Rewards and Risks in P2P Content Delivery

Raimund K. Ege
Northern Illinois University
Dept. of Computer Science
DeKalb, IL
ege@niu.edu

Li Yang
Dept. of Computer Science
University of Tennessee
Chattanooga, TN
Li.Yang@utc.edu

Richard Whittaker
School of Comp. & Info Science
Florida International University
Miami, FL
rwhitt01@cs.fiu.edu

Abstract—*The ever increasing speed of access to the Internet has enabled sharing of data on an unprecedented scale. Data of all forms and shapes is becoming easily accessible: large multi-media files are being routinely downloaded onto a plethora of end user devices. Peer-to-peer content delivery approaches enable massive scale in the amount of data volume that can be efficiently delivered. The openness of delivery demands adaptive and robust management of intellectual property rights. In this paper we describe a framework and its implementation to address the central issues in content delivery: a scalable peer-to-peer-based content delivery model, paired with an access control model that balances trust in end users with a risk analysis to the data provider. Our framework enables data providers to extract the maximum amount of return, i.e. value, from making their original content available. Our implementation architecture provides a protocol to leverage the greatest amount of reward from the intellectual property that is released to the Internet.*

Keywords-broadband file sharing; peer-to-peer content delivery; intellectual property rights for multi media

I. INTRODUCTION

Making multimedia content available online has become a Killer-Application for the Internet. Services such as iTunes, YouTube, Joost and Hulu are popularizing delivery of audios and video content to anybody with a broadband internet connection. Additionally, virtual communities are emerging (such as FaceBook, MySpace and Twitter) where users communicate directly with one another to exchange information or execute transactions in a peer-to-peer fashion. These services are currently struggling with the challenges of securing large-scale distribution. The dynamism of peer-to-peer communities means that principals who offer services will meet requests from unrelated or unknown peers. Peers need to collaborate and obtain services within environment that is unfamiliar or even hostile. Therefore, peers have to manage the risks involved in the collaboration when prior experience and knowledge about each other are incomplete. One way to address this uncertainty is to develop and establish trust among peers. Trust can be built by either a trusted third party [2] or by community-based feedback from past experiences [3] in a self-regulating system. Trust leads naturally to a decentralized approach to security management that can scale up in size, but must be balanced with a measure of risk that is the flip side of trust.

Conventional approaches rely on well-defined access control models [4, 5] that qualify peers and determine authorization based on predefined permissions. In such a complex and collaborative world, a peer can protect and benefit itself only if it can respond to new peers and enforce access control by assigning proper privileges to new peers. The nature of digital content requires access models that go beyond checking authorization upon initial access: authorization variables quickly change in a dynamic context. The Usage Control Model (UCON) [6] is an example of a framework to handle continuity of access decisions and mutability of subject and object attributes. Authorization decisions are made before an access, and repeatedly checked during the access. On-going access may be revoked if security policies are violated. The more dynamic the situation is the more likely access will be denied, therefore denying the data provider any benefit.

The general goal of our work is to address both the trust in peers which are allowed to participate in the content delivery process, and quantifying the risk and reward garnered from releasing data in to the network. We investigate the design of a novel approach to access control. If successful, this approach will offer significant benefits to emerging peer-to-peer applications. It will also benefit collaboration over the existing Internet when the identities and intentions of parties are uncertain. We integrate trust evaluation for usage control with an analysis of risk/reward. Underlying our framework is a formal computational model of trust and access control that will provide a formal basis to interface authentication with authorization.

Our paper is organized as follows; the next section will explain our approach to peer-to-peer content delivery. Section III will elaborate on how the data source and its peers can quantify gain from participating in the content delivery. Section IV explains our risk/reward model that enables a data source to initially decide on whether to share the content and keep some leverage after its release. Section V gives an overview of our implementation framework, and Section VI details the prototype implementation of our framework that employs the fairly new Stream Control Transmission Protocol (SCTP) which improves over the current stand-bearers Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) for multi-stream session-oriented delivery of large multi-media files over fast networks. The paper concludes with our perspective on how modern content delivery approaches will usher in a new generation of Internet applications. An earlier version of this paper was presented at the Fourth International Conference on Systems (ICONS 2009), Cancun, Mexico, March 2009 [1].

