Software Defined Virtual Caching Mechanism Over NDN

Pavan Gannavarapu, S Kumar; Computer Science, Troy University.

Abstract— **Software-Defined Networking (SDN) allows control applications to install forwarding policies in the underlying switches, using a standard API like OpenFlow. A content delivery network or content distribution network (CDN) is a large distributed system of proxy servers deployed in multiple data centers via the Internet. In this paper we are going to present a solution, how a content is distributed based on demand on a particular router called as Content Storage Router. At the same time our main focus is to achieve maximum diversity and performance.**

Keywords—Software Defined Networking; Named Data Networking; Cache Memory; Content Storage Router;

I. Introduction

In a CDN, content exists as different duplicates on deliberately scattered servers. A large CDN can have a huge number of servers around the world, making it workable for the provider to send the same content to many requesting client computing devices proficiently and dependably - not withstanding, when data transfer capacity is constrained or there are sudden spikes popular. CDNs are particularly appropriate for conveying spilling sound, video, and Internet TV (IPTV) programming, despite the fact that an Internet administration supplier (ISP) might likewise utilize one to convey static or dynamic Web pages.

Named Data Networks (NDN) router first checks the Content Store. If there is a data whose name falls under the Interest's name, the data will be sent back as a response. The Content Store, in its basic form, is just the buffer memory in today's router. Both IP routers and NDN routers buffer data packets. The difference is that IP routers cannot reuse the data after forwarding them, while NDN routers are able to reuse the data since they are identified by persistent names. For static files, NDN achieves almost optimal data delivery. Even dynamic content can benefit from caching in the case of multicast (e.g., teleconferencing) or packet retransmission after a packet loss. Cache management and replacement is subject to ISP policies. Caching named data may raise privacy concerns.

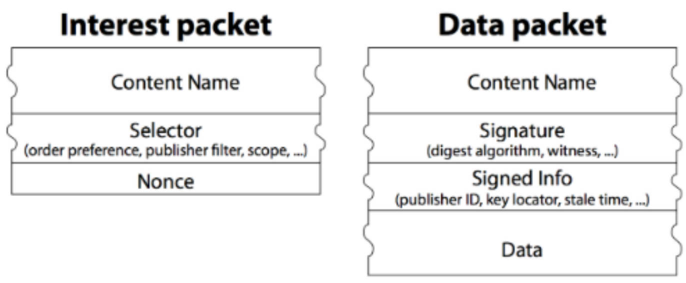
Today's IP networks offer weak privacy protection. One can find out what is in an IP packet by inspecting the header or payload, and who requested the data by checking the destination address. NDN explicitly names the data, arguably making it easier for a network monitor to see what data is being requested. One may also be able to learn what data is requested through clever probing schemes to derive what is in the cache. However NDN removes entirely the information regarding who is requesting the data. Unless directly connected to the requesting host by a point-to-point link, a router will only know that someone has requested certain data, but will not know who originated the request. Thus the NDN architecture naturally offers privacy protection at a fundamentally different level.

II. BackGround

A. NDN Architecture

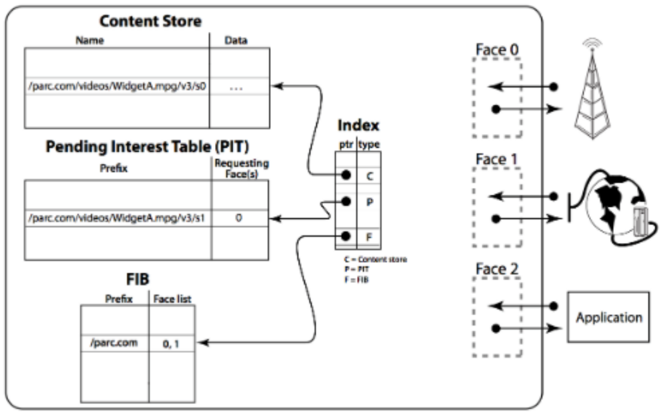
Communication in NDN is driven by the receiving end, i.e., the data consumer. To receive data, a consumer sends out an Interest packet, which carries a name that identifies the desired data. For example, a consumer may request /parc/videos/WidgetA.mpg. A router remembers the interface from which the request comes in, and then forwards the Interest packet by looking up the name in its

Forwarding Information Base (FIB), which is populated by a name-based routing protocol. Once the Interest reaches a node that has the requested data, a Data packet is sent back, which carries both the name and the content of the data, together with a signature by the producer's key. This Data packet traces in reverse the path created by the Interest packet back to the consumer. Note that neither Interest nor Data packets carry any host or interface addresses (such as IP addresses); Interest packets are routed towards data producers based on the names carried in the Interest packets, and Data packets are returned based on the state information set up by the Interests at each router hop.


