Technology
CBC Radio-Canada
Technologies

CBC TECHNOLOGY REVIEW

Issue 4 - July 2007 www.cbc.radio-canada.ca

pCDN: PEER-ASSISTED CONTENT DISTRIBUTION NETWORK

Bernard Jules
Senior Project Manager, Internet/NM Tech
Strategy & Planning

Mohamed Hefeeda
Professor, Simon Fraser University

ABSTRACT

The delivery of digital programming to home users via the Internet is at a turning point as key media industry players are presently remapping their strategies by creating new platforms and adopting new content-delivery business models for the massive deployment of on-demand digital content and for generating new revenue streams. With the increasing use of high-performance desktop computing combined with the wide availability of broadband access, peer-to-peer (P2P) is being embraced by key industry players and has become a viable alternative to distributing bandwidth-intensive content, such as high-quality video streaming in a secure and cost-effective manner. CBC/Radio-Canada's Technology Strategy and Planning Department has been making significant efforts to develop the next generation of content delivery platforms to face upcoming, foreseeable changes in the way content is being delivered and distributed to end-users and to alleviate the cost of escalating demand for high-quality video.

In partnership with the Simon Fraser University Network Systems Lab, CBC/Radio-Canada is investigating the P2P computing paradigm beyond file-sharing applications by testing a large-scale content distribution system for high-quality video streaming, both live and on-demand, capable of generating significant cost savings.

The intent of this article is to share with our audience some of the challenges being addressed while developing the P2P platform, such as: (i) optimizing video quality for heterogeneous clients; (ii) efficient video transmission from multiple senders; (iii) reducing the traffic load on Internet service provider (ISP) networks; (iv) seamless integration with digital right management (DRM); and (v) restricting content to certain geographical locations (geofencing). We also discuss various business models for P2P content distribution.

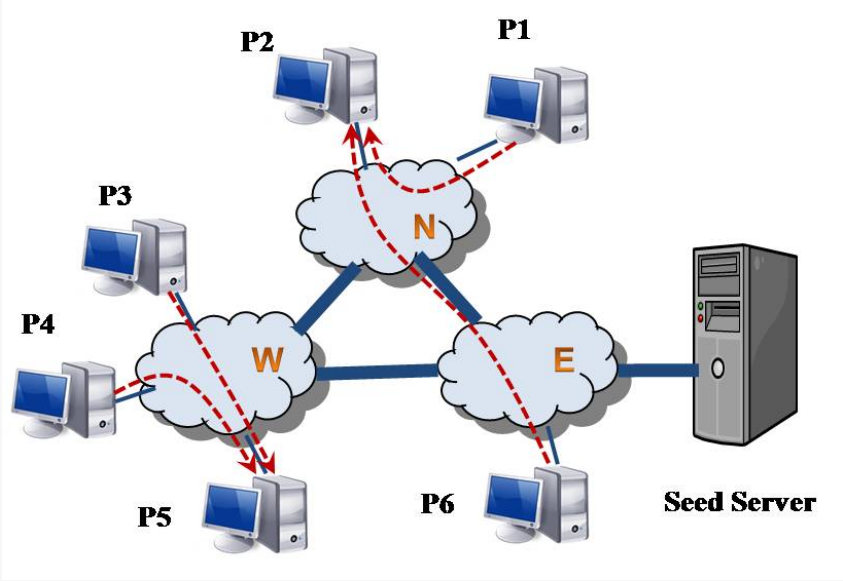


Figure 1. Architecture of the Peer-assisted Content Distribution Network

INTRODUCTION

Consider a large-scale content provider, such as the Canadian Broadcasting Corporation (CBC/Radio-Canada), which distributes multimedia objects to millions of clients scattered all over the Internet. One option for the content provider is to build on the traditional client-server model by establishing a gigantic server (or server farm) with huge network connections in the order of hundreds of megabits per second to the Internet. Clearly, this option is neither scalable nor cost-effective because the number of clients is rapidly increasing and they are demanding more and higher-quality content. The second option for distributing content is to use a content delivery network (CDN), such as Akamai and Digital Island. Although CDNs are scalable and yield a better performance, they are fairly costly, as they charge for each megabyte served to clients.

Peer-assisted content distribution networks (pCDN) provide the third, and most viable, option for distributing content. pCDN capitalizes on the tremendous success of the peer-to-peer (P2P) computing paradigm. pCDN is a promising approach that can alleviate much of the costs incurred in content distribution, because it provides: (i) improved scalability by aggregating resource contributions from peers and reducing the reliance on centralized servers; (ii) reduced cost by utilizing already-deployed resources and minimizing the need for expensive infrastructure; and (iii) rapid deployability by performing all processing at the end systems. Currently, there is a significant interest in the academic and industrial environments for developing peer-assisted content distribution systems.

