Overlay Network for the Distribution of Streaming Media

M.F.M Firdhous¹, A.D.L.S Rupasinghe², K.T Herath³, A.G.A Lakmal⁴, A.O Weerapperuma⁵, H.M.I.G.A.B. Wijekoon⁶

Faculty of Information Technology
University of Moratuwa
Sri Lanka
¹firdhous@uom.lk, {²laksiri.sampath, ³kasunh01, ⁴asankalakmal, ⁵oshadislt, ⁶asangawije}@gmail.com

Abstract - Overlay networks create a structured virtual topology above the basic transport protocol level. It allows implementing network services that are not totally tied to the underlying network infrastructure. Many overlay network protocols have been proposed to organize nodes into various topologies with emphasis on different networking properties. Presently there are a lot of services and applications implemented on these overlay based architectures.

In this paper, the authors present an overlay based network architecture for media streaming. It allows media to be streamed to many destinations by duplicating the stream on the way down without getting it from original source at every time. The media stream is received by downstream hosts from the nearest upstream host instead of original source. For the efficient management of the overlay network, the authors have designed two protocols, namely the Cluster Protocol and the Backbone Protocol. These protocols manage the hosts in the network without affecting the Quality of Service while the hosts are allowed to join or leave the network at any time.

Keywords: Overlay Networks, Multimedia, Cluster Protocol, Backbone Protocol

I. INTRODUCTION

The Internet is the global system of interconnected computer networks [1]. In recent times overlay networks have become one of the most prominent areas in Next Generation Internet research. Overlays permit designers to implement their own applications on top of the Internet using the Internet paths between end-hosts as “links” upon which data is routed, building a network on top of the Internet. As a result, overlays can be used to deploy new functionality almost immediately, instead of requiring years of upgrades to the Internet routers; they also present developers with a flexible and powerful platform on which to create new applications and services [2].

II. RELATED WORK

Chord, Pastry, OpenDHT and RON are few overlay network presently implemented on the Internet[3],[4],[5],[6]. However, presently applications developed on these overlay architectures are often stuck to that specific overlay network implementation, because different overlay researches and implementations provide very complex different interfaces and messaging mechanisms.

Virtual Private Networks (VPNs) is an example of simple routing overlay that provides better network-level security and authentication. Other routing overlays like the Resilient Overlay Network (RON) were designed to provide better resilience to network failures than Internet routing protocols [6]. Chord provides interfaces to find peers; it uses key sharing technology for communication with them. It also supports asynchronous Remote Procedure Call (RPC) using call-back functions [3]. Pastry provides interfaces to send data encapsulated in a Message object in different mechanism than the Chord [4].

All the above projects implement a single overlay protocol, and they are based on the TCP protocol. However, there are some frameworks that can support construction of overlay networks based on various underlying transport protocols. For example, JXTA is a set of open protocols that facilitate the building of peer-to-peer networks [7].

Application-layer overlay technique is an effective way to support new applications as well as protocols without any changes in the underlying network infrastructure. For example, Qbone and Mbone utilize overlay technique to support Quality-of-Service (QoS) and multicast services respectively on top of the existing Internet infrastructure [8],[9]. The connections between each pair of overlay network nodes are provided by overlay links. As the overlay applications are usually built at the application layer, it can effectively use the Internet as a lower level infrastructure to provide advanced services to end users, such as peer-to-peer (P2P) file sharing, overlay multicasting, Resilient Overlay Networks (RON), Service Overlay Network (SON), Quality-of-Service (QRON, OverQoS), overlay media streaming etc, [6],[8],[10],[11],[12],[13],[14]. Overlay nodes cooperate with each other to provide an overlay service platform, on top of which a variety of application-specific overlays can be constructed, such as multicast overlays, any cast overlays, and end-to-end QoS-aware overlays. Sample Overlay service network is shown in Figure 1. Many overlay protocols have been proposed to organize peers into various topologies with emphasis on different networking properties such as routing path length (FISSIONE), proximity metric (Pastry) etc., [4],[15].
Overlay networks employ globally consistent protocols to efficiently route messages based on their destination keys.