II. PEER to PEER CONTENT DELIVERY

Peer-to-peer (P2P) delivery of multimedia aims to deliver multi-media content from a source to a large number of clients. For our framework, we assume that the content comes into existence at a source. A simple example of creating such multimedia might be a video clip taken with a camera and a microphone, or more likely video captured via a cell phone camera, and then transferred to the source. Likewise the client consumes the content, e.g. by displaying it on a computing device monitor, which again might be a cell phone screen watching a YouTube video. We further assume that there is just one original source, but that there are many clients that want to receive the data. The clients value their viewing experience, and our goal is to reward the source for making the video available.

In a P2P delivery approach, each client participates in the further delivery of the content. Each client makes part or all of the original content available to further clients. The clients become peers in a peer-to-peer delivery model. Such an approach is specifically geared towards being able to scale effortlessly to support millions of clients without prior notice, i.e. be able to handle a “mob-like” behavior of the clients.

The exact details of delivery may depend on the nature of the source data: for example, video data is made available at a preset quality using a variable-rate video encoder. The source data stream is divided into fixed length sequential frames: each frame is identified by its frame number. Clients request frames in sequence, receive the frame and reassemble the video stream which is then displayed using a suitable video decoder and display utility. The video stream is encoded in such a fashion that missing frames don’t prevent a resulting video to be shown, but rather a video of lesser bit-rate encoding, i.e. quality, will result [7]. We explicitly allow the video stream to be quite malleable, i.e. the quality of delivery need not be constant and there is no harm if extra frames find their way into the stream. It is actually a key element of our approach that the stream can be enriched as part of the delivery process.

In our approach, multi-media sources are advertised and made available via a central tracking service: at first, this tracker only knows the network location of the server. Clients that want to access the source do so via the tracker: they contact the tracker, which will respond with the location of the source. The tracker will also remember (or track) the clients as potential new sources of the data. Subsequent client requests to the tracker are answered with all known locations of sources: the original and the known client. Clients that receive locations of sources from the tracker issue frame requests immediately to all sources. As the sources delivery frames to the clients, the client stores them. The client then assumes a server role and also answers requests for frames that they have received already, which will enable a cascading effect, which establishes a P2P network where each client is a peer. Every client constantly monitors the rate of response it gets from the sources and adjusts its connections to the sources from which the highest throughput rate can be achieved.

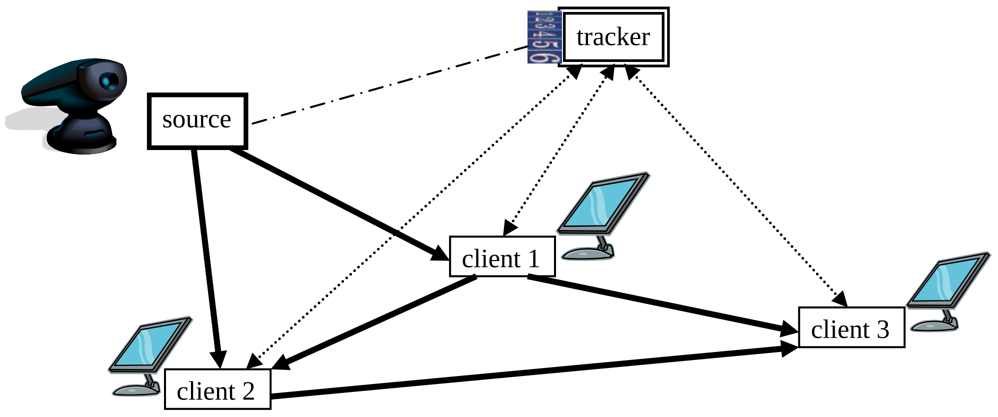

Figure 1. Content delivery network

Figure 1 shows an example snapshot of a content delivery network with one source, one tracker and three clients. The source is where the video data is produced, encoded and made available. The tracker knows the network location of the source. Tracker and source maintain a secure connection. Clients connect to the tracker first and then maintain sessions for the duration of the download: all 3 clients maintain an active connection to the tracker. The tracker informs the client which source to download from: Client 1 is fed directly from the source; client 2 joined somewhat later and is now being served from the source and client 1; client 3 joined last and is being served from client 1 and client 2. In this example, two of the clients are also serving as intermediaries on the delivery path from original source to ultimate client.

