Scalable Control Panel for Media Streaming in NDN

Kai Lei, LongYu Yu
School of Electronics and Computer Engineering (SECE),
Peking University, Shenzhen 518055, P.R. China
leik@pkusz.edu.cn
1201213683@sz.pku.edu.cn

Jun Wei
Samsung Research America, Silicon Valley
jun01.wei@samsung.com

ABSTRACT
An NDN-based scalable control panel for a media streaming system was designed and implemented in this paper. The system is developed based on a previous IP-based P2P media streaming system named Hippo [1], which contains a group of control servers to manipulate P2P functionalities, such as the tracker, etc. System scalability becomes one of the most difficult problems when the user size of P2P system grows very large. We took the advantages from the same principle of SNC [2] to design the NDN-Hippo's control layer. As for implementation, we took a two-step approach: First porting the control layer of Hippo to NDN-based system, then porting media traffic layer later. By separating control and media layers, our demo demonstrates that not only some management functions of tracker can be smartly and instinctively achieved in NDN, but also the scalability of NDN version of Hippo has been greatly improved.

Categories and Subject Descriptors
C.2.1 [Network Architecture and Design]: Network communications.

Keywords
P2P; NDN; SNC; Media Streaming

1. INTRODUCTION
Hippo [1], an IP based P2P VOD streaming system, was developed by The Center for Internet Research Engineering (CIRE) at Peking University. Like many other P2P systems, Hippo's control relies heavily on a tracker system. It is difficult to scale the tracker system when the user size of Hippo grows large. We ported Hippo to NDN with a scalable control layer described in SNC [2]. As for implementation, we took a two-step approach: First porting the control layer of Hippo to NDN-based system, then porting media traffic layer later.

In this demo, it presents the architecture of the new NDN control layer for NDN-Hippo, which is based on SNC. It shows that NDN-based control system is easier to manage than that of its IP-based counterpart. The server failover of NDN-Hippo requires no additional specific design compared to IP-Hippo. The media traffic part of the system has not finished yet, but it would not impact our demonstration since these two layers work independently.

ICN'14, September 24–26, 2014, Paris, France.

http://dx.doi.org/10.1145/2660129.2660138

2. SYSTEM ARCHITECTURE

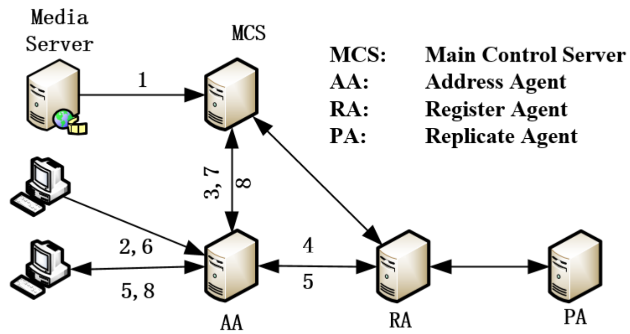

Figure 1: The architecture of the system

The Hippo media system is separated into two layers, one is the control layer and the other is the media layer. The control layer is responsible for system operation like user and server management, while the media layer is mainly responsible for streaming data sharing, transportation and distribution. This kind of architecture has two benefits. First, it separates the mass of media data and the small quantity of light-weighted control data. Second, it separates the control and the data function.

The control layer notifies the median layer with available data source information, having replaced the tracker of IP-based Hippo. Similar to many other P2P media systems, Hippo relies on a centralized server, which brings problems like single point of failure or traffic over concentration. It often gets into large scalability problems.

We reference SNC and modify the design of NDN-Hippo to suit for Hippo's need. Similar to SNC, when a new user joins, NDN provides the nearest server from the server pool. This select is implemented by a suppress mechanism, which can inform the other nodes to stop when the nearest node receives the interest.

The Hippo's control system architecture is shown in [1], including the following four components:

1. The Main Control Server (MCS) is the main node that initializes the server pool. Other server nodes ask it for a unique ID in the system. It also takes the

responsibility for controlling the distribution of media address information.

