IEICE TRANS. INF. & SYST., VOL.E95–D, NO.2 FEBRUARY 2012

PAPER Special Section on Architectures, Protocols, and Applications for the Future Internet

On Demand Content Anycasting to Enhance Content Server Using P2P Network

Othman M. M. OTHMAN[†a)], ***Member and*** **Koji OKAMURA**[††b)], ***Nonmember***

SUMMARY In this paper, we suggest a new technology called Content Anycasting, and we show our design and evaluation of it. Content Anycasting shows how to utilize the capabilities of one of the candidate future Internet technologies that is the Flow-based network as in OpenFlow to giving new opportunities to the future internet that are currently not available. Content Anycasting aims to provide more flexible and dynamic redirection of contents. This would be very useful in extending the content server's capacity by enabling it to serve more clients, and in improving the response of the P2P networks by reducing the time of joining P2P networks. This method relies on three important ideas which are; the content based networking, decision making by the network in a similar manner to anycast, and the participation of user clients in providing the service. This is done through the use of the flow-based actions in flow-based network and having some modifications to the content server and client.

key words: *future Internet, content delivery networks, content based networking, anycast*

1. Introduction

At the beginning of the Internet, it served the role of delivering contents using the simple client server model. But as time passes number of users grew fast, also the users' needs became broader and more diverse, specially with the introduction of new technologies, services and with the social changes that followed the wide adoption of the Internet. As shown by studies in AKARI [8] that the traffic size increases by factor of 1.7 every year. This led to an increase in server loads, causing many difficulties for the current delivery models to cope with.

Many solutions were proposed to solve the server overloading problem like increasing the bandwidth link, having redundant servers with load balancers or using Content Delivery Network concept which was introduced to solve the server overload problem and to provide high availability of the contents. And to enable the CDN many technologies were introduced like the Anycast [11] and Peer to Peer networks as in [9]. Anycast works by using the routers to deliver packets to one host out of a group of hosts; that is the nearest host to the sender based on routing information and metrics. In case of the Peer to Peer networks, it makes use of part of user's resources voluntary provided to contribute in providing contents to other users. However, both of those two technologies; the Anycast and Peer to Peer has it's own limitations, which will be discussed in some details in Sect. 2.

All of those solutions except for Peer to Peer have to be implemented on the server side leaving it more complicated. While on the other hand, (the clients side) clients are having stable internet connections with considerably high bandwidth specially with the introduction of the Fiber To The Building FTTB and the Fiber To The House FTTH, as an example in Japan the number of FTTH users exceeds the number of DSL users as shown in [7]. The thing that created a kind of imbalance between the server side and the client side.

Many researches have been done to solve the overloading problem of the servers as in [14], which does so by providing new clients with list of other clients that are getting the same content that the new client is interested in. After that the new client have to decide which client form that list to communicate with by sending pings to all of the clients in that list and choosing the best client based on the results of the pings.

Meanwhile, in the future internet research as in FIND [5] and GENI [6] in the United States, AKARI [8] in Japan, FIF [4] in Korea, and EIFFEL [3] in Europe, where researchers from all over the world are studying challenges of the current Internet and are proposing ways to study and solve those challenges. Many efforts have been done in developing and improving the flow based networks; one of the current researches, that is mainly aiming to enable researchers to run their experiments using flow-based technology on the actual production networks which is OpenFlow [10].

OpenFlow is a part of Stanford University's clean slate project. It enables more freedom and flexibility in decision making by splitting the decision making or routing from packet forwarding. According to OpenFlow decision making can be done and modified freely by the OpenFlow controller according to layer 2, 3 and VLAN headers while the forwarding is still done by routers or switches in addition to their original functionality. This freedom and flexibility enables it to play an important role in developing the future internet.

This paper suggests a new technology called content anycasting and shows its design and mechanism. Content

Manuscript received May 16, 2011.
Manuscript revised September 20, 2011.

[†]The author is with the Department of Advanced Information Technology, Graduate School of Information Science and Electrical Engineering, Kyushu University, Fukuoka-shi, 812–8581 Japan.
[††]The author is with the Research Institute for Information Technology, Kyushu University, Fukuoka-shi, 812–8581 Japan.
a) E-mail: omo_5@ale.csce.kyushu-u.ac.jp
b) E-mail: oka@ec.kyushu-u.ac.jp

DOI: 10.1587/transinf.E95.D.514

Anycasting shows how to utilize the capabilities of one of the candidate future Internet technologies that is the Flow-based network as in OpenFlow to giving new opportunities to the future internet that are currently not available. It aims to provide a more flexible, dynamic way of redirection, to tip off the imbalance between the server side and the client side, to improve the content server side by increasing the number of clients that the server can serve, to decrease the joining time of the P2P networks, and to have a faster response. This is done through the help of some functionalities that some of the flow-based technologies provide; as in OpenFlow.

The remainder of this paper is organized as follows. We first explain about the motivation for our research in Sect. 2. We then show and illustrate our design of the Content Anycasting redirection system showing its components and its content requesting method in Sect. 3. We then show how Content Anycasting can be utilized to implement a popular large file distribution system in Sect. 4. Also, in Sect. 5 we show how Content Anycasting can be used to make an enhanced P2P network. After that, future works are shown in Sect. 6. Finally we conclude our work in Sect. 7.

