IOSR Journal of Engineering (IOSRJEN)
ISSN: 2250-3021 Volume 2, Issue 8 (August 2012), PP 78-82
www.iosrjen.org

Application Based Route Optimization

Neha Mangla[†] , Dr.R.K. Khola

†Research Scholar, Gyan Vihar University, Jaipur, India
Professor,Department Of Electronics and Communication Engg, PDM College of Engineering, Bahadurgarh, India

Abstract: *- All the traffic seen by the internet is treated equally which is generally known as Internet neutrality. Internet Neutrality enforces that all network traffic should be treated as equal and Best effort routing policy should be followed. But with the advent of smart applications this is drastically changing. Each network application has its own bandwidth requirement. We face the problem when required bandwidth of critical applications does not match with internet bandwidth. Because of network neutrality principle, core router can't priorities one traffic over other and critical applications may get impacted. Such types of problems are still in research phase As a solution here we will see how application level routing optimization mechanism at edge routers can be useful for such types of application..*

Keywords: *CDN, ARR, DIP , ACDS, DE.*

I. Introduction[1][2][3]:

With the introduction of smart devices, there is massive explosion of smart applications on internet. Users are having access to more and more bandwidth hungry applications. The same smart devices are used both for entertainment/general and high critical applications .However these two categories of applications have different requirements but internet behavior for both is same. Internet treats the network data from each application at same priority. All the traffic in any application is digital content. The sources of content are large enterprises, web service providers, media companies, and news broadcasters. Variation in content and services delivered requires the Network to adopt application-specific characteristics, architectures and technologies. So Advance application aware routing mechanisms are required to treat delivery of such applications differently. Nowadays CDN are providing a fault tolerant ways to deliver such data but CDN generally don't take application awareness optimizations. Static data delivery is easy task for CDN While Streaming media delivery is challenging for CDNs. The current routing policies don't differentiate between application requirement and network conditions. Though there are some proprietary implementations, most of them require proprietary software at consumer and content provider end.

In this paper, we will examine a way to route user request depending upon request characteristics and dynamic network conditions to provide best possible quality of experience to the user. This approach can be deployed at service provider or CDN providers to serve user requests.

II. Routing History[4]:

IP routing history starts with static routing where administrator configures routing rules manually. This approach was fine for small networks but was not scalable enough to adapt to larger internet infrastructure. For scalable routing, over the period of time, new dynamic protocols like RIP, OSPF, BGP etc have been developed. These protocols are scalable in nature and are fault tolerant. With these protocols, routers can learn about a large number of routers and can learn new routes in case of failure of some intermediary router with minimal administrator intervention. These dynamic problems solved the internet problems till last decade. But with internet traffic explosion a new set of application aware routing policies are required to provide best QoE. In our work, software takes user request and provides better quality of services by taking current network conditions and better routing based on user application. Next section we will discuss the ways to do application awareness based routing.

2.1 Request Routing Concept for Application Awareness [5]:

2.1.1 General Request Routing System:

To solve problems based on applications a system is used called application request routing system. Application Request Routing (ARR) enables Web server administrators, hosting providers, and CDNs to increase Web application scalability and reliability through rule-based routing, client and host name affinity, load balancing of HTTP server requests, and distributed disk caching. With ARR, administrators can optimize resource utilization for application Generally a *request routing system* directs client requests to the replica server 'closest' to the client. However, the closest server may not be the best surrogate server for servicing the client

request The request-routing mechanisms can be adaptive or non-adaptive. Adaptive algorithms consider the current system condition to select a server for content delivery. Current condition of the system is obtained by estimating some metrics like load on the replica servers or the congestion of selected network links. Non-adaptive request-routing algorithms use some heuristics for selecting a cache server rather than considering the current system condition. A non-adaptive algorithm is easy to implement, while the former is more complex. Complexity of adaptive algorithms arises from their ability to change behavior to cope with an enduring situation. A non-adaptive algorithm works efficiently when the assumptions made by the heuristics are met. So our idea includes Application Awareness with Adaptive Content Delivery system (ACDS).

In next section we will discuss application awareness with DPI packets which will use ACDS.

2.1.2 Application Awareness with Deep Packet Inspection (DPI*):*

In Application based routing we will use DPI which tell us which request are most important than others. Through DPI we can inspect on line media deeply. DPI tracks the load on each server. Our software will run on edge router. On edge router we will transparently redirect all user traffic. DPI software will analyze user traffic and divide it into different priority classes. Each class of traffic will be served from internet as per internet conditions. In addition to DPI, we will be having a routing decision engine which will adapt as per network conditions to provide optimized routing behavior for current load.