For instance, Warner Brothers recently made a deal with BitTorrent to *sell* more than 200 films over BitTorrent's P2P network [Newsweek, 29 May 06]. Moreover, major content distribution networks, such as Akamai, consider the P2P paradigm a real threat to their content distribution business. This is because the P2P paradigm may achieve similar services at a fraction of the cost. Furthermore, recent research indicates that peers (end-user machines) have enough resources to support large-scale streaming systems.

The core idea of pCDN, as illustrated in Figure 1, is to have clients (peers) help each other in receiving multimedia content. Therefore, clients not only consume resources, but they also contribute them, making the system inherently scalable. Ideally, the content provider will only need to deploy a few *seed* servers to manage peers and to introduce new content into the system. At a high level, a peer-assisted content distribution system works as follows. A client requesting an object contacts one of the seed servers. Instead of serving the complete object, the seed server redirects the client to other peers who recently received (or are still receiving) that object. The requesting client contacts this potential list of sender peers and starts receiving the object from a subset of these peers. With additional technical details, this simple idea has worked well with widely deployed *file-sharing* P2P systems, such as BitTorrent. There remain, however, several research challenges to be overcome in order to employ the P2P computing paradigm in a commercially viable multimedia content distribution system.

In this article, we describe how pCDN can provide high-quality *multimedia streaming* to a large-scale user community in a cost-effective and robust manner. Multimedia streaming is more challenging than the widely deployed application that is P2P file sharing. In file-sharing systems, a client downloads the *entire* object before using it. This takes a long time for large multimedia objects. For instance, a 1-GB file (approximately a half-hour video clip) may take more than 90 minutes to download over a 1.5 Mbps DSL link. This is clearly an unacceptable amount of time for a client waiting to start viewing a video clip. In contrast, streaming systems overlap data streaming with the playback and therefore enable clients to start viewing video clips after a small (order of seconds) buffering delay. Streaming systems thus impose strict timing constraints on data transmission. Therefore, efficient data transmission algorithms need to be designed. These algorithms should consider the limited resources contributed by peers and the dynamic nature of the network. In addition, in streaming systems, video objects can be encoded and streamed at different rates to accommodate receivers with heterogeneous network and computing resources. This flexibility is accompanied by the problem of determining the best quality for each receiver constrained by available resources at the receiving and sending ends.

Furthermore, it is crucial that the pCDN system does not overload the Internet service provider (ISP) networks. Otherwise, the ISPs may filter or downgrade the traffic belonging to the pCDN systems and adversely impact the client-perceived quality. Lastly, the pCDN system should provide the means to enforce digital rights management (DRM), i.e., only provide content to legitimate clients. In the appendix, we summarize all the features of the proposed pCDN system and compare it against the common P2P file-sharing systems.

We start the next section by describing our current prototype system, which was developed by researchers at Simon Fraser University and tested on CBC/Radio-Canada's network.

We then present, in Section 4, the research problems and future extensions that we are currently working on. In Section 5, we present different business models for P2P content distribution systems. We conclude the article in Section 6.