2. Motivation

Current internet relies on the server/client model and the Peer to Peer model to provide service. Also various kinds of CDNs are used to overcome the server overloading problem. However many of the technologies used helped to create the imbalance between the server side and the client side. Not to mention that each of those technologies has its own limitations.

As an example, the anycast CDNs depends on having multiple fixed and pre-located replica servers. Moreover, it depends on the routing protocols to operate and this forces it to comply with the rules and delays of those routing protocols. As a result; anycast lacks the flexibility and the dynamicity of changing upon need, and requires more than one server to operate. While in case of P2P networks, they have limitations caused by their overlay nature the thing that imposes more communication in order to find the desired contents, locate other peers that already have that content, and get the content. Also, P2P networks lacks the knowledge of the network topology, which results in either poor decision making, or requiring extra phases or steps for the decision making.

And if we take a closer look at the strength points of Anycast and Peer to Peer technologies we will find that Anycast strength lies behind that its redirection decision-making is done by the network; that means that the network equipment like the routers are the ones who will decide where the packet will be sent to (to which node), while the strength point of the Peer to Peer networks lies in the contribution of users. Combining those two strength points along with the idea of content based networking shown in [12], [13] looks very promising, by having the decision making done by the network based on both regular addressing and content addressing and having users contributing in providing service. Specially, that nowadays many users have a stable internet connection with great capacities.

So we suggested a new technology, and designed it to make a more efficient use of the overall bandwidth by reducing load off the server links and distributing it to clients links, in a way that is more dynamic than the anycast through depending on flow-based networks and thus not having to comply to routing delays and back-offs, with less overhead than Peer to Peer networks by making clients able to get their desired contents without doing extra phase and thus relieving them from the burden of finding contents or peers. However content anycasting requires some changes to the network, the content server and client.

3. The Content Anycasting Redirection System

3.1 Content Anycasting System Overview

The proposed Content Anycasting system consists of content server, anycast manager, Flow-based routers or switches like an OpenFlow routers or switches and user clients (see Fig. 1).

3.1.1 Content Server

The content server is almost a regular content server that provides contents like large files or videos. Usually servers keep track of current user clients that are getting contents; also the content server must be modified to keep track of its current user clients upload capabilities. Where the upload capabilities are extracted form the first packet (START) of the Probe protocol (see Sect. 3.2.2). The use of the upload capabilities shows in finding out the number of new user clients that the current user client can serve - that is the *number of redirections*. This means that the content server will have a list of tuples (current client IP, upload capabilities), where each of those tuples will be used only once in creating a *redirection request* (see Fig. 2.) or part of it and then it will be discarded or moved to another list. By doing so the content serve assures that each client will serve the a suitable number of other clients based on the upload capabilities that the client have provided, since that no other part of the redirection system will manipulate the *number of redirections* after it has been decided by the content server using the upload capabilities that the client have declared in the probe protocol. And in order for a client to be involved in serving other clients after it did so in a previous time, it has to send a

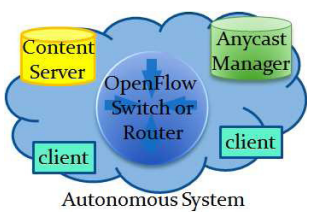


Fig. 1 System overview.

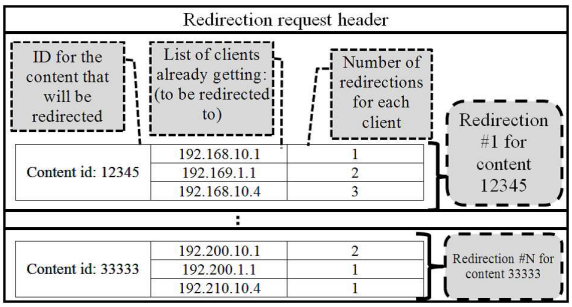


Fig. 2 *redirection request* example.

new START packet either for a new content or for the same content, which will be used again by the content server to create another *redirection request*, and so on.

Also the content server is responsible for signing all of it's contents. The signing process is done by dividing the file to logical chunks, signing one chink periodically (e.g. signing the 100th chunk for every 100 chunk), and appending the signature of that chunk to the end of it. In more details of the process of signing one chunk; it is similar to any digital signature process. It is done by running a hashing algorithm on the desired chunk to get a relatively small value which will be encrypted by the private key of an asymmetric encryption to generate the signature of the chunk. However the exact details of this process of signing contents is left for the content server operator to choose, for example type of asymmetric encryption and the period for signing a chunk and so on.

Moreover, each content in the content server must have a unique identifier, and the content server must manage the set of identifiers for the contents it serves.

The content server must be modified so that it sends a *redirection request* to the anycast manager in order to initiate the redirection. This should be done when the server reaches some threshold or condition. However, specifying this threshold or condition is left to the operators of the content server, so that each content server can choose the threshold that suits it, for example number of user clients or available bandwidth for the server, etc.. This *redirection request* includes the contents' ids and the list of user clients that are currently getting this content and the *number of redirections* for each one of them (See Fig. 2.).