III. OVERLAY TOPOLOGIES AND PROTOCOLS
Different kinds of overlay topologies are used to connect nodes on a network. Network topology is determined by the configuration of connections between nodes. For example, spanning trees, full mesh, star topology, end-system multicast topologies or random topologies are currently in use. Different topologies impact the network properties like the service performance quality or the security of overlay network in different ways. In this project the authors have implemented a topology named Cluster-Backbone topology which is based on hypercast project [16]. In this topology each node is connected to clusters by using star topology and all clusters are connected via backbone.

An overlay contains the distributed information in the network including connectivity information, location/name of super peers, closest peers, and adjacency. Overlay Protocols are used to manage this information. In this project protocols named cluster protocol and backbone cluster protocol are devised for the implementation the solution.

IV. DESIGN AND IMPLEMENTATION
Figure 2 shows the high level design architecture of the overlay based system developed in this project. In this design, a user requests a nearby user for the contents instead of the main application server. This ensures the distributed sharing of resources while protecting the system from security attacks. The system uses Cluster and Backbone protocols to build and control this overlay architecture.

V. CLUSTER AND BACKBONE PROTOCOLS
Cluster Protocol organizes the nodes of the overlay network in a star topology. Every cluster contains a cluster head and other member nodes arranged around the cluster head. Figure 3 shows some possible cluster arrangements.

Cluster head is responsible for managing the cluster members and providing the contents. Cluster head holds the information about the member nodes connected to the cluster and the resources consumed by those nodes. At the beginning, a node is in a hybrid state having the capability to transform to either a cluster head or to a cluster member. When a hybrid node connects to the overlay, it searches the network for the availability of a cluster head and based on the result of this discovery the hybrid node determines which role to play in the network. Figure 4 shows the state transition of a hybrid node.

The purpose of this cluster arrangement is to reduce the management overhead of the overlay network and to design a communication protocol that fits in to wide variety of network arrangements. Resources are exchanged with the nodes in the same cluster or nodes between different clusters.

Backbone Protocol connects multiple clusters and enables the exchange of application data between nodes in different clusters. This is done by connecting the cluster heads on another overlay topology, referred to as the backbone. In a backbone-cluster network, all clusters and the backbone belong to the same overlay network. Figure 5 shows a network that is created by the backbone protocol.
The backbone protocol disseminates information about cluster heads throughout the backbone by having cluster heads exchange overlay control and resource consumption information to neighbors in the backbone network.

**Protocol overview**

When a node enters the network, it transmits a HEAD_DISCOVERY message via multicast mechanism. The TTL value for the discovery message is set to 1 to confine the packet to the local network. Then it waits for two seconds for a reply. If there is already a cluster head present in the local network, it sends HEAD_CONFIRM message to the node which sent the HEAD_DISCOVERY message through unicast mechanism. When the HEAD_CONFIRM message is received by the hybrid node, it changes its state to member state and joins the cluster under that cluster head. If there is no reply, the hybrid node changes its state to cluster head state and starts acting as a cluster head to nodes that join the network later. The Cluster Confirm message is used to confirm the link between a cluster head and member. Head and member are now considered neighbours in a cluster. Once this logical link is established, both the cluster member and cluster head send each other HELLO messages periodically to confirm the head-member relationship.

When a node departs from a cluster, it sends a GOODBYE message to its neighbours. A cluster member sends the GOODBYE message only to its head, whereas a cluster head sends the GOODBYE message to all its members. After sending the GOODBYE message, a cluster member does not depart immediately. It responds to messages from the cluster head for some more time. During this time, whenever the node receives a message from the head, responds with a GOODBYE message repeatedly. This makes sure that the cluster head receives the GOODBYE message without fail.

If the cluster head does not receive a HELLO message for a long time from one of its neighbours, it assumes that this neighbour is no longer present. Eventually, state information about this neighbor is removed. Therefore, when a node fails and simply stops transmitting messages, state information about this node will eventually expire.