III. Proposed Framework for serving critical Applications on any device [6]:

Adaptive content delivery is a system technology that transforms Web content and delivery schemes according to viewers' heterogeneous and changing conditions to enable universal access. The goal of universal access is to provide the necessary Internet infrastructure to allow users to access any information over any network from anywhere through any type of client device. The goal of adaptive content delivery is to take into account these heterogeneous and changing conditions and provide the best information accessibility and perceived quality of service over the Internet.

Here we discuss Decision making framework for optimizing adaptive content delivery. As the number of input parameters and possible output actions becomes large, the problem of developing an efficient and effective Decision Engine(DE) can become quite complex.

The basic framework for our system consists of three major parts – network discovery, decision engine and content adaptation algorithm module. The network-discovery module detects and collects all necessary information that the DE needs to know in order to dispatch a particular content adaptation algorithm.

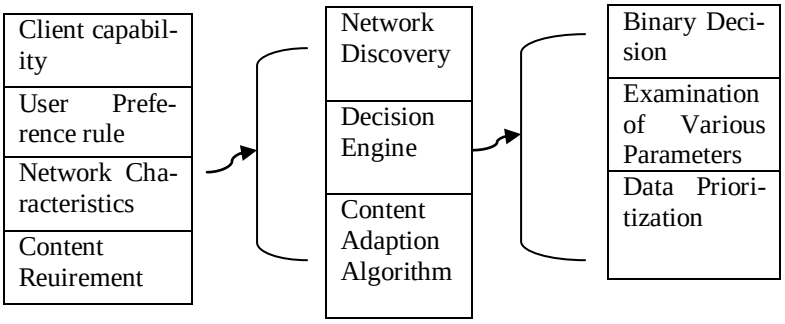

Fig.1 Efficient Decision Engine

3.1Network Discovery:

The main purpose is to provide an estimate of the available bandwidth of bottleneck path. In general software that we seek to design should be able to measure hop-by-hop available bandwidth of Internet path and should be light weight. There are large array of tools and techniques publicly available to estimate bandwidth; we first review these open source tools.

measurement techniques are of two categories: passive measurement and active measurement. Here we discuss one more method of measuring bandwidth TCP/IP Protocol Stack.

Passive Measurement works on network traces collected earlier. Although they are efficient and accurate, their scope is limited to network paths that have recently carried user traffic.

Active Measurement *uses* Active probing techniques which are more useful since they can be used to determine the instantaneous bandwidth. However, active probing techniques will require that additional probe packets are injected into a network path. Typically methods of active probing are:

Packet Pair/train dispersion(PPTD*)* : The basic idea is that the sender injects a pair of packets into the network to the destination of interest and each packet pair consists of two packets of same size sent back to back. The dispersion of a packet at a specific link of the path is the time distance between the last bits of each packet. The destination host then sends an echo for each packet. By measuring the changes in the spacing between the two packets, the sender can estimate bandwidth properties of the network path. While the packet pair mechanism has been shown to be a reliable method to measure the bottleneck link capacity on a path through the network, the use of packet pairs to measure the available bandwidth has had more mixed results.
Sending many packet pair and using statistical methods to filter out erroneous bandwidth measurements mitigates the effect of cross traffic. .

Variable Packet Size (VPS) Probing: VPS probing aims to measure the capacity of each hob along a path. In VPS probing, multiple packets of a given size are sent. This technique uses the TTL field of the IP header to force the probing packets to expire at a particular hop. The source uses the ICMP error messages received from the routers to measure the RTT to that hop. TTLs of the probe packets can be suitably designed such that the TTL of a pair of probe packets expire at each hop. The RTT to each hop consists of three delay components in the forward and reverse paths: serialization delay, propagation delay and queuing delay. The serialization delay of a packet of size L at a link of transmission rate C is the time to transmit the packet on the link, equal to L/C. The propagation delay of a packet at a link is the time it takes for each bit of the packet to traverse the link, and is independent of the packet size. Finally, queuing delays can occur in the buffers of routers or switches when there is contention at the input or output ports of these devices.

VPS sends multiple probing packets of a given size from the sending host to each layer 3 device along the path. The technique assumes that at least one of these packets, together with the ICMP reply it generates, will not encounter any queuing delays. Therefore, the minimum RTT measured for each packet size will consist of two terms: a delay that is independent of packet size and mostly due to propagation delays, and a term proportional to the packet size due to serialization delays at each link along the packet's path.
Unfortunately, VPS probing may yield significant capacity underestimation errors if the measured path includes store and-forward layer 2 switches. Such devices introduce serialization delays of the L/C type, but they do not generate ICMP TTL-expired replies because they are not visible at the IP layer. Modifying VPS probing to avoid such errors remains an active research problem.

