEER project, it is envisaged that chunk replication control consists of both off-line computations that attempt to globally optimize chunk availability, and on-line procedures used for selecting the dCDN where newly acquired chunks are stored. The former computations are centralized whereas the latter procedures could be either centralized (e.g. within the dTracker) or distributed (e.g. to take advantage of geographical distribution of both demands and chunks).

4.4 Network monitoring

A dCDN is an orchestration of many network functions such as storing video chunks on pieces of equipment and transmitting them to clients. To achieve the desired gain and efficiency of the proposed data distribution architecture, one cannot neglect quality of service, quality of experience and traffic conditions in the network. Hence, we propose that the control plane of the dCDN benefits from monitoring tasks, which expected gains and complexity are described in this paragraph.

The network-monitoring component is very interesting for many reasons. On the one hand, it increases the benefits of the CDN and the network service provider that implements the dCDN. Indeed, it controls the usage of network resources (used bandwidth, impact on concurrent traffic) in the network that operates the dCDN. Hence, one can easily decrease the cost of deploying the dCDN. On the other hand, controlling the quality of service and the quality of experience allows better distribution policies so that the satisfaction of clients is improved and their trust to the offered service grows from one experience to another.

Deploying a monitoring architecture can however be complex and its benefits must be compared against the complexity of the monitoring process. In the VIPEER project, monitoring will be a key component of the control plane of the dCDN. In fact, increasing the number of the points of measurement and the number of measured metrics can increase the complexity of the monitoring process and make its scalability a major concern. Monitoring traffic (measurement probes, measurement reports, etc) must be limited so as not to increase global network traffic. Otherwise, the gain obtained by monitoring is not guaranteed.

To take benefit of the monitoring process, the monitoring architecture must be made as simple as possible and the selected monitoring protocols must be simple and traffic friendly. This is the objective of the monitoring architecture we describe in the following paragraphs.

Ensuring a good quality of experience has become a major concern for ISP/CDN operators to meet users' needs. Indeed, because of the flattening of prices, quality has become the major key to make a difference between different providers. Note that monitoring quality of experience (QoE), which is a measure of user perception, will not only serve as a global indicator of customer satisfaction to make decisions for the long term (i.e., network planning) but could also help react in real time to prevent any quality degradation.

A dCDN ensures mainly the two following tasks: storing chunks and orchestrating their distribution. In this paragraph, we underline the relationship between each of those tasks and potential monitoring functions.

4.4.1 Benefit of monitoring for the storage of video chunks

Monitoring can help determine the best storage strategy by selecting the dCDN nodes in which to store chunks. Different monitoring actions can be performed:

- One can for instance measure the available bandwidth on each of the dCDN nodes in order to select the node with the highest bandwidth. In the case when the piece of equipment where we want to store a chunk is a setup box (a terminal, a client), measuring the available bandwidth is not sufficient since clients can have concurrent applications that need guaranteed quality of service (real time applications). In order not to disturb the user activity, we propose to classify the ingoing traffic of a client in order to understand its composition and take the good decision on whether sending chunks or not and in the positive case, select the good uploading rate.
- One can use the QoE metric measured at terminal level in order to help determine the optimal quality of the chunk to be received. As the QoE degradation is usually a direct result of network traffic conditions, it is thus useful to either maximize the storage of frequently used chunks (i.e. caching policy) in the path concerned with such degradation or to take a decision of peer switching.

Another important feature related to storage in the dCDN architecture is the content-oriented caching policies. In order to optimize the delivery of content, we can use prefetching approaches, as shall be assessed in WP4. As the technique is to anticipate user's requests and push into the dCDN content that is to be downloaded, we need to collect user's behavior. We propose here to analyze user's requests history and provide algorithms to predict which content will be further requested. We thus need to collect traffic and store it in a Collector. The dCDN will then use the output of the prefetching module to organize the prefetching storage. The prefetching module may be also hosted in the Collector.

4.4.2 Benefits of monitoring for distributing video chunks

Chunk distribution by the dCDN must present a good and stable quality of service (and hence quality of experience) to the user that has requested the video. As the network conditions are changing, one needs to track the QoS/QoE perceived by users in real time to react rapidly to network conditions change and to prevent any degradation of the perceived quality.

When the dCDN gets the information that the QoE perceived by a user is going to be degraded:

- It can change the dCDN node from which the content is downloaded in order to keep the same video rate.
- When e.g. no other dCDN node with the same quality is available, it is possible to reduce the video rate (i.e. change the chunk) to better fit with the current available bandwidth. Indeed, reducing the video quality may directly impact the video interruptions, which has a worse impact on the perceived quality than the compression factor.
- Changing both the rate and the dCDN node is also an option.