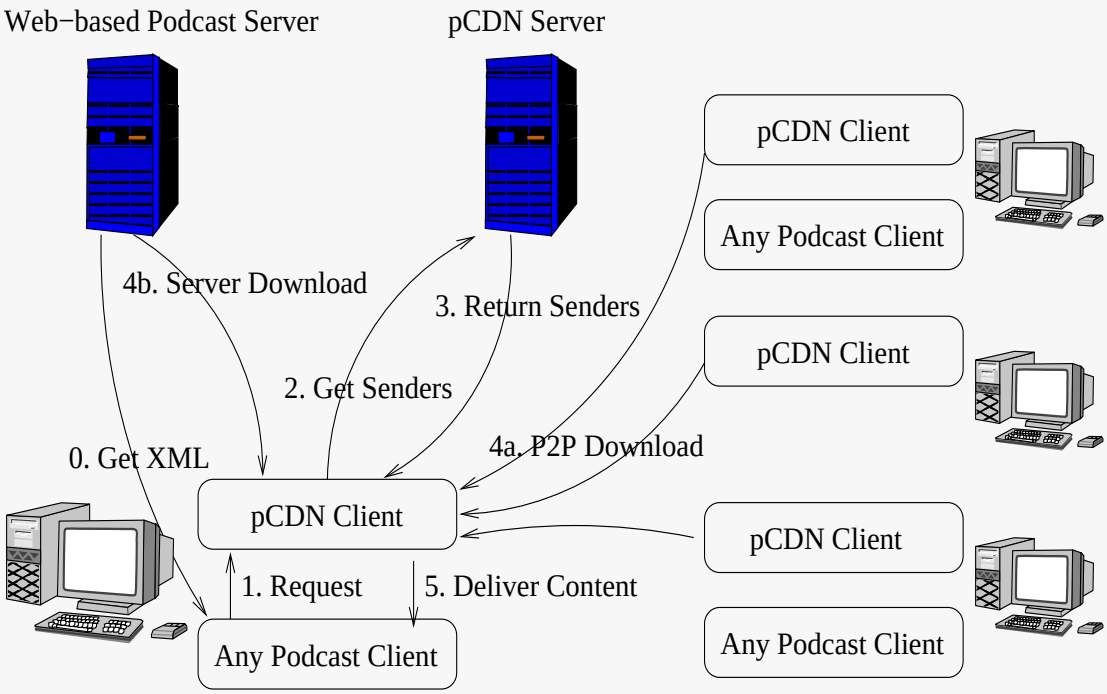


Figure 2. Operation of the Peer-assisted Content Distribution Network

CURRENT PROTOTYPE SYSTEM

A pCDN system has been tested at CBC/Radio-Canada for its podcast services. Preliminary testing took place in January 2007, both at the Simon Fraser University Network Systems Lab and at CBC/Radio-Canada's Vancouver facilities using Radio3 podcast content. The prototype was tested on Windows, MAC and Linux clients.

Podcast service is on-demand radio that allows users to choose various radio channels and listen to them on computers or portable multimedia players. Existing podcast services employ one or more centralized podcast servers and directly supply multimedia content to individual users. Podcast service providers publish feed files that announce available podcast episodes to users. An episode is simply a multimedia file that can be streamed or downloaded to users using podcast client software, such as iTune. Users select interested feed files and subscribe to interesting podcasts using podcast client software. Podcast client software parses the subscribed feed files, presents details on each episode to users, and enables users to choose interesting episodes.

Once the selection is made, podcast client software pulls multimedia files from a podcast server using the HTTP protocol. Podcast service is an excellent candidate for peer-assisted systems, such as pCDN, for a number of reasons: (i) podcast episodes are huge and result in tremendous server workload and bandwidth consumption; and (ii) podcast episodes are published periodically, for example, once a week, and the latest episodes often have extremely high popularity, as users are eager for up-to-date contents.

Figure 2 illustrates the proposed peer-assisted system, which consists of four components: (i) a Web-based podcast server; (ii) a podcast client; (iii) pCDN server; and (iv) pCDN client. The first two components have been used in client/server-based podcast services. The Web-based podcast server stores and, if necessary, serves episodes to podcast clients. It also provides feed files (in XML format), which announce available episodes to podcast clients. A podcast client obtains XML files and provides a description of available episodes to users. Unlike client/server-based service, a podcast client downloads podcasts from a pCDN client running on the same computer instead of a Web-based podcast server. A pCDN client intercepts podcast requests and serves them back to the podcast clients on the same computer as follows. It first sends a query to the pCDN server to locate pCDN clients that have the requested episode. It then fetches the episode from these pCDN clients. A pCDN client may choose to download that episode directly from the podcast server if it is not available from other pCDN clients. Lastly, a pCDN server maintains a database of episodes that have been previously advertised by pCDN clients. Based on this database, the server provides a list of potential sender pCDN clients that contain a specific episode to requesting receiver pCDN clients.

Here is a description of a download operation. A pCDN client receives a download request from a podcast client on the same computer. To serve this download request, the pCDN client seeks the best senders by sending a query to the pCDN server. The pCDN server locates the most suitable senders and returns them to the requesting pCDN client. This potential sender list may be empty, which indicates this episode is not yet available on the P2P network. The requesting pCDN client may connect to those senders (which are also pCDN clients) and fetch the podcast from them. Alternatively, it may download the podcast from the podcast server if the P2P network cannot meet this request. For example, a fairly new podcast may only exist in a few pCDN clients. Eventually, the pCDN client forwards this episode to the podcast client on the same computer.