3.1.2 Anycast Manager

The anycast manager is responsible for creating redirections and managing those redirections. The redirection system might require more than one anycast manager for example one for each autonomous system, which means one anycast manager for each network of independent organization that implements BGP (Border Gateway Protocol) (see Fig. 3).

The first function that the anycast manager provides is managing *redirection request*s which mean deciding which is the best anycast manager to handle the *redirection request* and forwarding the *redirection request* to it.

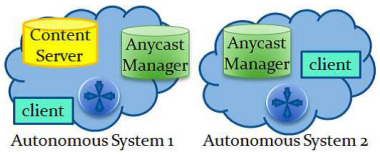


Fig. 3 An example of a redirection system with two autonomous systems and two anycast managers.

Header Fields													Counters	Actions
Ingress Port	Ether source	Ether dst	Ether type	VLAN id	VLAN priority	IP src	IP dst	IP protocol	IP ToS	TCP/UDP src port	TCP/UDP dst port			
X	X	X	X	X	X	X	Content server IP	Modified TCP #	X	Content ID				Modify IP dst to be the current user client IP address. ----- Send packet header to controller. ----- Delete this entry if idle for more than X seconds.

Fig. 4 An example of the flow table entry generated by the redirection controller.

The second function it provides is creating the redirection by sending instructions to all of the Flow-based routers or switches present within the same autonomous system as the anycast manager to redirect the *content request*s that are originally sent to the content server to other user clients that are within the same autonomous system as the requesting client. In more details for the second function, the anycast manager works as an OpenFlow controller, since OpenFlow is chosen as the flow based technology used in this study. Through acting as an OpenFlow controller the anycast manager will be able to add the redirections (shown in Fig. 4) to all of the OpenFlow routers or switches present within the same autonomous system as the anycast manager, enabling them to redirect packets according to those redirections. And about redirections themselves, they are OpenFlow's flow-entries that are carried out by the OpenFlow routers or switches to which they were sent and stored in the flow tables. It should be noted that redirections have an idle time out value, which means that the router or switch that stores them will terminate those redirections after being idle for a specified time. The idle time out value is important to prevent old and unused redirections form residing on the router or switch forever. Deciding the value for the time out is the responsibility of the anycast manager. However, studying the bases for choosing the idle time out value will be carried out as part of the future works - see Sect. 6.

The third function the anycast manager provides, is maintaining the redirections and terminating them. This can be explained as follows. After the second function as explained in the previous paragraph is carried out, the anycast manager creates a counter and stores it along with the redirection and the *number of redirections* which was carried in the *redirection request*. By doing so, the anycast manager will have a list of tuples (redirection, counter, *number*

of redirections). This list stores all of the currently used redirections in the network along with their counters and termination conditions (*number of redirections*). And according to the redirection as shown in 4; whenever a redirection is used by a Flow-based router or switch, the using router or switch sends the header of the incoming matching packet (*content request*) to the anycast manager. Then the anycast manager increments the counter for the appropriate tuple that matches the received header, and when the counter reaches up to the corresponding *number of redirections* value the anycast manager sends instructions to all of the Flow-based routers or switches to terminate that redirection. However, this method for terminating redirections might not be the optimal one, but it is the best available using the current implementation of OpenFlow. This problem is briefly discussed in Sect. 6 of the future works.

By implementing the previously described functionalities, there would be no need for the anycast manager to keep track of and synchronise the flow table in each router or switch, instead the router or switch is the one who reports in case of a match to the anycast manager, who will in turn terminate the redirection upon the need.

3.1.3 Flow-Based Routers or Switches

In case of our study we used OpenFlow as the chosen flow-based technology due to its high flexibility and its wide range of supported actions. The role of the OpenFlow routers or switches is to perform the redirections that where created by the anycast manager. As mentioned before that the redirections are OpenFlow's flow table entries, where each incoming packet is checked if its destination IP address and its content id matches the content server IP address (as a destination IP) and the content id in one of the flow table entries. And if a match occurs the Flow-based router or switch uses one of the Openflow functionalities to change the destination IP address in the first packet of the *content request* from the IP address of the content server to the IP address of a client within the same autonomous (current client) system, where the special phases for requesting content will be shown in Sect. 3.2. By doing so, the first packet of the *content request* will be delivered to a client that lies within the same autonomous system without any effort done by the user clients.

3.1.4 User Clients

User clients in our system have a modified behavior that they will act as servers for the contents they are currently getting from the content server. Also the method of getting the content is different than the normal method as described below - in Sect. 3.2.

3.2 Requesting Content

In order to make use of our system a special method for getting the content is required. This new method is divided into three phases; the first one is related to finding the content id, while the second is related to sending the *content request*, and the third one aims to establish the connection and downloading the content.