Note that similar decisions can be taken when the video rate can be increased, in case of improvement of the available bandwidth or the availability of the chunks at new dCDN nodes.

To take the good decision, the monitoring task must analyze the causes of QoE's degradation by considering other monitoring metrics (e.g., by localizing congestion). When selecting a dCDN node from where to transmit chunks, one can measure the available upload bandwidth in order to be sure that the selected video rate fits into the bandwidth. Furthermore, if the node is a terminal (setup box), knowing the composition of traffic at the application level can help to understand the QoS requirements of the applications. The use of such information allows a seamless support of chunks distribution by the end users who are delivering the content.

When having the possibility to choose among many neighbors, another metric beyond available bandwidth is the delay between neighbors. Selecting the nearest neighbor leads in most cases to the selection of a neighbor from the same area. Nevertheless, if the content is not available in that area, we do not exclude the possibility to retrieve it from remote areas of the dCDN. Indeed the network service provider that operates the dCDN wants to avoid downloading content from the CDN if possible in order to reduce inter-domain traffic.

4.4.3. Measurement methodology

In this paragraph, we define the metrics we need to track and the measurement methodology to retrieve them. From the monitoring requirements discussed in the previous section a series of metrics have emerged as required:

1. download/upload available bandwidth,
2. composition of traffic at the application level,
3. perceived QoE (expected MOS),
4. delay related metrics (Round Trip Delay, One-Way Delay, Jitter, etc...),
5. users request logs (timestamp, userID, contentID, etc.).

Two types of measurements can be performed to obtain the values of the above metrics: active measurements and passive measurements. Active measurement consists of injecting packet probes into the network, which interfere with concurrent traffic. To keep the monitoring activity scalable, one needs to reduce this type of measurements. When using passive measurements, the measuring module exploits ongoing traffic to measure the desired metrics. Here are some techniques we propose to use to obtain the values of these metrics.

Passive measurements:

1. To get information on the upload/download available bandwidth on dCDN nodes, one can use a simple local interface tracking protocol such as SNMP.

2. To obtain the composition of the upload/download traffic at the application level, a traffic classification module can be implemented on terminal equipment. This module can use various techniques (Deep Packet Inspection, Statistical classification) in order to determine the application corresponding to each flow. Such a module, NicoFix, has been developed in the framework of WP2 and described in D.2.1 and D.2.2.
3. The QoE perceived by receiving users can be estimated from some statistics obtained from the player running the video (play-out interruptions, etc...). The correspondence between QoS and QoE can be done by some learning techniques. This is the approach followed for the development of the QoE monitoring module developed in WP2 and described in D.2.1 and D.2.2.
4. User's request logs can be obtained directly from the master dTracker, which receives requests redirected from the CDN.

Active measurements:

1. To measure round trip metrics (RTT, etc...) between two equipment, one can use the ICMP protocol (e.g. ping).
2. To measure One -way delays, protocols such as OWAMP (One Way Active Measurement Protocol, IETF RFC 4656) propose to send a time stamped packet from the source to the destination and compute the one way delay at the reception by measuring the arrival time. This type of measurement needs a synchronization between the source and the destination (GPS, NTP,...).

One can easily see that to obtain the matrix of delays between all dCDN nodes, one needs to send a huge amount of probes (NxN order of magnitude, where N is the number of dCDN nodes). This NxN complexity can be the source of scalability problems. As we said earlier, in VIPEER, we overcome the delay problem most of the time by splitting the set of dCDN nodes into geographic areas. To maintain the knowledge about the proximity of areas, we propose to measure inter-area delays. This information could be used by the master dTracker in order to select an area from where to retrieve a chunk.

4.4.4 dCDN partitioning and collection of measurement

From the above sections, we see that for an efficient operation of the dCDN, a lot of information has to be exchanged between the dCDN nodes. In order to avoid scalability issues and since a dCDN can be operated at a regional area, we recommend partitioning a global dCDN into areas, each area being controlled by a dTracker.

The various dTrackers controlling the various areas of a global dCDN can exchange information on the contents and a global view of traffic conditions in their corresponding areas. In the early development of the VIPEER project, we shall consider only one area (see Figure 6).

To implement the monitoring functions listed in the previous sections, a dCollector is introduced in the functional architecture. The dCollector collects all monitoring information provided by the dCDN nodes and the QoE feedback from the end users. The dCollector communicates with the dTracker in order to help the latter in selecting the nodes serving end users for a given content. The dCollector should be aware of the routing inside the dCDN. For this purpose, since the dCDN is in the same AS, the dCollector could be a dead leaf of the routing tree in the AS (e.g. similar to RouteExplorer).