We notice that adopting this pCDN system is transparent to both content providers and users: Content providers can use the same podcast servers without reconfigurations, while users can use any standard podcast client software as before. We are implementing pCDN server and client using Java programming language. The pCDN server uses an efficient hash table to maintain a list of episodes. Each episode is associated with pCDN clients that have or are downloading it. Upon receiving a request for a certain episode, the pCDN server will return up to ten pCDN clients associated with this episode. pCDN server carefully selects potential senders based on the status of their access link and on their joining time so that the most robust and the most recently joined peer is selected first. pCDN server also intelligently chooses potential senders to support Network Address Translation traversal (NAT-traversal). Network Address

Translation is an Internet protocol, which is used to hide the internal IP address from the Internet. To maximize transfer speed, a NAT-traversal algorithm is critical, as too many users have wireless networks at home and therefore are behind NAT devices. pCDN client runs on the client computer and is transparent to users. It supports concurrent episode downloads. It sends a request to pCDN server for potential senders. A receiver pCDN client attempts to connect to at most six pCDN clients from the returned potential senders. A pCDN client becomes a sender if a connection is successfully established; otherwise it is marked as unavailable. The pCDN receiver uses a weighted round-robin packet scheduling algorithm to request episode pieces from senders. This scheduling algorithm enables pCDN client to adapt to dynamic network and sender conditions, thus maximizing transfer speed. Upon receiving a piece request message, a sender starts sequentially pumping pieces to the receiving pCDN client at a constant rate controlled by leaky-bucket algorithm. The leaky-bucket algorithm maintains quality of service without causing network congestion and client buffer overflow. When the next piece of episode is not available, the sender prunes the connection to the receiver. The receiving pCDN client shuts down this connection and turns to other senders. Similarly, if any network exceptions are encountered during the transmission, the receiver seamlessly closes the connection and reruns the scheduling algorithm among active senders. This exception-handling mechanism enables pCDN client to recover from network errors. Lastly, the receiver pCDN client maintains a minimum number (a configurable parameter) of active senders. The receiving pCDN client switches over to HTTP download mode if there are too few available senders in the network by fetching the file from a podcast server.

RESEARCH PROBLEMS AND FUTURE FEATURES

In this section, we briefly describe several research problems that need to be solved in order to realize the full potential of pCDN systems. We outline our approach to solving these problems. All the features of pCDN systems and the research problems relating to these systems are summarized in the Appendix.

OPTIMIZING VIDEO QUALITY FOR HETEROGENEOUS CLIENTS

To provide high-quality and to accommodate receivers with heterogeneous resources, we propose employing fine-grained scalable (FGS) video coding. FGS coding has been proposed as part of the recent MPEG-4 and H.264 standards. An FGS-encoded video sequence consists of two streams: a nonscalable base layer, which provides basic quality, and an enhancement layer, which adds incremental quality refinements proportional to the number of bits received. FGS enables the enhancement layer to be truncated at any bit location, thus providing fine control of the stream's bitrate. This fine control of bitrates enables the streaming system to maximally utilize the bandwidth of receivers in order to provide the highest possible quality. FGS encoding also provides great flexibility and efficiency to streaming servers. Servers encode the video only once with the highest quality, and they can easily control the bitrates of various streaming sessions by truncating the bitstream at appropriate locations. Truncation is a simple operation, which can be done in real time.

The fine-grained nature of video encoding, the limited computing and network resources of peers, and the dynamic nature of P2P networks make providing high-quality and reliable streaming services a challenging research task.

These conditions require, for example, that the system concurrently stream from multiple senders, as well as maintain a set of backup senders, which raises a number of issues including: (i) determining the average size of the active sender and backup sender sets that yield good quality and can tolerate peer failures; (ii) carefully choosing the active senders to reduce the load on the backbone network and reduce quality fluctuations; (iii) allocating streaming rates and data segments to active senders; and (iv) handling sender failures and providing seamless transition from backup senders to active senders. Some of these issues were partially resolved in our earlier work.

EFFICIENT VIDEO TRANSMISSION FROM MULTIPLE SENDERS

Since video objects are large in size, they have to be segmented. Segmentation also enables a client to receive different portions of the same object from different senders at the same time. Unlike file sharing, P2P streaming imposes timing constraints on the data transfer from the senders to the receiver. In streaming, data arriving at the receiver after its playback time is essentially useless. This means that every segment should have a deadline for arrival at the receiver. The more segments missing deadlines, the worse the user-perceived quality will be.
We are designing segment transmission scheduling algorithms to minimize the number of segments that miss their deadlines. As a first step, we are employing a greedy scheduling algorithm, in which we transmit the segment with the earliest deadline first from the "closest" sender that has this segment. There are a few variations of this algorithm. For instance, instead of choosing the closest sender, we may choose the sender with the highest bandwidth or the most reliable (based on the past history) sender.