When the cluster head leaves the overlay, it sends its control and resource sharing information to a randomly chosen member within the cluster. The request for the transfer of head is transmitted via HEAD_TRANS_REQUEST message using unicast. When a member receives this message, it transforms its state to cluster head and sends HEAD_TRANS_ACCEPT message to the former head. After this, the new head informs its presence and state to every other node in the cluster via multicast HEAD_CONFIRM message. Figure 6 shows the state transition of nodes that may take place from the time a node joins an overlay network to its departure from the network.

The main interactions of the Cluster Protocol are shown in Figure 7.

If a node shares a resource in the network, the URL of the shared resource and the node’s IP address are updated in the cluster heads’ resource sharing table. The registration of the shared resource is performed via SHARERESOURCE message along with the required information to update the resource table. And if a node no longer wants to share the resource, it can de-register the resource via REMOVERESOURCE message.

Backbone protocol is used to transfer and share the cluster heads and resource sharing details in the network as well as to redirect a member to another cluster when the resource it
requires is not shared in the present cluster. The cluster and backbone protocol messages have a 16 bytes size fixed header. Figure 8 shows the format of the header.

![Figure 8: Format of the protocol header](image)

In the header MessageType is used to identify the type of the message (HEAD_DISCOVER, HEAD_CONFIRM, etc.) and the message body length is used to identify the size of the encrypted message body.

VI. IMPLEMENTATION

Figure 9 shows the block diagrammatic overview of the system and Figure 10, Figure 11 and Figure 12 show the class diagrams of Group Server, Local Server and Multicast Server respectively.

![Figure 9: Overview of the Application](image)

![Figure 10: Class Diagram of the Group Server Implementation](image)

![Figure 11: Class Diagram of the Local Server Implementation](image)

Figure 13 shows the general architecture of video streaming module.

Video streaming in this overlay based architecture is backed by two popular libraries which can be utilized under GPL license. The two libraries are,

1. Libvlc library:
   Libvlc is a library written in C++ which can be used to build applications that uses VLC an Open Source free software written by the VideoLAN project. VLC is a portable multimedia player, encoder, and streamer supporting many audio and video codecs and file formats. Libvlc library has several modules which can be used to create desired application on top of VLC. The modules of this library are listed in Figure 14.

![Figure 12: Class Diagram of the Multicast Server Implementation](image)

![Figure 13: General Architecture of Video Streaming Module](image)

![Figure 14: Modules of the Libvlc Library](image)

2. Qt application framework:
   Qt is a GUI software toolkit which simplifies the task of writing and maintaining GUI based applications for the X Window System. Qt is written in C++ and is fully object-oriented. This package contains the shared libraries needed to run qt applications, as well as the help files for qt. Qt uses C++ with several non-standard extensions implemented by an additional pre-processor that generates standard C++ code before compilation. Qt can also be used in several other programming languages via language bindings. It runs on all
major platforms, and has extensive internationalization support.

This video streaming module is capable of receiving the live streaming media from a particular host and distributing it at the same time. This is done using telnet and http interfaces to manage the video lan manager. By controlling the video lan manager, the program can add unicast or broadcast functionality to the desired media content thereby acting as a server for that particular media content.

VII. TESTING AND EVALUATION

Figure 15 shows the test environment implemented to test the application. A PC installed with Ubuntu 9.04 was configured as a router and two Layer 2 switches along with PCs were used to setup the test network.

![Figure 15: Test Environment](image)

The tests performed include the distribution of streaming video to a limited number of clients that act as relay nodes. An mpeg movie was distributed at a rate of 150 to 800kbps including audio and video streaming channels.

The tests carried out included the test for the stability of the architecture in cases of failure of nodes, the ability to integrate different media types and players and quality of the media distributed. The test results confirmed the agility of the network in case of node failures, and the quality of the stream was acceptable with no noticeable delay or delay jitter.

VIII. CONCLUSIONS

This paper presented the implementation of an overlay network that can be used to distribute streaming media over the existing TCP/IP network. The network created resembles a multicast network with nodes acting as multicast distribution points eliminating the need for multicast routers. Also the increase in the number of receiving nodes does not affect the performance of the server as the receiving nodes in turn act as redistributors for other receiving nodes. The final testing of the system showed that the network was resilient to node failures with no noticeable degradation of quality.

REFERENCES