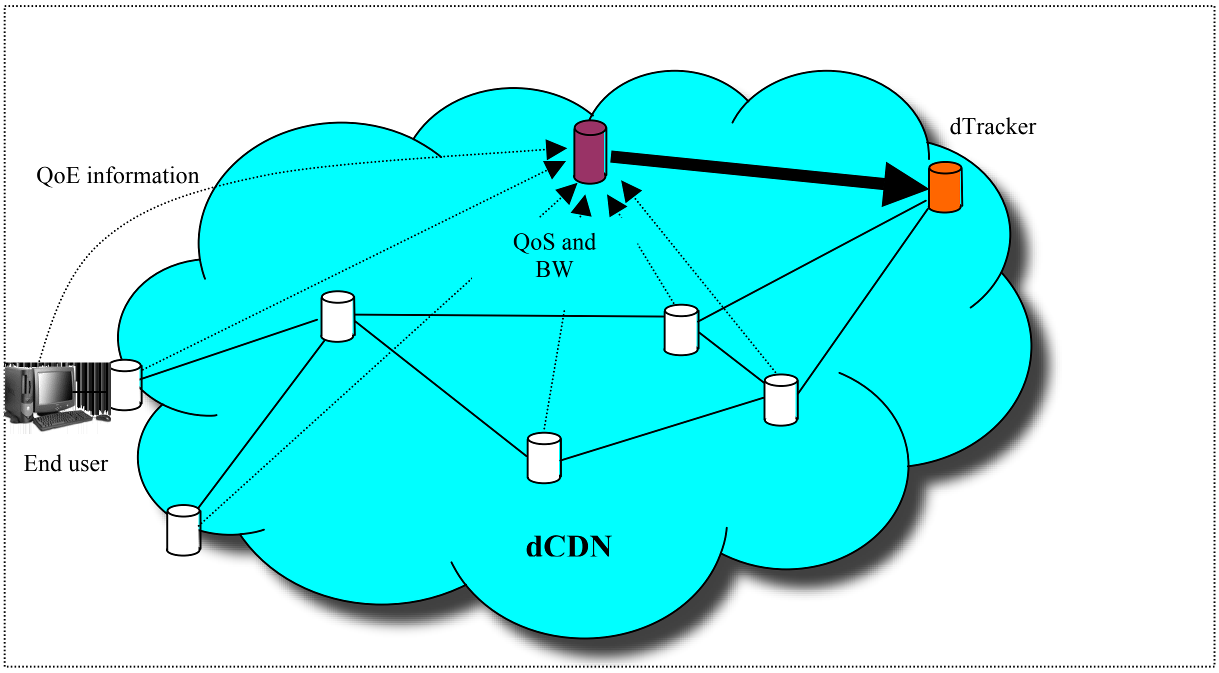

Figure 6: a dCollector collecting monitoring information

There are different method for reporting QoS and bandwidth information to a collector:

- Either report the value of some counters (number of lost packets, bandwidth occupation, etc.) as with SNMP procedures.
- Or report traffic samples in records such as with Netflow.

WP2 shall specify the exact method for reporting monitoring information.

Finally, note that beyond bandwidth and QoS, it may be interesting to report the hit ratio of the chunks contained in a dCDN node. This would facilitate the correlation between traffic conditions, topology and content available in the dCDN.

As the dCDN nodes will be partitioned into areas and each area will be governed by a dTracker at the control plane, we propose to position a single dCollector per dCDN area. Hence, each dTracker will have access and work together with a dCollector local to its area. dCollectors from different areas will not need to communicate together since trackers will play the role of gateways.

4.4.5 Content localization procedure

The dCDN nodes should exchange information on the content they store. This information should be reported to the dTracker so that it can forward a manifest to an end user with the network addresses of the dCDN containing the desired chunks.

Once the information is exchanged and maintained in the nodes, semantic routing is possible. The dTracker could have an up-to-date map of the various chunks stored in the dCDN or it could request information on the location of chunks via a poll procedure. The nodes receiving a request can forward it to ad-hoc nodes by using the "routing" table of chunks.

Nevertheless, note that there is always a discrepancy between the location map maintained by the dTracket and the actual location of chunks since the replication procedure may discard some chunks in nodes without reporting to the dTracker.

5. Mapping adaptive HTTP streaming elements to dCDN functions

Section 2 has identified 3 functions to be implemented in order to improve dynamic support of HTTP streaming architectures:

- Statistical and Contextual Elements (SCE). The role of the SCEs is to collect static as well as dynamic information at different locations, e.g. sent back by users or network elements.
- Statistical Control (SC). Based on the information collected by the SCE, a SC controls the adaptation to dynamic modifications of network state.
- Adaptation Engine (AE). The AE performs static or dynamic (in real-time) adaptation of the content. The VIPEER architecture does not propose real time adaptation, but implements static adaptation by proposing several versions of a given temporal chunk.