III. UNITS of RISK and REWARD

We assume that the data made available at the source has value. Releasing the data to the Internet carries potential for reaping some of the value, but also carries the risk that the data will be consumed without rewarding the original source. There is also a cost associated with releasing the data, i.e. storage and transmission cost. For example, consider a typical "viral" video found on YouTube.com: the video is uploaded onto YouTube.com for free, stored and transmitted by YouTube.com and viewed by a large audience. The only entity that is getting rewarded is YouTube.com, which will accompany the video presentation with paid advertising. The person that took the video and transferred it to YouTube.com has no reward: the only benefit that the original source of the video gets is notoriety.

In order to provide a model or framework to asses risk and reward, we need to quantize aspects of the information interchange between the original source, the transmitting medium and the final consumer of the data. In a traditional fee for service model the reward *"R"* to the source is the fee *"F"* paid by the consumer minus the cost *"D"* of delivery:

$$R = F - D$$

The cost of delivery *"D"* consist of the storage cost at the server, and the cost of feeding it into the Internet. In the case of YouTube, considerable cost is incurred for providing the necessary server network and their bandwidth to the Internet. YouTube recovers that cost by adding paid advertising on the source web page as well as adding paid advertising onto the video stream. YouTube's business model recognizes that these paid advertisings represent significant added value. As soon as we recognize that the value gained is not an insignificant amount, the focus of the formula shifts from providing value to the original data source to the reward that can be gained by the transmitter. If we quantify the advertising reward as *"A"* the formula now becomes:

$$R = F - (D - A)$$

Even in this simplest form, we recognize that *"A"* has the potential to outweigh *"D"* and therefore reduce the need for *"F"*. As YouTube recognizes, the reward lies in *"A"*, which is paid advertising that accompanies the video.

In our prior work we focused on mediation frameworks that capture the mutative nature of data delivery in the Internet [8, 9]. As data travels from a source to a client on lengthy path, each node in the path may act as mediator. A mediator transforms data

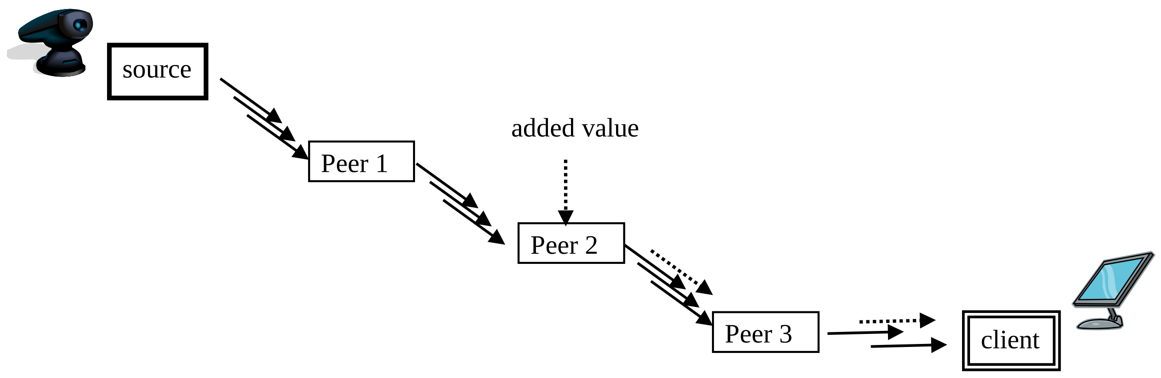

Figure 2. Varying peer behavior

from an input perspective to an output perspective. In the simplest scenario, the data that is fed into the delivery network by the source and is received by the ultimate client unchanged: i.e. each mediator just passes its input data along as output data. However, that is not the necessary scenario anymore: the great variety of client devices already necessitate that the data is transformed to enhance the client's viewing experience. We apply this mediation approach to each peer on the path from source to client. Each peer may serve as a mediator that may transform the content stream in some fashion. Our implementation employs the stream control transmission protocol (SCTP) which allows multi-media to be delivered in multiple concurrent streams. All a peer needs to do is add an additional stream for a video overlay message to the content as it passes through.

