Technical Report: Named Content Delivery Network

Xiaoke Jiang
jiangxk10@mails.tsinghua.edu.cn

Jun Bi
junbi@tsinghua.edu.cn

Institute for Network Sciences and Cyberspace, Tsinghua University
Department of Computer Science, Tsinghua University
Tsinghua National Laboratory for Information Science and Technology (TNList)

ABSTRACT

CDN (Content Delivery Network) focuses on delivering requested data to users, no matter where the data comes from; but the fundamental goal of IP is to connect hosts. The essential mismatching leads to complexity and inefficiency. More specifically, 1) CDN has to build components to map *what* to *where*, which is resource consuming; 2) CDN has to monitor real-time network state on the application layer, which is complex and not accurate. In contrast, NDN (Named Data Networking), provides the information and function that traditional CDN devotes a great deal of effort to achieve, since NDN routes by name, its routing plane holds the"what", information of content distribution, and its stateful forwarding plane can detect and adapt to dynamic of the Internet. Thus this work enhances current CDN with NDN, here dubbed Named Content Delivery Network, or nCDN. In nCDN, CDN itself focuses on services such as accounting, data analysis etc; NDN runs over IP and takes charge of content routing and delivery.

nCDN is more adaptive to the dynamic of the Internet and improves the performance, especially in a scenario where content copies are hosted in several hosts. nCDN makes it easier to implement optimization solutions and CDN Interconnecting. Our simulations demonstrate that nCDN is better than traditional CDN on almost all aspects, including the scalability, reliability, and QoS.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

General Terms

Design

Keywords

ICN; NDN; CDN

1. INTRODUCTION

Over the past few decades, our population has witnessed the growth, maturity and prosperity of the Internet. Network information and traffic have experienced an explosive growth. As a result, popular web services have often suffered from congestion and bottlenecks due to large demands from their huge user-base. A lot of technologies are proposed to address this problem. Some technologies, such as web cache and CDN, mainly aim to provide overall performance using redundancy resources; while some other technologies, such as multipath TCP, multicast and NDN, which enhance existing design or even create new design, mainly focuses to improve data transmission. Generally, These two categories of technologies can cooperatively work to achieve better performance and the former technologies are usually built over the latter kind.

CDN, which make use of geographical distributed data replicas to cover the large amount of concurrent request from all over the world, focuses on delivering requested data to users, no matter where the data comes from; but the fundamental goal of IP is connecting specified hosts. The essence mismatching leads to complexity and inefficiency of existing traditional CDN, which keeps connection-oriented data transmission schema, delivers data after selecting and connecting the host which holds the requested data. The drawbacks include:

- Have to maintain mapping from "what" to "where", which is resource consuming. For example, query-based inter-cluster operative caching.
- Content multihoming, referring to the scenario that the same content(s) is/are provided by several hosts, is not fully utilized. For example, request is redirected to original server after a cache miss of web page content on surrogate.
- Multicast is not well supported. ALM (Application Layer Multicast) is used to transmit live stream only, while webpages and VoD (Video on-demand) do not get any benefits.

Meanwhile, IP does not provide enough information about network state, especially real-time state. As a result, CDN has the following pitfalls:

- Heavily rely on global view, provided by centralized controller and real-time monitor.

- Congestion control and load balance is not well supported.

Note that the above disadvantages are related to content delivery, which is the core part of CDN. To cope with the problems, CDN has to build a lot of components. As a consequence, traditional CDN is bloated and complex. In Section 2 we will refer to detailed CDN design and implementation, which expose the above drawbacks.

In contrary with IP, NDN focuses on data itself and ignores the data container. In fact, NDN shares some design elements with CDN, for example, named contents, cacheable nodes, and redundancy resources. What is more, NDN's routing plane holds information of content distribution and stateful forwarding plane detects and adapts to dynamic of the Internet. The intrinsic consistency makes NDN a perfect underlying layer for CDN:

- Lightly rely on global view, since NDN's routing plane and forwarding plane have provided enough information.
- Naturally support multicast
- Performs very well in content multihoming scenario, due to name-based routing.
- Build-in hop-by-hop congestion control and flow regulation. The stateful forwarding plane and Strategy Layer are flexible to adapt to the dynamic of network.

Thus this work enhances current CDN with NDN, here dubbed Named Content Delivery Network, or nCDN. nCDN embeds NDN into its architecture to take charge of content routing and delivery. Considering that IP is the de facto network protocol, NDN in nCDN runs on IP. And thus, CDN will be simplified and focus on high-level services, such as accounting, data analysis, etc. Those services is built on application layer and can run on any network protocol. In fact, we can just leave those services unchanged and run on IP as out-bound channel.

nCDN is more adaptive to the dynamic of the Internet and improves the performance, especially in the scenario that content copies is hosted in several hosts. nCDN makes it easier to implement optimization solutions and CDN Interconnecting. Our simulations demonstrate that nCDN is better than traditional CDN on almost all aspects, including the scalability, reliability, and QoS.