Network monitoring functions shall be assigned to SCEs. In the dCDN architecture, the resources associated to every potential storage location and to every customer have to be monitored. QoE perceived by customers also has to be monitored. Various types of monitoring probes shall thus be disseminated, and monitoring data shall be aggregated and processed within the SCEs. In particular, SCEs realize part of the functions associated with dCollectors (identified in the previous Section).

The elements implementing SC shall:

1. Process monitoring data collected by SCEs.
2. Control both chunk replication and caching;
3. Control the mapping between customers' requests and actual dCDN nodes where chunks are stored.

In the VIPEER architecture, these functions can be either centralized or distributed, which implies that the SC can be distributed on several network elements collaborating thanks to appropriate protocols and procedures. The first function is realized by dCollectors, and the two others by dTrackers.

6. Functions to be implemented

This section recaps all functions identified in the previous section, and maps them on both dCDN nodes and on the dTracker.

6.1 Functions implemented in a dCDN node

A dCDN node is a storage element, where chunks are stored and forwarded to users when requested, and when sufficient network resources are present. Such a node implements both data plane and control plane functions.

6.1.1 Data plane

For the data plane, we have the following Functional Blocks (Figure 7):

- Forwarding and Error Control (FEC) for multiplexing packets of chunks and Internet traffic and sharing the bandwidth among all the flows.
- Admission Control (AC) for possibly blocking the transmission of some chunks if the amount of resources is not sufficient.
- Storage FB to manage the storage capacity of a dCDN node.

6.1.2 Control plane

A dCDN node should implement the following functional blocks:

- Bandwidth, QoS and Content monitoring. When the replication policy is completely distributed, it is necessary to be able to locate a given content in the dCDN. This is why it is worth introducing some information on the content stored in a dCDN in the reporting made by dCDN nodes to the dCollector.
- Prefetching of chunks (by observing the chunks traversing a dCDN node).
- Localization of chunks and content search.

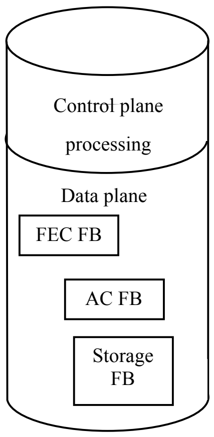

Figure 7 : Functional architecture of a dCDN node

6.2 Functions implemented in a dTracker

The dTracker is only in the control plane and implements the following functions:

- Receive manifests from the CDN.
- Update the manifest with the url or IP addresses of those nodes of the dCDN, which have the desired chunks.
- Inject new content in the dCDN

6.3 Functions implemented in a dCollector

The dCollector is only in the control plane and implements the following functions:

- Receive monitoring information provided by the dCDN nodes.
- Receive the QoE feedback from the end users.
- Interact with the dTracker in order to help the latter in selecting the nodes serving end users for a given content.

7. Conclusion

VIPEER assumes that video is distributed using HTTP streaming. In standard HTTP streaming solutions, a single server provides all successive temporal segments, while the user adapts the segment's encoding rate to the available throughput.

VIPEER enhances this architecture by duplicating and caching the various "chunks" of data in "dCDN nodes" within the operator's network, and by allowing the selection the appropriate dCDN node to download the requested chunk.

The major building blocks for this architecture are:

(1) an interface with a client CDN,

(2) a policy for caching and duplicating chunks in the dCDN nodes,

(3) mechanisms for monitoring available storage space, link bandwidth and delivered QoE,

(4) protocols for serving customers by selecting for each demand and each customer the appropriate dCDN node where the requested content is cached.

The present deliverable has identified the functions required to implement these building blocks, and their relationships.

Several points remain open in this deliverable, and shall be addressed in further deliverables:

- The role and location of Adaptation Engines within the dCDN architecture. Content protected by DRM can be transcoded only if the necessary keys are distributed to the AE. Relying on AE within the dCDN allows the reduction of the amount of storage by caching only those chunks that are actually requested by customers, instead of all available chunks referenced within the manifest. However, this requires a trust relationship between the Content Provider and the ISP, which may not exist. This problem is irrelevant for many DRM-free video contents. This point shall be addressed in D1.3.
- The procedures used for actually distributing video contents. This requires explaining how the identifiers transmitted in the manifest are used to communicate to the customers the actual location of the successive chunks. This includes the procedures used by the client CDN and the dCDN to collaborate in caching chunks and serving customer requests. This point shall be addressed in D1.3.
- The caching and replication policy. This point shall be addressed by WP4.
- The exact distribution of the monitoring elements that shall be clarified by WP2, and their interaction with the video distribution architecture. This point shall be addressed in D1.3.