3.2.1 Phase 1: Getting Content ID

Before this phase begins the user client browses the content server's web pages looking for the contents that he is interested in. When the client finds the desired content that has the content id as part of its URL, the client side software (web browser for example) will store the content id in order to use it in the second phase. Using the URL to hold the id for the content was chosen in this study due to the ease of use and integrating with other technologies like the World Wide Web. Moreover contents like files or streams usually have a special URL which make it easy to add their id to their URL, also URLs will be preloaded; this is because the URLs are usually part of web pages and thus there will be no need for a special step in the Probe protocol to get the content id, instead it will be extracted from the URLs that were loaded in the previous page.

3.2.2 Phase 2: Content Request

This phase takes place after the client gets the content id and before starting to download the content. In this phase a 3 way handshake is used. And in order to enable this, a new "Probe protocol" is introduced (see Fig. 5).

The probe protocol is similar to the UDP, but have a new protocol number which make it a new protocol that is different form the regular UDP protocol. According to the Probe protocol the source port and destination ports are still used to distinguish the applications, however their position in the Probe protocol's header is different than that of the regular UDP protocol. In addition to the source and destination port the content id field of 32 bit is used for the redirection, where the content id is used along with the IP address of the content server to identify contents. To explain more; the scope where the content ids have to be unique is within each content server. This means that content id have to be unique only within the same content server and that contents with the same content ids can exist within different content server. This content id is used in the process of the redirection that is done by the OpenFlow routers or switches.

Also an IP address field is added so that the packet

Bit offset	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31		
0	Content id		
32	Length		Checksum
64	Source Port		Destination Port
96	Command	Upload Capability Fraction	Upload Capability Multiplier
128	IP address		
160	Xid		
192	Number of File Chunks		Period of Signed Chunk

Fig. 5 The Probe header (modified UDP), where the highlighting shows the modified fields.

sender uses this field to tell the other part of the communication about its IP address in case of the ACK packet, while in case of the START packet the sender will place the content server IP address in it so that the receiver can use it to distinguish if the packet have been redirected by the network or not by comparing it against the destination IP address of the IP header.

In addition to those fields there is also, the "Upload Capability Fraction" and the "Upload Capability Multiplier", which will be filled by the sending client when sending START packet. Those two fields are useful for the content server to extract the upload capabilities of the sending client by multiplying them, while if the receiver is a client (as a result for a redirection) then it will ignore those values. And finally it has Xid or the transaction id which will store a value like a time stamp that will be used to assure the consistency of the 3 way handshaking of the probe protocol.

Also the probe protocol header has two other fields; the "Number of File Chunks" and "Period of Signed Chunk", that will be used in the START/ACK packet of the probe packet handshake, thus they will be filled up by the content server or the current client (the current client will fill it up with values it got form the content server after connecting to it in the first place). And the new client is the one who will use them as explained in Sect. 3.2.3. As their names indicate, the Number of File Chunks holds the number of logical chunks the content file is divided. While, the Period of Signed Chunk holds the number of chunks that the content server counts to reach a chunk that should be signed.

According to the Probe protocol, the client initiates this probe handshake by sending a START packet that contains the content id in it along with the server IP address in the probes IP field. This packet will be redirected by the OpenFlow switches or routers to another client (the current client) (as explained in Sect. 3.1.3). And in response to this initiation the other client (current client) that received the packet will respond with START/ACK packet that contains its own IP address in the probe header's IP field to be used later by the receiving part (the new client). Finally the new client will respond by acknowledging the received acknowledgement (ACK/ACK) so that the other client (current client) knows that his packet was received.

3.2.3 Phase 3: Getting the Content

At this time the new client realizes that it has been redirected to another client and initiates a TCP session using the IP address of the other client (current client) it received in the second phase directly without relying on the redirection system. Figure 6 (a) shows the Probe's 3 way handshake and the TCP session in case of a match (see Fig. 6 (a)).

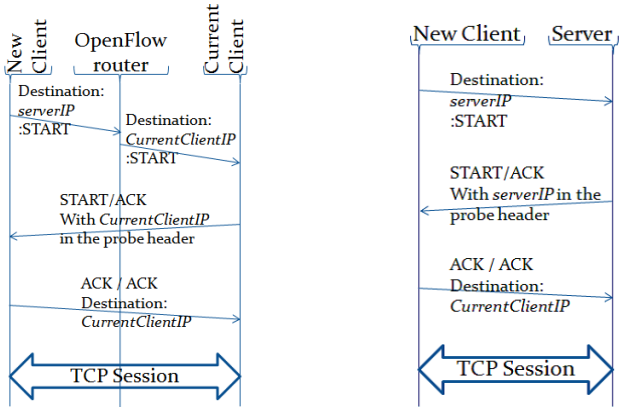
However, the method of requesting content adopts the same best effort security measure as in the current client server model. And thus, the new client is not protected from receiving packets form an eavesdropper, same as the case of current client server model. And for that reason, a new mechanism is designed to enable the user client to detect if the content that it receives is the original content (i.e. the one form the content server), regardless of the place it currently getting the content from (e.g. from another client or from the content server). This mechanism makes use of the signed content (as explained in Sect. 3.1.1). According to this mechanism, the TCP session has two states; *untrusted* and *trusted*. When the TCP session starts it will be in the *untrusted* state, in this state the application have to keep checking if it received a signed chunk or not. If a signed chunk is received, the receiver have to check if the signature was made by the content server it self (by calculating the hash value for that chunk and comparing it with the value obtained by decrypting the signature using the public key of the content server). After that, if the signed chuck was found authentic (singed by the content server) the application will change the state of the TCP session to *trusted* and continue using the session. On the other hand, if the application does not find a signed chunk after waiting for three *periods of signed chunks* it will terminate the TCP session and discard all of the chunks it have received through this session, because there might be a chance that the other part of this TCP session (client or server) is sending a tampered version of the content.