Interest packet
Content Name
Selector
(order preference, publisher filter, scope, ...)
Nonce
Data packet
Content Name
Signature
(digest algorithm, witness, ...)
Signed Info
(publisher ID, key locator, stale time, ...)
Data

NDN routers keep both Interests and Data for some period of time. When multiple Interests for the same data are received from downstream, only the first Interest is sent upstream towards the data source. The router then stores the Interest in the Pending Interest Table (PIT), where each entry contains the name of the Interest and a set of interfaces from which the matching Interests have been received. When the Data packet arrives, the router finds the matching PIT entry and forwards the data to all the interfaces listed in the PIT entry. The router then removes the corresponding PIT entry, and caches the Data in the Content Store, which is basically the router's buffer memory subject to a cache replacement policy. Data takes the exact same path as the Interest that solicited it, but in the reverse direction. One Data satisfies one Interest across each hop, achieving hop-by-hop flow balance.

An NDN Data packet is meaningful independent of where it comes from or where it may be forwarded to, thus the router can cache it to satisfy potential future requests. This enables NDN to automatically support various functionality without extra infrastructure, including content distribution (many users requesting the same data at different times), multicast (many users requesting the same data at the same time), mobility (users requesting data from different locations), and delay-tolerant networking (users having intermittent connectivity). For example, consider a consumer who is watching a streaming video in a moving vehicle. The consumer may request some data but then move to a new local network. Though the Data will arrive at the old location and be dropped, it is cached along the path. When the consumer retransmits the Interest, it will likely pull the Data from a nearby cache, making the interruption minimal. Data cached close to consumers improves packet delivery performance and reduces dependency on a particular data source that may fail due to faults or attacks.



B. Content Distribution

In general, content distribution is about delivering the same piece of content to multiple clients. There are many approaches for delivering content efficiently in the network.

The simplest approach is the client-server model. Where the server stores the content and the client can only get content from the server directly. To reduce the workload of the server, caching proxies are deployed, for instance, proxies are typically deployed on the edge of the network.

The second popular approach is Content Distribution Network. CDN essentially forms an overlay network, where the same piece of content is replicated and stored at multiple locations. When users send content requests to the CDN, the CDN routes the requests to the closest content storing place. By taking advantage of locality, the content can be delivered to the user efficiently.

createCache *(<size,remove pol,write pol>)*

returns a reference to the newly created cache c

removeCache *(Cache c)*

configureCache *(<size,remove pol,write pol>, Cache c)*

getCacheStats *(Cache c)*

returns cache statistics

The third well-known technique is the Peer-to-Peer (P2P) method. In this approach, content is divided into chunks, and clients download content chunk by chunk. The global availability of the content pieces are maintained by the trackers. Once the clients finish downloading some content chunk, it serves as a content provider for that chunk. This way, the border between the clients and the servers blurred. P2P is still very popular in the network, but there are studies show that the percentage of the P2P traffic in the network is not as significant as before.

In our NDN content distribution evaluation, we use the simple client-server approach, as the underlying NDN network architecture provides the caching functionality by design.

III. Software Defined Virtual Caching Mechanism

A. System Definition

The content distribution system in NDN consists of three components, NDN Clients, NDN Servers and NDN routers. All three of these roles run the same ccnd daemon program, so that all of the hosts involved in the system have the capability of forwarding and caching packets. The clients run client-end software while servers run server-end application on top of ccnd. The router doesn't run any other application, but is dedicated to the forwarding functionality. To distribute a content, the content is initially stored in the server, the clients download the content by sending out interest packets. The server application replies to the interest packet by sending the content packet. When the content packet is forwarded along the path, it is cached in the NDN router's content store.

B. Parameters and Factors

In computer network systems, there are many parameters that can affect the system performance. We present all the parameters for the NDN-based content distribution system in this section. The system parameters are introduced first, followed by the discussion of workload parameters. The selected factors are listed in the end.

i) System Parameters: Our performance study is conducted in the ONL. The hardware parameters cannot be changed easily, while some of the software parameters, for instance, the configuration of the CCNx on each node could be modified.

ii) Workload Parameters: Since there is no real-world NDN packets trace available, we have designed and used synthetic workloads for this measurement study.

iii) Factors: The factors are selected from the parameters listed above. By changing the Content Store capacity, the number of pieces of content that can be stored in the NDN router is configured. Generally speaking, a larger CS can hold more content and thus provide a higher probability of satisfying the incoming requests. In the mean time, larger CS does increases the lookup time. Recall each NDN packet is assigned a unique name and packets are routed based on their names. The length of the NDN packet name in this paper is defined as the number of components in the name. Longer names increases the name lookup time and thus affects the system throughput. The content diversity is defined under the assumption that every client only asks for one content, the value of content diversity is the number of different contents being requested from the clients. For instance, if all of the clients ask for same content, the content diversity is 1. Similarly, if all of them ask for different content, the content diversity is the same as the number of clients in the system.