Figure 2 shows a sample path from the content source to its consumer. The multi-media source is fed as a multi stream into the content delivery path. Each peer on the path receives a number of streams and will do its best to deliver the streams to the next peer. "Peer 1" is an example of a peer that copies its input faithfully to its output. "Peer 2" shows a peer that adds an additional overlay stream to its output. Peer 3 is an example of a peer that filters out a stream to make its output more suitable for a specific target device.

The formula for reward can now be extended into the P2P content delivery domain, where a large number of peers serve as the transmission/storage medium. Assuming "*n*" number of peers that participate and potentially add value the formula is now:

$$R = F - \sum_{i=1}^{n} (D_i - A_i)$$

D_i and A_i are now the delivery cost and value incurred at each peer that participates in the P2P content delivery. The reward available to the data originator is potentially very large given the number of peers that be involved. A second dimension is opened up when we consider that the data will be consumed by many clients, so that the ultimate reward formula is the sum of all rewards gained from each client "*c*":

$$R_c = F_c - \sum_{i=1}^{n} (D_i - A_i)$$

Whether or not the data originator will gain any reward depends on whether the client pays fee "*F*" and whether the peers are willing to share their gain from the added value. In a scenario where clients and peers are authenticated and the release of the data is predicated by a contractual agreement, the source will reap the complete benefit.

In our model we quantify the certainty of whether the client and peers will remit their gain to the source with a value of trust *"T"*: *T* represents the trust in the client that consume that data, *T* represents the trust in each peer that participates in the content delivery:

$$R = \sum_{c} \left(F_c - \sum_{i=1}^{n} (D_i - A_i) \right) * T_c$$

The formula captures the ultimate truth that no reward will be materialized when there is no arrangement for trust.

IV. TRUST MODEL

It comes down to the question whether to accept a new peer into the content delivery network. For every request from a peer a measure of trust, i.e. a trust value, must be computed. The trust is evaluated based on both actual observations and recommendations from referees. Observations are based on previous interactions with the peer. Recommendations may include signed trust-assertions from other principals, or a list of referees that can be contacted for recommendations. The trust value, calculated from observations and recommendations, is a value within the [0, 1] interval evaluated for each peer that requests to be part of the content delivery.

The trust is assumed to follow a beta distribution, and is represented by the two parameters of the beta distribution. The beta distribution, a conjugate prior, is chosen because of its reproducibility property under the Bayesian framework. When a conjugate prior is multiplied with the likelihood function, it gives a posterior probability having the same functional form as the prior, thus allowing the posterior to be used as a prior in further computations. For a given requester, we define a sequence of variables T_1, T_2,…, T_k to characterize the trust at sampling time k.

For instance, at k^{th} sampling time, N_k observations were collected about the peer. Let G_k be the number of normal requests or behaviors. If there is no history of malicious behavior by the peer associated with a request (i.e., neither malware nor spyware were observed), the request is deemed as normal behavior.

Now suppose a prior probability density function (*pdf*) of trust T_{k-1}, denoted by f_{k-1}(t), is known about the peer. Then the posterior *pdf* of trust for this peer (given N_k = n and G_k = g) can be obtained from Bayes theorem [10, 11] as follows:

$$f_k(t) = \frac{f_k(g!\,t,n) f_{k-1}(t)}{\int_0^1 f(\,g!\,t,n)\, f_{k-1}(t)\, dt}$$

where $f_k(g!\,t,n)$ is called the likelihood function and has the form of a binomial distribution:

$$f_k(g!\,t,n) = \binom{n}{g} t^g (1-t)^{n-g}$$

The prior *pdf* f_{k-1}(t) summarizes what is known about the distribution of T_{k-1}. Under the assumption that prior *pdf* f_{k-1}(t) follows a beta distribution, it can be shown that the posterior *pdf* also follows a beta distribution.