Using the Probe protocol the role of the redirection system is minimized to modifying one packet for each *content request*'s probe that has a matching redirection, otherwise the system will treat the *content request* in a traditional manner and send it directly to the content server (see Fig. 6 (b)).

3.3 Steps for Using Content Anycasting

First, Fig. 7 (a) show the content server, anycast manager, client A which is currently downloading the desired content, client B which is a new client that is interested in downloading the same content and an OpenFlow router shown as the circle in the figure.

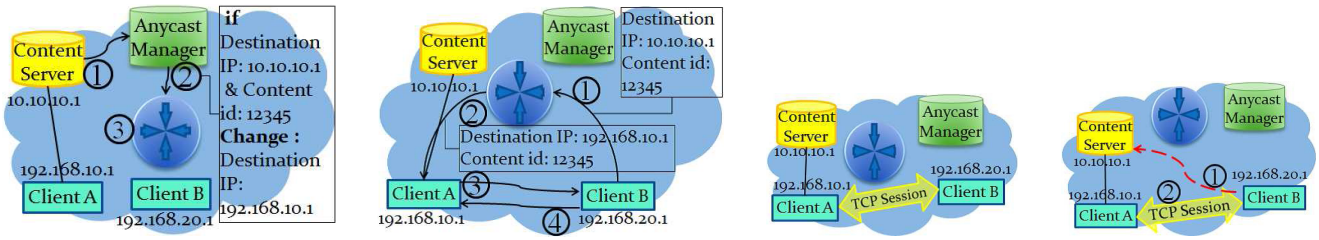
In step 1 the content server sends a *redirection request* to the anycast manager. In step 2 the anycast manager analyzes the request and prepares the required redirections and



(a) Content request in case of redirection.

(b) Content request in case of no redirection.

Fig. 6 Content request handshake.



(a) System initiation. (b) Content requesting. (c) Getting the content. (d) Getting the content in case of enhanced P2P.

Fig. 7 Steps for using content anycasting.

sends them to the OpenFlow router. And in step 3 the OpenFlow router saves the redirection, which is an OpenFlow table entry to its flow table.

While for the case of multiple autonomous systems (as in Fig. 3); the content server sends the redirection request to the anycast manager present within its same autonomous system. In case of Fig. 3 the content server will send one *redirection request* that has all of the current clients to the anycast manager in autonomous system 1 (AS1). Then the anycast manager in AS1 looks up the autonomous system for each of the clients IP's contained in the *redirection request* (presuming that anycast manager has the ability to look up in which autonomous system each IP address is present). After that, the anycast manager of AS1 groups the clients according to their autonomous system, and places each group within a new *redirection request*. Then it sends each of the new *redirection requests* to the anycast manager that is responsible of the autonomous system of that group. As an example, in Fig. 3 the anycast manager of AS1 receives the *redirection request* and groups the IPs that are located in autonomous system 2, and sends a new *redirection request* to the anycast manager of AS2 that has only IPs of clients in AS2. After that, each of the anycast managers in AS1 and AS2 will follow step 2 and 3 as explained for Fig. 7 (a) in the previous paragraph, by creating redirections (that has IPs of clients within the same autonomous system) and sending those redirections to all of the OpenFlow routers or switches within that autonomous system. Finally, each anycast manager carries out all of the following steps of this section locally.

Next, Fig. 7 (b) shows phase 2 of the *content request*. In Fig. 7 (b) step 1 client B sends the Probe protocol's START handshake that contains the content id. In step 2 the OpenFlow router redirects the packet sent by client B to go to client A instead of the content server, using the manner explained in Sect. 3.1.3. But in case of not finding a matching redirection the router will process the packet in the conventional way by sending the packet to the content server. Then in step 3 client A will acknowledge the START of step 1. And finally, client B will confirm that it received the acknowledgement.

Finally, Fig. 7 (c) shows the third phase of requesting the content; where client B and client A completes the handshake of the Probe protocol. Then client A will start sending the content to client B directly without using the redirection on the router using a direct TCP session.

4. Popular Large File Distribution

In this section we will show how content anycasting can be used to increase the number of clients getting the popular large file. This is done through having the content server sending *redirection request* that has the IP address of clients that are downloading the popular file from the content server. The scenario that shows how this works, is the same as the steps shown in Sect. 3.3.

4.1 Evaluation of Using Content Anycasting in Popular Large File Distribution System

In order to judge the effectiveness and advantages of using Content Anycasting in building popular large file distribution system, we designed and implemented a simple simulator using Java programming language as a preliminary method of evaluation. It is designed to compare the content server load in case of using Anycasting as a method of constructing the Large file distribution system with the case of using Content Anycasting for that purpose. In more detail this simulator runs using a topology of 5 areas resembling 5 autonomous systems.