D. Goals

Today our aim is to

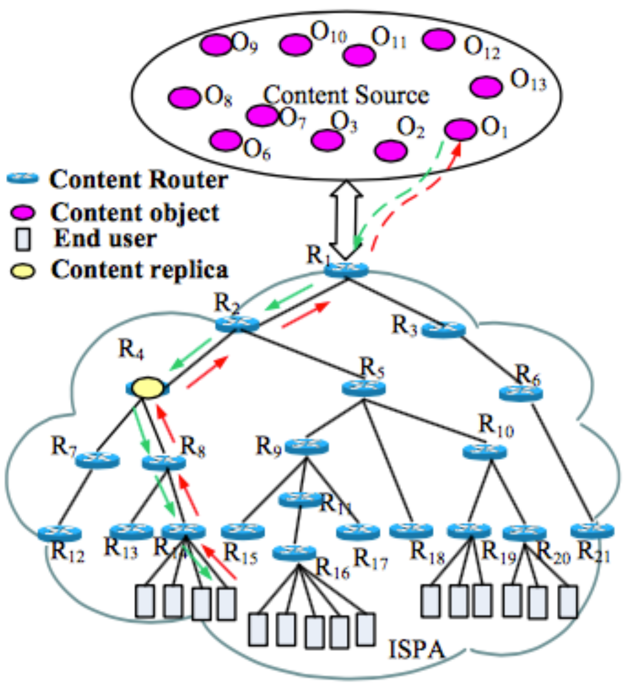
1. Increase the Performance.
2. Increase the Diversity of the Content.

Algorithm: Performance Measurement

```
for layer $\leftarrow$ root level to bottom do
    for each node Rj in the layer do
        if $Rj \in U$ then
            Req[]j $\leftarrow$ getCacheStats( )
        Else
            Req[]j $\leftarrow$ createCache
            (<size,remove pol,write pol>)
        end if
            Aj $\leftarrow$ Sort-Dec (Req[]j,
            remove pol)
    end for
end for
Return(Aj)
```

In order to increase the performance we need to change the cache size.

Performance P = Replace LRU/Number of machines that actual packet contains.



Algorithm: Diversity of the content.

```
At Router A[i]
for (i=1; i= Mi ; i++)
        if ( n is greater than Mi )
                $q = n \times p;$
                $t = i \div q;$
                return t;
        endif
```

Diversity of the Content:

$$N = \Sigma\ {}^{M}i / (n \times p)$$

${}^{M}i$ is the number of machines storing the ith packet.

n is the total number of pieces of diversity.

p is the total number of machines.

IV. References

1. *http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.366.6736&rep=rep1&type=pdf*
2. *http://www.cse.wustl.edu/~jain/cse567-11/ftp/ndn/index.html*
3. *Software-Defined Caching: Managing Caches in Multi-Tenant Data Centers - Ioan Stefanovici⋆, Eno Thereska, Greg O'Shea, Bianca Schroeder⋆, Hitesh Ballani, Thomas Karagiannis, Antony Rowstron, Tom Talpey†. University of Toronto⋆, Microsoft Research, Microsoft†*
4. *Rule-Caching Algorithms for Software-Defined Networks. Naga Katta1, Omid Alipourfard2, Jennifer Rexford1 and David Walker1 1Princeton University ({nkatta,jrex,dpw}@cs.princeton.edu) 2University of Southern California ({ecynics}@gmail.com)*
5. *NDNFlow: Software-Defined Named Data Networking. Niels L. M. van Adrichem and Fernando A. Kuipers Network Architectures and Services, Delft University of Technology Mekelweg 4, 2628 CD Delft, The Netherlands {N.L.M.vanAdrichem, F.A.Kuipers}@tudelft.nl*
6. *Popularity-driven Coordinated Caching in Named Data Networking. Jun Li, Hao Wu, Bin Liu, Jianyuan Lu, Yi Wang Dept. of Computer Science Tsinghua University, Beijing*
7. *https://onl.wustl.edu*
8. *http://www.cse.wustl.edu/~jain/cse567-11/ftp/ndn/index.html#ONL*