2. The RA (Register Agent) is the node that clients register, and it allocates the group ID for clients, generating a code for client authentication.
3. The PA (Replicate Agent) is in charge of backup the state of other nodes, preparing for recovery whenever other nodes fail-over.
4. The AA (Address Agent) provides service with the name follow SNC. It accepts the request of client, and provides verification and returns the relevant data.

In addition, there is a Hippo control server named Media Server responses for activate the MCS.

These four kinds of nodes except the Media Server are not running at beginning. They all are installed in every node of the server pool, and only run after someone calls them. The agent running firstly is the MCS. It must be called by Media Server, and will serve the other agent.

From that time, the other agents can join the cluster. MCS will request replicate service, so the nearest node will run the PA. And users will request the data it need, which will firstly select a node to be AA. After that AA will select a RA to register. These 3 nodes need to ask MCS for an ID before they run their service. The system works as follows:

Register: 1) The Media Server assigns a nearest server as the MCS, which signifies the server pool is created. 2) One client registers in the server cluster by sending an interest to it. 3) The node who receives the interest joins the server pool by sending an interest to MCS to apply an ID. 4) Then it requests the RA to process the registering. 5) After request, RA produces an access code and returns back to the client, the client can use it to request media service.

Request Data: 6) Then the client can send an interest to request the data with the access code. 7) The AA which receives the interest authenticates the access code, and requests data from MCS. The AA can be ever server in this cluster. 8) Then returns the data back to client.

3. DEMONSTRATION

In this demo, we demonstrate 1) the NDN-based Hippo control system allows users to select the nearest control server; 2) the failover of Hippo server is automatically supported; 3) the Hippo registration work flow in NDN. We build a simple test system as shown in Figure 2. The implementation uses CCNx.

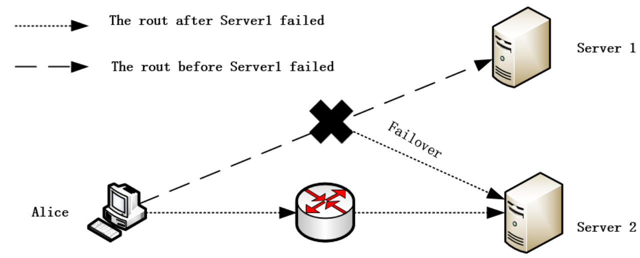

Figure 2: The Network Topology

3.1 Server selection from server pool

Similar to SNC, we implement suppression mechanism in Hippo control system. With suppression, each server delays its response by a delayed timer before returning a suppression interest message. Upon receiving suppression message, a server suppresses its pending response.

With carefully designed suppression mechanism, usually a server that is nearby responds first to a user's registration request. This process is demonstrated in our implementation of Hippo NDN-based control layer.

3.2 Server failover

The server failover in Hippo is handled by heartbeat mechanism. It becomes difficult to manage when introducing server cluster for large size of Hippo users.

The NDN-based Hippo tracker server cluster supports failover automatically without addition effort. There is no need to maintain heartbeat with tracker server. Upon failure of a server, the second nearest server from the server cluster will be automatically pick up and serve.

In our system in [2], the four different servers act a little differently upon failure. MCS and RA need to save data in the Replicate server, so that if they failed, other nodes can request the back-up and take place of them. Detail design is similar to SNC. The failover for AA is straightforward. The client only needs to send interest as normal and there will be another tracker server from the cluster pops up and automatically handles the request.

3.3 Registration process

The register function and the server-less distribution are the important features of this system. To validate the set-up, we put some information in the system. Then we can register in and access the data from different node in this system.

4. REFERENCES

[1] Kai LEI, Lihua Li, Cheng Peng, Longyu Yu, "HIPPO: an Adaptive, Scalable, Hierarchical P2P Live Streaming System", CSETIS 2014, Guilin, China, Mar.2014.

[2] Debessay Fesehaye, Jun Wei, "SNC: Scalable NDN-Based Conferencing Architecture", IEEE CCNC, Las Vegas, NV, Jan 10-13, 2014