Then running simulation using anycasting with one replica content server located in each one of the areas. This resulted in 20% of the load being distributed on the 5 replica servers (see Fig. 8).

And then we used the same topology to build a file distribution system using content anycasting with only one content server. Different cases were studied, where each client is capable of serving 1 or 2 or 3 or 4 other clients (see Fig. 8). This resulted in having the content server load being 50%, 33%, 25% and 20% respectively for the previous mentioned cases (each client is capable of serving 1, 2, 3, 4 other clients).

This shows that using content anycasting is capable of achieving the same load on one server rather than 5 in case of anycasting if all of the clients were capable of serving 4 other clients.

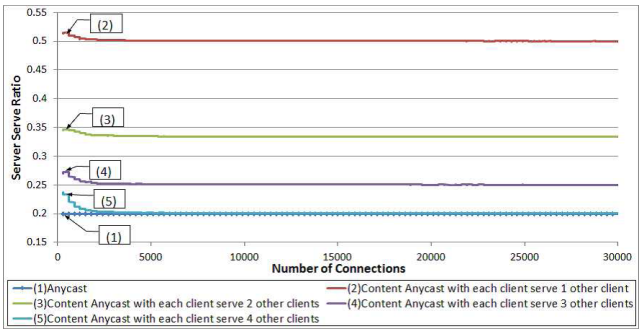


Fig. 8 The content server load in case of using current anycast, and content anycasting with 4 cases; where each client is capable of serving 1, 2, 3, 4 other clients.

5. Enhanced P2P Netwrok

In this section we will show how content anycasting can be used to enhance the P2P network by decreasing the joining delay of new clients and by making the management entity of the P2P network more resilient to the flash crowds (large number of clients' requests in a short period of time) and thus increasing its availability. To elaborate more about the availability of of typical P2P management entity; in case of BitTorrent, [15] shows that 45% of the trackers appears to have an average uptime smaller than 1.5 days (tracker is the central management entity/server for the BitTorrent network).

This enhancement of the P2P network is done on the assumptions that clients have a stable internet connection with a high bandwidth (for example as a requirement by the IPTV provider) and also that they are less likely to leave the network (as in case of video stream of popular sport events).

This is done by having the clients reporting that they have joined network to the server, so that the server will know about all of the clients that are currently joining the network even the clients that have joined after being redirected and getting content by another client. And thus the server can use those clients in order to make them server others after checking the desired properties of the P2P network like the depth of the P2P topology tree and sending the appropriate ones in a *redirection request* to the anycast manager. This will be illustrated in an example in the following section (Sect. 5.1).

This has the advantage of bringing the construction of the P2P network one step ahead. For example when a new client wants to join the network it does not have to contact the server to find which client to get the connection from, instead the client request will be directly connected to the client that he will be connecting to, and thus joining the P2P network. And after the client joins the P2P network; he will inform the server about joining the P2P network. This means that the new client will first join the network and then contacts the server. While in case of other P2P networks, the new client have to contact the server before joining the network.

5.1 Example of using Content Anycasting to Make an Enhanced P2P Network

To elaborate how the content anycasting can be used to construct and enhanced P2P network the same steps and explanations used in Sect. 3.3 for Fig. 7 (a) and Fig. 7 (b) are identical in case of the enhanced P2P network except for the last figure, where Fig. 7 (d) shows the third phase of requesting the content; where client B and client A completes the handshake of the Probe protocol. Then client B will send to report to the server that it will join the P2P network through client A as shown in step 1. And then client B will start receiving the content from client A directly without using the redirection on the router using a direct TCP session in step 2.

5.2 Evaluation of using Content Anycasting to Make an Enhanced Network

In order to judge the effectiveness and advantages of using Content Anycasting in making an enhanced P2P network, we designed and implemented a simple simulator as a preliminary method of evaluation. This simulator was built using Java programming language. It is designed to compare the content server load, and the overhead of joining the P2P network in case of using a typical method of constructing the P2P network (using a central management entity for the P2P network) with the case of using Content Anycasting for that purpose. In more detail this simulator runs using a topology of 5 areas resembling 5 autonomous systems.

Which we will use to compare the overhead of joining the P2P network for the regular P2P network construction method with that of the enhanced P2P network using content anycasting. And also we will use it to compare the server load in case of flash crowd (large number of clients' requests content in a short period of time).

In order to assess the overhead of joining the P2P network we counted the average total number of hops that the clients' request and their reply packets had to travel across the network before the client is able to get the content. Figure 9 shows the results that we obtained by comparing the case of regular methods of constructing a P2P network (using a central management entity for the P2P network) with the case of using content anycasting (again with four cases where each client serves 1, 2, 3, and 4 other clients). This figure shows that the average hop count needed before getting content according to the topology we used was about 15 hops in case of the regular method of constructing the P2P network, while the average was around 5 hops in case of using Content Anycasting. Which means; that using content anycasting to construct a P2P network would improve the overhead of joining the P2P network.