In particular, if $f_{k-1}(t) \sim beta(\alpha_{k-1}, \beta_{k-1})$, we have $f_k(t) \sim beta(\alpha_{k-1} + g_k, \beta_{k-1} + n_k - g_k)$ given that $N_k = n_k$ and $G_k = g_k$. Therefore, $f_k(t)$ is characterized by the parameters α_k and β_k defined recursively as follows: $\alpha_k = \alpha_{k-1} + g_k$ and $\beta_k = \beta_{k-1} + n_k - g_k$. Initially, there is no knowledge about the peer: we assume that trust values follow a uniform distribution of the interval [0,1], i.e. f_o (t) $\sim U[0,1] = beta(1,1)$ which indicates our ignorance about the new peer's behavior at time 0. Time 0 is when the peer first becomes known to the content delivery network.

At time k, trust value T_c for a given peer c is now:

$$T_c = \frac{\alpha_k}{\alpha_k + \beta_k}$$

There are two alternative ways to update trust values. One is to update trust values based on all known observations and recommendations. The other ways is to update trust values based on recent information only. The advantage of the latter one is two folds: reduce the computation complexity and detect a change in the peer's behaviors early. For instance, if a peer has been misbehaving for a short time period, then recent observations together with actual reports are more reflective of the behavior change than would be if trust was based on all available observations.

Meanwhile, recommendations from referees bring in new information T_{rq} on the peer's behaviors. We combine the new data T_{rq} with our own observation T_{oq} on the condition that the referee is highly trusted or the recommendation passes the deviation test. The deviation test is to decide whether a recommendation is trustworthy or not. Recommendation R is learned from past interactions the referee had with the requestor. Trustworthiness of a recommendation also follows a beta distribution. f_k(t) is adjusted by recommendations: $T_{oq} := T_{oq} + \mu T_{rq}$ where T_{oq} is trust we have in the peer, T_{rq} is trust that the referee has in the peer, and µ is the trust in the referee's recommendations.

In summary, when it comes down to the question whether to accept a given peer into the content delivery network, we now have a tool to assess the potential gain balanced by the risk posed by the new peer:

$$R = \sum_c \left(F_c - \sum_{i=1}^{n} (D_i - A_i) \right) * T_c$$

Our model correlates the reward gained from accepting the new peer with the risk posed by the new peer and to enable an informed decision.

V. IMPLEMENTATION FRAMEWORK

Peer-to-peer networks are open by definition. While being open, ready to give access, our BitTorrent-style of delivery uses a tracker approach. The tracker keeps information that is global to the data exchange and can be the place to gather and disseminate control information. Each BT-style distribution of original content requires at least one tracker. Additional trackers can easily be established. The first and subsequent trackers need to carry trust with the original data source. If there is more than one tracker, we use a public key infrastructure approach[12] to authenticate and certify each tracker. The number of trackers needed is small and will pose little overhead to our model.

The tracker is the location where the decision on which peers may participate in the content delivery is made. While it may seem that the original source should be the decision maker, for the purpose of our model the original source is assumed to have delegated this authority to the tracker(s).

Our framework therefore features 3 types of participants:

1. tracker, where all information on the current status of the content delivery network is maintained and all access decisions are made.
2. client, where the consumption of the data occurs.
3. source, where the data is available for further dissemination. The original source is the first source. Clients that have downloaded and consumed the data will immediately become new sources.

What we called "peer" in our discussion so far, starts out by requesting access from the tracker, then becomes a client and ultimately a new source for that data that is being tracked and access-controlled by the tracker(s).

A. Client

The key to a smooth scaling of this ad-hoc p2p network is the algorithm used by the client to request frames from a source (either the original source or another client). A client consists of three processes:

1) a process to communicate with the tracker. The client initiates the negotiation with the tracker to enable the tracker's decision on whether the peer is admitted into the content delivery network. Upon success, the tracker informs the client which sources the client should use, and the client will update the tracker on its success in downloading the source data;
2) a process to request data from the given sources. Since the original data may be very large and exist in multiple fragments. For example, video data is typically made available as a series of frames. Fragments or frames may be requested from multiple sources. The bittorrent protocol uses algorithms to determine which sources are most likely to yield the best throughput; and
3) a process to receive frames/fragments from sources and to assemble them into usable data.