Self Loading Periodic Stream(SLoPS): In SLoPS, a series of equal-sized packet probe trains is sent at a particular rate. Depending on the trend of one way delays experienced by the stream, the sender varies its sending rate and attempts to bring the stream rate close to the available bandwidth. If the streaming rate R is greater than the path's available bandwidth, the stream will cause a short term overload in the queue of the bottleneck link increasing the one way delays of the probing packets. On the other hand, if the streaming rate is lower than the available bandwidth, the one way delays of the probing packets will not increase. While SLoPS overcomes the inaccuracy in existing probing techniques, it requires a large number of packet streams and a very long measurement time which makes it unsuitable for real-time applications.

Train of Packet Pair (ToPP):TOPP like SLoPS, sends packet streams and gradually increases the stream rate to measure the available bandwidth.
To measure bandwidth tools are available. Bandwidth tools can be classified as either a double-end host (DS) or single end-host (SE) tools. Since DE tools have access to both ends they are generally more accurate than SE tools, but they are less scalable since cooperation of both ends is required.
Below is the list of open source tools :

Tool	**Measurement**	**Methodology**	
Bing	End-To-End Capacity	PPTD	
Bprobe	End-To-End Capacity	Packet Pair	
Cprobe	End-To-End Available bandwidth	Packet Pair	
Clink	Per-hop capacity	VPS	
Nettimer	End-To-End Capacity	Packet Pair	
Pathchar	Per hop capacity	VPS	

TCP/IP Protocol Stack

here are three properties in network stack which can be used for getting the performance parameter of TCP connection and i.e., bandwidth.

1. With a nonblocking TCP socket, if there is no room at all in the socket send buffer, we return immediately with an error of EWOULDBLOCK. If there is some room in the socket send buffer, the return value will be the number of bytes the kernel was able to copy into the buffer. (This is called a short count.)
2. Linux enables TCP Window Scaling by default.
3. We can access all TCP/IP properties (like window size, rtt) at runtime using socket calls.

The benefit of using TCP/IP stack properties to analyze network bandwidth is that it can work in non-intrusive way. The agent which is delivering data can use these properties to measure the network characteristics at run time.

To use the TCP_Info structure supported from linux 2.6 onwards, which hold the meta data about the TCP Connection. The information can be retrieved using the getsockopt() function call on tcp_info. We can view the content of struct tcp_info structure by viewing /usr/include/netinet/tcp.h

3.2 Inputs for Decision Engine:

Client Capability consist of Hardware (Screen Size ,Colors, Audio Output, CPU, Memory, Storage) and Software(OS , Audio Player, Video Player ,Browser, WAP, Scripting Language). User's Preference Profile consists of Preferred Language, Text Summarization, Sound Block, Video Block, and Image Block. Network Characteristics can be bandwidth, Round Trip Time, Congestion, and Error Characteristics. Content Requirements like Bit rate (the number of bits used per unit of playback time to represent a continuous medium such as audio or video.

3.2.1 Header Information for finding features and capabilities [7]8][9][10]:

There are different methods for detecting the features and capabilities of a user Agent. The first method is to make use of HTTP headers such as Accept, User Agent, Accept-Charset and Accept Language. The other method is to retrieve information from a UAProf (User Agent Profile) document. Accept header contain a list of MIME Media Type that will be accepted by the user agent. We can use the accept header to find out the file types that can be handled by the user agent. The user agent header contains the list of text that can be used to identify a user agent and client device. Most of the time, we can find the device model and manufacturer from the User-Agent header. It may also contain the information such as the client device's OS version, browser information, java capabilities etc. The Accept Charset header Find Character Sets that are supported by a user agent. This HTTP header is useful to non-English sites. The Accept-Language header contains information about the language preference of a user. This HTTP header is useful to multilingual sites for deciding the best language to serve to the client. User Agent Profile (UAProf) **[6] is** The World Wide Web Consortium (W3C) which is developing a standard for the process of discovering client capability and user preferences. This standard is called Composite Capability/Preference Profiles (CC/PP). It is a mechanism that allows the client to describe the capabilities and preferences associated with its user and user agent. This information includes the hardware platform, system software, applications and user preferences. User agent profiles are stored in a server called the profile repository. Very often a profile repository is maintained by a mobile device manufacturer. For example, the user agent profiles describing the capabilities of Nokia cell phones are stored in a profile repository maintained by Nokia. A user agent profile contains a number of components and each component has a number of attributes. For example, if you download and open Nokia 6230i's user agent profile, you will find the Screen Size attribute under the Hardware Platform component, like this:...