And in order to assess the server load in case of flash crowd, we set our simulator to periodically burst a large number of connections and then used it to count the number of connections that reached the server (or the P2P man-

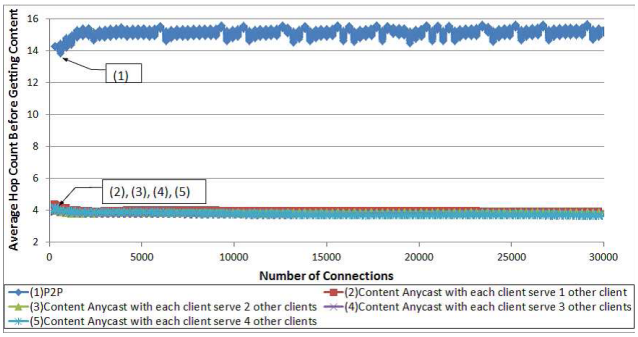


Fig. 9 The average distance in hops that the client request have to travel prior to get the content in case of using central management P2P, and content anycasting with 4 cases; where each client is capable of serving 1, 2, 3, 4 other clients.

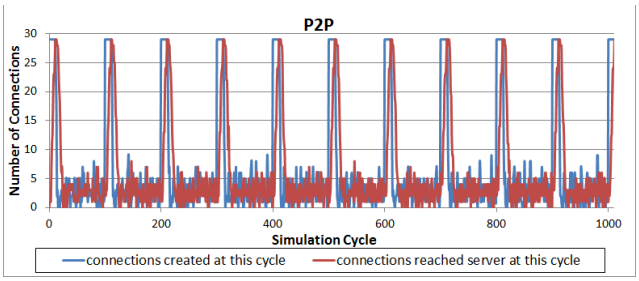


Fig. 10 Connections created by clients and connections served by the server in each cycle of the simulation in case of regular P2P with central management.

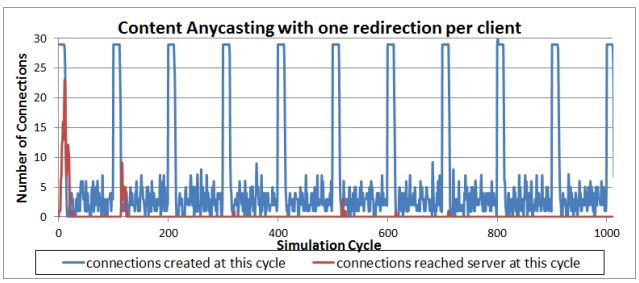


Fig. 11 Connections created by clients and connections served by the server in each cycle of the simulation in case of content anycasting.

agement entity) at each simulation cycle. Next we ran this simulation two times; first, using the regular method of constructing P2P network (see Fig. 10), second using content anycasting to construct the P2P network (see Fig. 11).

Those figures show that in case of using the regular methods to construct P2P network; all of the requests sent by the clients have to reach the server (or the central management entity) to be processed and replied back to the clients. This is due to the way they operate; in which the client have to consult the server or the management entity so that they reply with the information needed to contact other peers. While in case of using content anycasting the server does not have to process all of the requests because the new clients requests will be redirected by the network towards the other peers, and so there is no need to consult the server (or the management entity) prior to getting the content.

Also, a mathematical estimation for the load of implementing the content anycasting on the network is shown as follows. Assuming that **D** is the number of OpenFlow routers or switches in a network, **C** is the number of clients that are currently getting contents (current clients), and **R** is the average number of redirections for each client (average number of new clients that each client can serve). This estimation will be calculated in units of basic OpenFlow operations. First the anycast manager will send a total number **= 2 * D * C** of redirections (OpenFlow's flow modifications). And thus, each router or switch will receive **= 2 * C** of redirections (OpenFlow's flow modifications). The multiplier of two in the previous estimations is because; one flow modification will be sent to install a redirection and another will be sent to terminate a redirection. Also the value **= (R * C)/D** represents the number of times on average each OpenFlow router or switch will carry out OpenFlow's actions. However those estimation are still insufficient to cover up all the details, and so, more work have to be done in the future works - Sect. 6.

6. Future Works

In this study we faced many aspects that requires further investigation. First, the determination of the time out value (see Sect. 3.1.2) must be studied in more detail, such as the relationship between the time out value and the number of redirections stored in the router or switch, have to be studied deeply. After that, an algorithm to be implemented in the anycast manager for deciding the optimal time out value based on the number of redirections in the routers or switches in the network and their capabilities, and other related parameters.

Second, the method for terminating redirections (see Sect. 3.1.2) is not an optimal one, however this method is built to suit specifications of OpenFlow. And so, any Flow-based network technology including OpenFlow must support an alternative method in addition to relying on a centralized controller to control the routers or switches. Thus, a new mechanism or an enhancement to the method of interaction between the components of Flow-based technologies as in OpenFlow have to be proposed, to facilitate the use of Flow-based technology for future internet applications.