Segment transmission scheduling from multiple senders to one receiver is similar to the problem of job scheduling on parallel machines, which may take a prohibitively long time to determine. Hence, absolute optimal schedules may take a prohibitively long time to determine. However, approximation and heuristic algorithms with good performance have been proposed in the literature. We plan to explore the adaptation and application of some of these algorithms to the segment scheduling problem. In addition, we will design new approximation algorithms optimized for the segment scheduling problem in dynamic P2P environments.

For the object segmentation problem, we will start with a simple segmentation scheme. Every object is divided into equal-sized segments. While simple, this segmentation may not provide optimal performance in terms of playback quality, buffer size, and storage overhead. We will design better segmentation schemes and assess their performance gain as well as their complexities. The new segmentation schemes will depend on object size distribution, relative object popularity, object type (audio, video), object encoding, and client behavior. The interaction between the object segmentation schemes and the segment transmission scheduling algorithms, and the impact of that interaction on the user-perceived quality will be analyzed and experimentally investigated.

PROTECTION AGAINST ISP NETWORK OVERLOAD

Multimedia content distribution imposes a huge load on the backbone networks owned and operated by ISPs. Thus, ISPs bear nontrivial costs to carry multimedia traffic. A large fraction of this cost is incurred when the traffic crosses inter-ISP links. This is because ISPs pay significant charges for these links.

Notice that ISPs typically have abundant bandwidth inside their network. Therefore, it is critical to transfer multimedia traffic while minimizing the load on inter-ISP links. For example, serving peer P5 from P3 and P4 inside the ISP labelled W in Figure 1 is better than serving P5 from any other peers, because this session will not consume expensive inter-ISP link bandwidth.

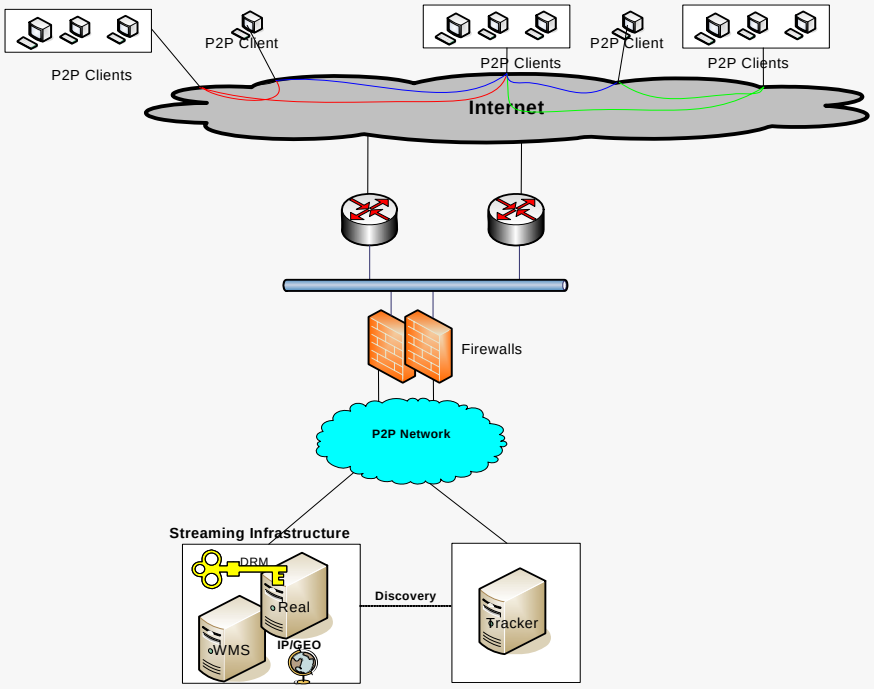
To reduce the load on ISPs and avoid their potential reactions to P2P traffic, we propose network-aware matching of senders with receivers. In addition to reducing the load on ISPs, network-aware matching enhances overall system performance. For example, a client located in Vancouver cannot avoid the slowness of coast-to-coast connections if it is being served by senders at the east coast. We are designing network-aware matching algorithms that consider two types of inputs: (i) static metrics, such as network topology and distance between clients; and (ii) dynamic metrics, such as network condition and client healthiness. For any given content and requesting client, this algorithm produces a list of senders that are in the best position to achieve the highest transfer rate and thus the best user experience.