All three processes share the following data:

- a list of recommended sources to download from. This list is originally received from the tracker, but can be modified by the client based on download success;
- a list of backup servers to download from, received from the tracker, but the client can move servers from this list to the list of recommended servers based on the client's download success;
- current bandwidth utilization at the client.

The client continuously requests fragment/frame sequences until the end of the transmission is reached. It requests a fragment/frame sequences from each available source. The "receive" process runs in a continuous loop that accepts frames from servers. While frames within a sequence will arrive in the correct order the receiving process still needs to order the frame sequences number. The process also records which server delivered the frames.

The most vital process in the client is its communication with the tracker. At first, the communication focuses on qualifying the client for participation the content delivery network. During the ongoing download, the communication is meant to validate the client's continued credentials. Since trust in the peer is calculated on a continuing basis, the client needs to be in constant communication with the tracker. No lapses in the continuous communication are allowed.

B. Source

A peer that is admitted into the content delivery network will operate initially as a client. As soon as sufficient data has been downloaded, the tracker will determine whether the client can also serve as source for further downloads. The decision depends on the amount of data the client holds, and which portions of the original data are already collected in the client. Once the tracker determines that the client can serve as source, further negotiation is necessary to assess which added values the new source will contribute to the calculation of reward:

$$R = \sum_{c} \left(F_c - \sum_{i=1}^{n} (D_i - A_i) \right) * T_c$$

The tracker needs to know the values of *"D"* and *"A"* that this new source will incur. *"D"* is the cost of transmission and storage that the new source will have to pay, whereas *"A"* is the added value that the new source might be able to realize by being part of the delivery network. Based on the outcome of the negotiation with the tracker, the new source will then serve as a new mediator for the original data. The higher the trust *"T"* is that the tracker places onto the new source, the higher to overall reward will become for the original data owner. However, even a smaller amount of trust will realize an additional gain for the original source. In addition, ongoing monitoring of downloading peers must be maintained.

C. *Tracker*

The core of the content delivery model is the tracker. While the tracker might initially be a single unit, it can easily be duplicated, as long as a strong trust relationship is maintained between the trackers, and continuous exchange of peer information is maintained.

The tracker is first enabled by the original source of the data content. The tracker knows the location of the original/first source. The tracker starts by initializing its database of peers. The initial state of peer database can be augmented by historical data and/or a distributed-hash-table style if control data dissemination.

Peers that wish to participate in the content delivery must first locate the tracker. Public directories are the usual places where trackers are listed. Search engines exist for the sole purpose of publicizing content that is available for download.

A peer will start by establishing a connection to a tracker. The tracker will consider the request from a new peer and gather the necessary data on the trust in the new peer. The tracker will seek information to establish the peer's trust value:

$$T_c = \frac{\alpha_k}{\alpha_k + \beta_k}$$

If the peer is new and not yet listed in the tracker(s) database, then a new entry is created. The tracker will also determine the new peer's contribution to the reward formula. The peer will contribute a value *"D"* and *"A"*, to reflect the additional cost and added value. The tracker is the location where the determination is made whether the gain possible from admitting the new peer outweighs the risk of releasing the data content to an untrusted peer. Only if the overall reward formula shows a potential gain, then is the new peer accepted. Initially, the peer is admitted as client. As the peer accumulates downloadable volume, the tracker may elevate the status to create a new source that is allowed to provide new added content.

VI. PROTOTYPE IMPLEMENTATION

Our current prototype is implemented using the Java programming language. Since we are using the newly standardized SCTP protocol, we require the use of the OpenJDK version 7 [13] which is currently undergoing beta evaluation. To enable truly large numbers of truly large frames in our multimedia content delivery network, we keep all elements of the implementation in the 64bit space. Unfortunately 64bit implementations of SCTP are not yet standard within the Microsoft Windows family of operating system, so we are currently limited to running our prototype elements on Linux 64bit operating systems that provide direct kernel support for the new protocol via the lksctp [14] library.

SCTP [15] is a Transport Layer protocol, serving in a similar role as the popular TCP and UDP protocols. It provides some of the same service features of both, ensuring reliable, in-sequence transport of messages with congestion control.