7. Conclusion

Providing new opportunities to the future internet is very important. This requires the creation and adoption of new technologies. Content Anycasting shows how to utilize the capabilities of one of the candidate future Internet technologies that is the Flow-based network as in OpenFlow to giving new opportunities to the future internet that are currently not available. In this paper we suggested a new technology, and described our new design for content anycasting, which aims to improve the availability of the content server, to increase the number of clients served, and to reduce the delays

imposed by P2P networks. Through this paper we showed our design of content anycasting, which relies on both the destination address and the content id to take the redirection decision. We also showed two examples that make use of content anycasting that are; providing an enhanced popular large file distribution, and an enhanced P2P network. Our preliminary simulation shows that using content anycasting with only a single content server; load can be reduced to 20% (in case that each client is capable of serving 4 other clients) which is the same load using 5 replica content servers in case of the regular anycast. Which means that according to our simulation content anycasting is capable of achieving the same load on content server as the regular anycasting while reducing the number of servers by 80%.

Moreover content anycasting showed that it is capable of reducing the average hop count needed prior getting the content by 74%(in the case studied in our simulation), because it makes use of the network to redirect packets and so helps to redirect requests to a peer within the same network rather than randomly choosing peers regardless of their location. Also, content anycasting showed it can be used to take the process of querying about peers one step ahead by pre installing redirections on the network rather than having the server to reply to all of the queries. And thus it improves the availability of the server in many cases and especially in case of flash crowds.

References

[1] BitTorrent Protocol Specification v1.0, http://wiki.theory.org/BitTorrentSpecification
[2] V. Cerf, Y. Dalal, and C. Sunshine, "SPECIFICATION OF INTERNET TRANSMISSION CONTROL PROGRAM," Dec. 1974.
[3] EIFFEL, http://www.future-internet.eu
[4] FIF, http://fif.kr
[5] FIND, http://www.nets-find.net
[6] GENI, http://www.geni.net
[7] M. Fujino, "National broadband policies: 1999–2009, Japan," Oct. 2009. http://www.soumu.go.jp/main_sosiki/joho_tsusin/eng/presentation/pdf/091019_1.pdf
[8] M. Hirabaru, M. Inoue, H. Harai, T. Morioka, H. Otsuki, N. Nakauchi, S. Xu, V.P. Kafle, T. Miyazawa, J. Phuritatkul, T. Umezawa, M. Ohnishi, I. Yairi, H. Yagi, K. Fujikawa, R. Li, M. Ohta, F. Teraoka, M. Murata, H. Morikawa, A. Nakao, H. Imaizumi, F. Kubota, and T. Aoyama, Oct. 2008. "New generation network architecture AKARI conceptual design (ver1.1)," AKARI Arechitecture Design Project. http://akari-project.nict.go.jp/eng/concept-design/AKARI_fulltext_e_translated_version_1_1.pdf
[9] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, 2005. "Should internet service providers fear peer-assisted content distribution?," Proc. 5th ACM SIGCOMM Conference on internet Measurement (Berkeley, CA, Oct. 2005). Internet Measurement Conference. USENIX Association, Berkeley, CA, 6-6.
[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," SIGCOMM Comput. Commun. Rev. 38, 2, pp.69–74, March 2008.
[11] C. Partridge, T. Mendez, and W. Milliken, "Host anycasting service," RFC 1546, Nov. 1993.
[12] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K.H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," SIGCOMM Comput. Commun. Rev. 37, 4 (Aug. 2007), pp.181–192, 2007.
[13] V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, and R.L. Braynard, "Networking named content," Proc. 5th International Conference on Emerging Networking Experiments and Technologies (CoNEXT '09). ACM, New York, NY, USA, 1-12, 2009.
[14] T. Tsuji, M. Kaneko, Y. Arakawa, S. Okamoto, and N. Yamanaka, "Study and implementation on P2P based contents delivery network system using cooperation of server and clients," IEICE Technical Report, PN2007-90, March 2008.
[15] G. Neglia, G. Reina, H. Zhang, D. Towsley, A. Venkataramani, and J. Danaher, "Availability in BitTorrent Systems," 26th IEEE International Conference on Computer Communications, INFOCOM 2007, pp.2216–2224, May 2007.

Othman M. M. Othman recieved M.S. Degree in Graduate School of Information Science and Electrical Engineering from Kyushu University, Japan. And received B.S. in Faculty of Engineering form An-Najah National University, Palestine. He is a First year Ph.D. student and belongs to the department of Advanced Information Technology, Graduate school of Information Science and Electrical Engineering, Kyushu University, Japan.

Koji Okamura is a Professor at Department of Advanced Information Technology and also at Computer Center, Kyushu University, Japan. He received B.S. and M.S. Degree in Computer Science and Communication Engineering and Ph.D. in Graduate School of Information Science and Electrical Engineering from Kyushu University, Japan in 1988, 1990, and 1998, respectively. He has been a researcher of MITSUBISHI Electronic Corporation, Japan for several years and has been a Research Associate at the Graduate School of Information Science, Nara Institute of Science and Technology, Japan and Computer Center, Kobe University, Japan. He is interested in Internet and Next Generation Internet, Multimedia Communication and Processing, Multicast/IPv6/QoS, Human Communications over Internet and Active Networks. He is a member of WIDE, ITRC, GENKAI, HIJK projects and Key person of Core University Program on Next Generation Internet between Japan and Korea sponsored by JSPS/KOSEF.