INTEGRATED SUPPORT FOR GEOFENCING

On many occasions, content providers may need to restrict the distribution to certain geographic locations. For example, CBC/Radio-Canada has the right to broadcast the 2008 Beijing Olympic in Canada. Thus, CBC/Radio-Canada would need to restrict access to its online content to Canadian clients. We call this feature "geofencing". We are currently developing algorithms and data structures to provide geofencing to large-scale content providers in an efficient manner. By "efficiency", we mean that the decision to grant or deny a session to a client should be done very quickly, i.e., in the order of milliseconds, and should impose low overhead on the processing servers so that they can scale to millions of requests per second. Our approach is based on commercially available databases (such as the one from MaxMind) that map an IP address of a client to its geographical location (country, province and city).

P2P INTEGRATION WITH DIGITAL RIGHTS MANAGEMENT (DRM)

DRM allows content creators, providers and distributors to keep ownership of their content by controlling the rights to its usage. The intent in integrating with a DRM module in the current research is to take advantage of the P2P platform to deliver content that is free of charge and rights-protected. Later in this article, we will look at the different business models for P2P as well as how DRM and P2P can be used to meet the needs and interests of different consumers.

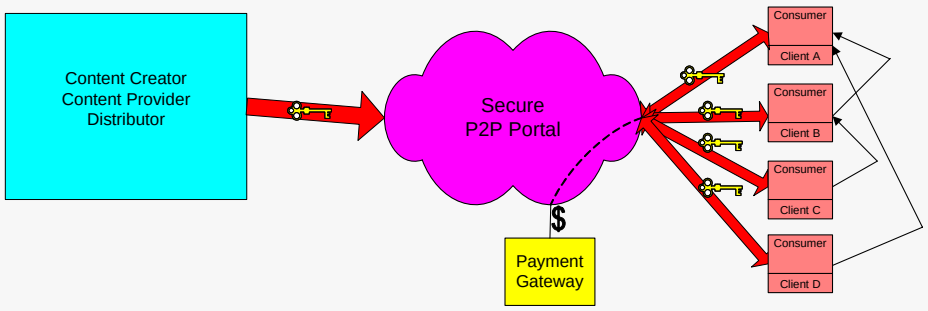


BUSINESS MODELS FOR P2P CONTENT DISTRIBUTION SYSTEMS

P2P has the potential to be the most utilized Internet delivery model, in particular, for digital content. However, in order for P2P to be adopted and become popular with media companies, it must find a viable revenue-generated model capable of sustaining its long-term growth. In this article, we present four models. The first one is a business-to-client (B2C) model supported on its own with a paid content offering. The second is also a B2C model, but it is supported only by advertising. The third model is a B2C free-access, content-based model that is also supported by advertising. The fourth model is a C2C (Client-to-Client) ad-supported model presented in the context of Web 2.0. Also, all of the models enable the content to be geotargetted by country, province or city.

P2P-B2C DRM-PAID MODEL

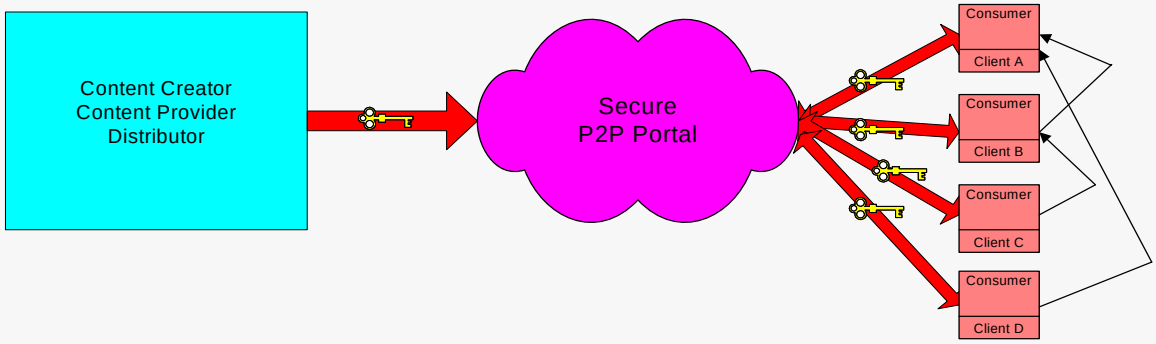
In this model, client A connects to the P2P portal and selects a particular DRMed program for downloading. The DRM server, before providing this client with a user license for download, directs it to the payment gateway so that it can pay for the content. Once the payment transaction is completed, client A gets its license with an authorization to download the content, together with the associated viewing and copyrights. Now client B likes the content of client A and downloads it. An attempt by client B to view the content will direct it to the payment gateway. Should clients C or D decide to download the same content, any attempt to view the file will lead them to the same process as client B.