We chose SCTP because of its ability to delivery multimedia in multiple streams. Once a client has established a SCTP association with a server, packages can be exchanged with high speed and low latency. Each association can support multiple streams, where the packages that are sent within one stream are guaranteed to arrive in sequence. Each source can divide the original video stream into set of streams meant to be displayed in an overlay fashion. Streams can be arranged in a way that the more streams are fully received by a client, the better the viewing quality will be. When sending a packet over a SCTP channel we need to provide an instance of the MessageInfo class, which specifies which stream the packet belongs to. The first stream is used to deliver a basic low quality version of the video stream. The second and consecutive streams will carry frames that are overlaid onto the primary stream for the purpose of increasing

```
01 SocketAddress socketAddress = new InetSocketAddress(port);
02 channel =  SctpMultiChannel.open().bind(socketAddress);
03 MessageInfo info;
04 while ((info = channel.receive(bb, null, null)) != null) {
05   // determine requestor
06   Association association = info.association();
07   // determine which frame range
08   bb.flip();
09   int fromFrame = bb.getInt();
10   int toFrame = bb.getInt();
11   // send frames to requestor
12   for (int i=fromFrame; i<= toFrame; i++) {
13     bb.clear();
14     bb.putInt(i);
15     bb.put(framePool.getFrame(i));
16     bb.flip();
17     channel.send(bb,
               MessageInfo.createOutgoing(association, null,0));
18   }
19 }
```

Figure 3. SctpMultiChannel maintains one-to-many association

the quality. In our framework we also use the additional streams to carry content that is “added value”, such as advertising messages or identifying logos. The ultimate client that displays the content to a user will combine all streams into one viewing experience.

The second feature of SCTP we use is its new class “SctpMultiChannel” which can establish a one-to-many association for a single server to multiple clients. The SctpMultiChannel is able to recognize which client is sending a request and enables that the response is sent to that exact same client. This is much more efficient than a traditional “server socket” which for each incoming request spawns a subprocess with its own socket to serve the client. Figure 3 shows the Java source code where an incoming request is received. Each packet that is received on the channel carries a MessageInfo object which contains information on the actual client that is the actual other end point of this association. The Java code on line 06 retrieves the “association” identity from the incoming message “info” instance. The association is then used to send the response via the same SctpMultiChannel instance but only to the actual client that had requested the frames. The code on line 17 shows that a new outgoing message info instance is created for the same “association” that carried the incoming request. The message info instance is then used to send the response packet to the client. The code to receive SctpMultiChannel packets is logically similar to any UPD or TCP style of socket receive programming. Figure 4 shows a sample.

```
SocketAddress socketAddress =
          new InetSocketAddress(peer.address, peer.port);
SctpChannel channel = SctpChannel.open(socketAddress, 1, 1);
// send requested frame range to peer
ByteBuffer byteBuffer = ByteBuffer.allocate(128);
byteBuffer.putInt(fromFrame);
byteBuffer.putInt(toFrame);
byteBuffer.flip();
channel.send(byteBuffer, MessageInfo.createOutgoing(null, 0));
// here is where we read response
byteBuffer = ByteBuffer.allocate(64000);
while ((channel.receive(byteBuffer, null, null)) != null) {
   byteBuffer.flip();
   int frame = byteBuffer.getInt();
   System.out.print("Message received: " + frame);
   …
```

Figure 4. Packets from SctpMultiChannel being received by client

The three major components of the framework are implemented as “SourceMain”, “TrackerMain” and “ClientMain”, which are composed from classes that implement the core behavior of maintaining communication sessions, accepting requests for frames and delivering them, and requesting and receiving frames. The major classes are FrameRequestor and FrameServer. The original source starts out as the sole instance of FrameServer. The first client starts out as the sole instance of FrameRequestor. As the client accumulates frames it then also instantiates a FrameServer that is able to receive requests from other clients. A client that contains both a FrameRequestor and FrameServer instance becomes a true peer in the P2P content delivery framework.

In summary, tracker, source and client together contribute to build a highly efficient delivery network.