<prf:component>
<rdf:Description rdf:ID="HardwarePlatform"> <rdf:type rdf:resource= "http://www.openmobilealliance.org/tech/profiles/UAPROF/ccppschema-20021212#HardwarePlatform"/>
... <prf:ScreenSize>208x208</prf:ScreenSize>
... </rdf:Description> </prf:component>
... In general header is useful in following situations:

- To identify a specific device model with the user agent header
- To Differentiate Devices or User Agents Made by Different Companies with the User-Agent Header
- To Determine Whether a User Agent is a Web Browser on a Personal Computer or a Micro browser on a Device with the User-Agent Header

3.3 Algorithm Decision Engine based on our Idea[11][12]:

The inputs to the DE are a set of media objects contained in a document and the information about their content types, content lengths, and purposes of usage in a Web page. The DE first checks the user preferences to see if it needs to remove or substitute any redundant objects in order to save bandwidth. Other information for making decisions, such as how much a user is willing to trade off image quality for download time, is acquired by the DE at this time. The DE further checks the client capability and the network characteristics of the client. Based upon the collected information, the DE then determines if it needs to launch a particular content adaptation algorithm for a particular object.

We have separated the decision-making process into three stages. The first stage involves a binary decision that controls the dispatch of modality transform and data trans coding. Since these two types of algorithms are usually performed to enable the viewing of media objects whose original representation cannot be handled by the client, the resulting decision is either "yes" (i.e., perform the transformation) or "no" (i.e., do not perform it). If a user does not want automatic transformation, the corresponding media object would be removed because the client device cannot display it anyway.

The second stage of decision-making involves a careful examination of various parameters to find out the best trade-off between information abstractions and download time. These parameters include the user specified preference in terms of quality and response time, the current network bandwidth between the client and the server, the estimated processing time for conducting information abstraction, and the predicted output data size. The goal is to determine when it is beneficial to perform information abstraction and how much abstraction (compression) should be done.

The final stage of the decision-making involves data prioritization. At this stage the decisions for transforming all media objects into appropriate representations have been made and the server now needs to decide how to arrange them in the delivery pipeline according to their importance. Simple rules, such as text before image, image before audio, and audio before video, can be used to prioritize different media objects. Within the same media object, if an encoding scheme such as layered coding or multi-resolution image compression is supported; we can also prioritize the data accordingly. For a more advanced application, we can incorporate the user's real-time interaction or feedback to prioritize the data on the fly.

IV. Test Result :

We have made software using Linux OS and TCP/IP properties of network stack to determine the network properties. Then we tested this software on mpeg media as being opens standard, we can deep inspect media characteristics.

We used following parameters to test the setup

Total time to download data

Requests/per second

Video start time in client

Jitter

We compared its performance with normal Linux router, and Normal Linux router with squid proxy with load balancing enabled. With normal Linux router, the performance of our system was less then Linux router at low loads (as it requires extra handling at application level). But with increased load, performance of our system was much better because of better routing decisions.

We also test it against squid proxy (with caching disabled in squid and URL based load balancing enabled). We found that our system performance was better as it was adapting itself in real time as per network conditions and media characteristics.

V. Conclusion & Future Work:

With our experiments, its evident that with better routing policies, a much better quality of experience can be provided to users. Applications can be characterized as per their criticality and network bandwidth requirements which can enable a new perspective for Next generation networks.

In future, we will add support for more protocols and media types. Also we would like to test our system with other application aware routers/load balancers.

References:

[1] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl, "Globally Distributed Content Delivery, "*IEEE Internet Computing*, pp. 50-58, September/October 2002.

[2] G. Peng, "CDN: Content Distribution Network," Technical Report TR-125, Experimental Computer Systems Lab, Department of Computer Science, State University of New York, Stony Brook, NY, 2003. http://citeseer.ist.psu.edu/peng03cdn.html

[3] Real Time Audio and Video in the World Wide Web http://www.w3.org/Conferences/WWW4/Papers/211/

[4] Behrouz a. Forouzan . "Data Communication and Networking ,"McGraw Hills, 4th edition.

[5] B Cain "Known Content Network (CN) Request-Routing Mechanisms," RFC 3568, July 2003. http:// www. *tools.ietf.org/html/rfc3568*

[6] Ying Ma, Ilja Bedner, Grace Chang, Allan Kuchinsky, and HongJiang Zhang "A Framework for Adaptive Content Delivery in HeterogeneousNnetwork Environment," *Proc. SPIE 3969, 86 (1999);*January 2000.

[7] Tutorial about Detecting User Agent Types and Client Device Capabilities http://www.developershome. com/wap/detection/

[8] Mobile browser id http://www.zytrax.com/tech/web/mobile_ids.html

[9] Composite Capability/Preference Profiles (CC/PP): A user side framework for content Negotiation http://www.w3.org/TR/NOTE-CCPP/

[10] CC/PP exchange protocol based on HTTP Extension Framework http://www.w3.org/TR/NOTE-CCPPexchange

[11] RDF Vocabulary Description Language 1.0: RDF Schema http://www.w3.org/TR/rdf-schema/

[12] Mozilla Developer Center – User Agent String Reference – https://developer.mozilla.org/en/User_Agent_Strings_Reference