The advantage of this model is that it can offer the same content available in physical stores at a reduced price. It is able to do this because it does not have to bear any overhead costs for either bandwidth or a large physical infrastructure. In this model, consumers can also subscribe to the portal and pay a monthly fee for a fixed download.

P2P-B2C DRM AD-SUPPORTED MODEL

The second model is also a B2C model, but it is supported only by advertising. The main difference between it and model 1 is the fact there is no payment transaction. If client B downloads content from client A and wants to view it, instead of directing client B to the payment server, client A directs client B to the DRM server to get a user license with the associated viewing rights, together with an authorization to download.

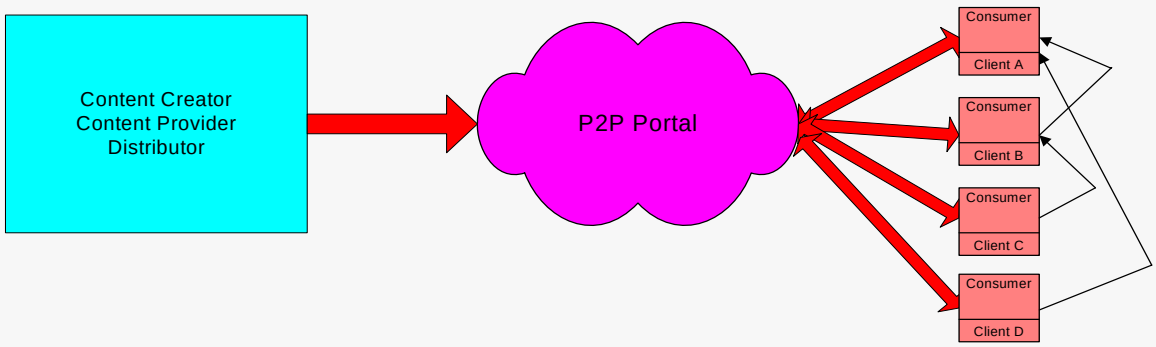


Additional peers (client C or D) that download the file and want to view it will go through the same process as client B.

In addition to providing the same economic benefits as model 1, it is aimed at increasing the P2P uptake rate by offering free content to the consumer, while at the same time protecting the rights of content owners.

P2P-B2C AD-SUPPORTED MODEL

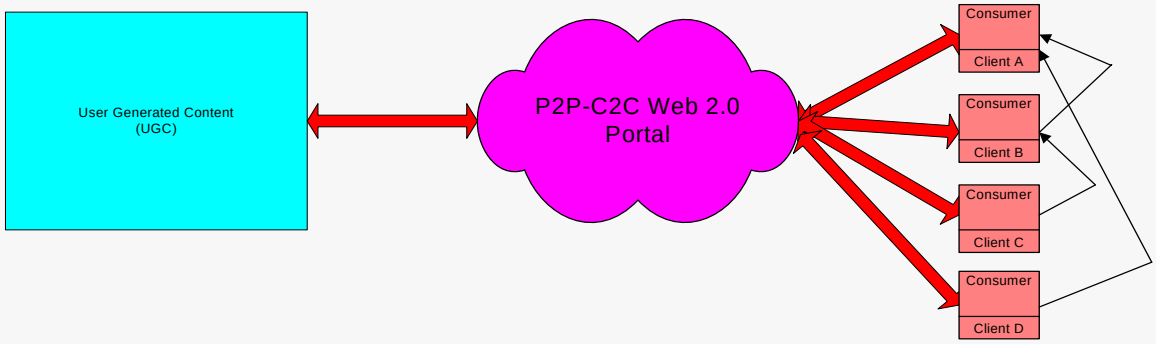
The third model is a B2C model supported by advertising. It differs from the two previous ones in that it only offers content that is free of rights.



Once a client downloads a file, they will be directed to the best available client that has the file.

P2P-C2C UGC AD-SUPPORTED MODEL

The fourth model is a C2C (Client-to-Client) model supported by advertising. It offers only user-generated content (UGC), and the content is free of rights.



In the UGC model, client A connects to the P2P Web 2.0 portal and downloads a video file. Now, client B likes the content of client A and downloads it. An attempt by client B to view the content will connect it to the Web 2.0 portal, thereby allowing it to provide comments and recommendations.