VII. CONCLUSION

In this paper we have described a model and framework for a new generation of content delivery networks. Our framework is designed enable content originators to assess the potential reward from distributing the content to the Internet. The reward is quantified as the value added at each peer in the content delivery network and gauged relative to the actual cost incurred in data delivery but also correlated to the risk that such open delivery poses. We described an implementation architecture that follows a bittorrent-style of P2P network, where a tracker disseminates information on which sources are available to download from. This information is constantly updated and communicated to new clients. New clients join the content delivery network and become new sources for new clients to download from. Such P2P content delivery has great potential to enable large scale delivery of multimedia content.

Consider the scenario we described earlier in the paper: a typical “viral” video found on YouTube.com: the video is uploaded onto YouTube.com for free, stored and transmitted by YouTube.com and viewed by a large audience. The only entity that is getting a reward is YouTube.com, which will accompany the video presentation with paid advertising. The only benefit that the original source of the video gets is notoriety.

Using our model, the original data owner can select other venues to make the video available via a peer-to-peer approach. The selection on who will participate can be based on how much each peer contributes in terms of reward but also risk. Peers will have an interest in being part of the delivery network, much like YouTube.com has recognized its value. Peers might even add their own value to the delivery and share the proceeds with the original source.

Whereas in the YouTube.com approach the reward is only reaped by one, and the original source has shouldered all the risk, i.e. lost all reward from the content, our model will enable a more equitable mechanism for sharing the cost and reward. Our model might just enable a new and truly openness of content delivery via the Internet.

REFERENCES

[1] Raimund K. Ege, Li Yang, Richard Whittaker. Extracting Value from P2P Content Delivery. Proceedings of the Fourth International Conference on Systems (ICONS 2009), pages 102-108 Cancun, Mexico, March 2009.

[2] Y. Atif. Building trust in E-commerce. IEEE Internet Computing, 6(1):18–24, 2002.

[3] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. Reputation systems. Communications of the ACM, 43(12):45–48, 2000.

[4] E. Bertino, B. Catania, E. Ferrari, and P. Perlasca. A logical framework for reasoning about access control models. In SACMAT '01: Proceedings of the sixth ACM symposium on Access control models and technologies, pages 41–52, New York, NY, USA, 2001.

[5] S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. ACM Transaction Database System, 26(2):214–260, 2001.

[6] J. Park and R. Sandhu. The UCON usage control model. ACM Transaction Information System Security, 7(1):128–174, 2004.

[7] C. Wu, Baochun Li. R-Stream: Resilient peer-to-peer streaming with rateless codes. In Proceedings of the 13th ACM International Conference on Multimedia, pages 307-310, Singapore, 2005.

[8] R. Whittaker, G. Argote-Garcia, P. Clarke, R. Ege, Optimizing Secure Collaboration Transactions for Modern Information Systems, Proceedings of the Third International Conference on Systems (ICONS 2008), pages 62-68, Cancun, Mexico, 2008.

[9] R. K. Ege, L. Yang, Q. Kharma, and X. Ni. Three-layered mediator architecture based on dht. Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN 2004), Hong Kong, SAR, China. IEEE Computer Society, pages 317–318, 2004.

[10] L. Yang, R. Ege, Integrating Trust Management into Usage Control in P2P Multimedia Delivery, Proceedings of Twentieth International Conference on Software Engineering and Knowledge Engineering (SEKE'08), pages 411-416, Redwood City, CA, 2008.

[11] A. Papoulis. Probability, Random Variables, and Stochastic Processes. McGraw-Hill, New York, 1991.

[12] Gutmann, P., 1999. The Design of a Cryptographic Security Architecture, *Proceedings of the 8th USENIX Security Symposium*, pages 153-168, Washington, D.C., 1999.

[13] java.net – The Source for Java Technology Collaboration, The JDK 7 Project, http://jdk7.dev.java.net. [accessed September 22, 2009]

[14] The Linux Kernel Stream Control Transmission Protocol (lksctp), a SourceForge project to provide SCTP for the Linux kernel, http://lksctp.sourceforge.net/. [accessed September 22, 2009]

[15] R. Stewart (ed.), Stream Control Transmission Protocol, Request for Comments: 4960, IETF Network Working Group, September 2007, http://tools.ietf.org/html/rfc4960. [accessed September 22, 2009]