SUMMARY AND CONCLUSIONS

The explosive growth rate for on-demand content, such as video and podcasts, that we have been witnessing for the past two years is more than likely to continue. CBC/Radio-Canada New Media components as well as other content providers will continue to be impacted by the high costs of delivering content to consumers. The arrival and instant popularity of collaboration Web sites, such as Wikipedia, Youtube, MySpace, Facebook and others, together with more user-friendly and collaborative instant messaging systems, are further indications of future trends and of how things will develop. Internet users now expect to access their content wherever they happen to be and on whatever devices they have in a secure manner. This explosive growth will continue to impact both live and on-demand content and is currently making the traditional client server model cost-ineffective for every stakeholder involved in the value chain, from content creators to content consumers. One of the practical ways to evolve beyond the traditional client-server model and to overcome its costs inefficiencies is to adopt a model such as the peer-assisted content distribution network (pCDN).

In addition to the cost savings, the current model as presented in this article, once completed and deployed, is expected to provide the following additional benefits to CBC/Radio-Canada:

- Provide high-quality live streaming of audio and video content.
- Offload repetitive traffic from the origin server to peers.
- Reduce the traffic load on ISP upstream connections.
- Minimize end-user latency.
- Provide secure content delivery by protecting the rights of content owners when used with DRM.
- Provide geofencing of content when used in combination with the IP/geolocation database.

For the pCDN model to yield these benefits, it must adopt a business model that can: (i) protect the rights of content owners; (ii) provide a sustainable return on investment; and (iii) encourage the implementation of network-aware matching algorithms to prevent overloading of ISPs' backbone networks.

As for its immediate future, there is no doubt that the pCDN model is bound to become one of the major methods of content delivery from business to consumer (B2C) and from consumer to consumer (C2C). The rapid evolution of high- performance client computing and the fast pace of broadband deployment are two of the key drivers that will boost its uptake rate, both among businesses and consumers.

APPENDIX: P2P STREAMING VERSUS P2P FILE SHARING

This appendix summarizes the differences between the proposed P2P streaming system and the common P2P file-sharing systems. It highlights the features that will be supported by our system. Currently, there is no P2P content distribution system in the literature that integrates all of these features.

Issue	P2P Streaming (our work)	File Sharing (e.g., BitTorrent, Kazaa)
Delay	A few seconds. The client starts viewing the video buffering a small amount of data. Viewing is overlapped with downloading.	Minutes. It depends on file size. Could be hours in the case of large video files. The file is not usable until fully downloaded.
Streaming quality and robustness	Ensures video quality by controlling the senders' sending rate. Uses path diversity to avoid correlated failures. Optimizes the structure of scalable video streams.	Not supported because it tries to shorten total download time, not optimize video quality.
Digital rights management (DRM)	To be investigated and supported.	Not supported.
Geolocation awareness	To be supported in two ways: (1) clients are served by nearby senders; and (2) access to objects can be controlled based on location. This is critical for DRM.	Not supported.
Awareness to ISPs and backbone networks	To be supported. Will minimize the load on backbone networks. Will try to equalize load across ISPs. ISPs' recommendations and/or requirements will be considered in the design.	Not supported.
Data integrity and security	To be supported.	Not supported. Clients may get corrupted or manipulated files.
Interaction with other content distribution systems, e.g., multicast	To be supported.	Not supported.
Connectivity in the presence of NAT and firewalls	To be supported.	Partially supported.
Customizable quality for clients	To be supported. Each client will get the maximum quality supported by its resources.	Not supported.
Optimization for already-deployed infrastructure of industrial partner	To be considered in the design of the system.	Not supported.

Bernard Jules completed a bachelor's degree in electrical engineering at Montreal's École de Technologie Supérieure and earned two MBA degrees, one at the University of Quebec in information technology management, the other at the University of Paris Dauphine in business management. He is a member of the Institute of Electrical and Electronic Engineers (IEEE). Mr. Jules has more than 19 years of experience in computer networking, 14 of which have been devoted to managing of Internet technology and computer research and development projects. He presently occupies the position of Senior Project Manager of Internet and New Media Technology for CBC/Radio-Canada's Technology Strategy and Planning Group.

Mohamed Hefeeda is an assistant professor at the Simon Fraser University School of Computing Science, in Canada. He received a doctoral degree in computer science from Purdue University in 2004. His research interests include peer-to-peer (P2P) systems, multimedia networking, and network security. He heads the Network Systems Lab at SFU and is currently supervising six graduate students. His research is funded by Canadian funding agencies and industry through several grants. Dr. Hefeeda has substantial experience in the area of P2P computing and distributed multimedia streaming. In his doctoral thesis, he proposed a framework for cost-effective P2P content distribution systems. Two of his papers on this topic (in ACM Multimedia 03 and ICDCS 02) are among the first, and frequently cited, works on P2P streaming systems. According to scholar.google.com, each of these two papers is cited more than a hundred times. Additional information can be found at http://www.cs.sfu.ca/~mhefeeda/.