This paper is organized as follows. First, backgrounds on NDN and CDN are listed in Section 2. The methodology of adopting CDN over NDN is described and discussed in Section 3 and Section 4, respectively. Performance is evaluated in Section 5. Finally, we conclude the paper in Section 6.

2. BACKGROUND

In this section, we give a brief introduction on CDN and NDN, respectively.

2.1 Content Delivery Network

2.1.1 CDN Evolution

Several solutions attempted to improve the Quality of Service (QoS). One approach is to modify the traditional web architecture by improving the web server hardware, such as processor, memory and disk and this approach evolves to server farm, which is comprised of multiple web servers sharing the burden of answering requests for the same website. Server farm shows better scalability and fault tolerance; however, it does little to improve the network performance due to network congestion.

Another approach is caching proxy deployment by Internet Service Provider (ISP). ISP constructs different levels of local, regional, international caches at geographically distributed locations, referred to as hierarchical caching, resulting to additional performance improvements and bandwidth savings. This approach requirement end users to configure their browsers to send their web request through caches instead of directly to the original producers.

To address those limitations of the above approach, CDN is proposed, originated from a MIT research laboratory. CDN connect computers together across the Internet to cooperate transparently for delivering content to the end users.

The first generation of CDNs focused on static or dynamic web contents, while the second generation has shifted to Video-On-Demand (VoD), audio and video streaming. CDN technologies are still on development. The latest generation of CDNs show the trend to interconnect different CDNs so they can interoperate and collectively behave as a single delivery infrastructure.

2.1.2 CDN Design

In the context of CDN, content consists of two main parts: the encoded media and metadata. The encoded media includes static, dynamic and continuous media data, such as audio, video, documents, images and web pages. While metadata is the content identifier that is used to content identification, discovery and management.

In the context of CDN architecture, there are three key stakeholders: content producer, CDN provider and end user. CDN providers build caching and/or replica servers, which are called surrogates or edge servers, in different geographical location. end users' requests are redirected to the best surrogate and the selected surrogate is in charge of delivering contents to the end users, which achieve transparency for end users.

There are some key techniques in the context of CDN, including content distribution and management, request-routing, performance measurement [1][1].

Content distribution and management include surrogate placement, cached content selection and delivery, content outsourcing and cache organization. request-routing includes request-routing algorithm and request-routing mechanism. performance measurement is in charge of measuring internal and external network state.

In this paper, nCDN makes a breakthrough on data transmission techniques, such as content outsourcing, request routing and content delivery. Request-routing, which is responsible for routing client requests to their appropriate surrogates, is one of the most important techniques. Request-routing includes two parts, one is the request-routing algorithm, which selects the best surrogate; the other is the request-routing mechanism, which informs the end user about the selection.

Request-routing algorithm encompasses adaptive and non-adaptive algorithms. Adaptive algorithms takes the real-time system conditions into consideration while non-adaptive ones make use of heuristics. Non-adaptive algorithms are simple but not very accurate, especially in the face of events like flash crowds [2]; while adaptive algorithms are complex and robustness. Adaptive algorithms rely on monitoring and detecting the real-time network state, such as client-server latency [3], path length [4], etc.

Request-routing mechanism includes Global Server Load Balancing (GSLB) [5], DNS-based request routing [6], HTTP redirection, URL rewriting etc. GSLB and DNS-based request routing resolve domain name into numerical IP address of best surrogate. HTTP redirection leads to lack of user transparency problem and overhead. URL rewriting leads to delay for URL parsing, the possible bottleneck and non-cachebility of the contents. In reality, HTTP redirection, DNS-based request-routing and GSLB are the most widely used request-routing mechanisms.

Content delivery is in charge of delivery content internal CDN. High-performance transmission system is built. Techniques such as tried distribution, path optimization, packet loss reduction, transport protocol optimization, etc, are used to improve the efficiency. Furuther more, traditional CDN keeps connection-oriented transmission.

2.1.3 CDN Pain points and Challenges

The real commercial CDN is a quite complex system, which consists of complicated subsystems. For instance, the latest design and architecture of Akamai[7] includes edge server platform, mapping system, communication and control system, data collections systems and service, which are made up of log collection, real-time data collection and monitoring, analytics and reporting; and other additional system services, including DNS, monitoring agents, global traffic manager, storage, client side delivery, management portal.

What is more, there are also some technical challenges, especially on real-time monitoring. Technical challenges of CDN [8] include:

- Monitoring and controlling tens of thousands of widely distributed servers, while keeping monitoring bandwidth to a minimum.
- Monitoring network conditions across and between thousands of locations, aggregating that information, and using it to generate new maps every few seconds.
- Reacting quickly to the change of network state and workloads.
- Measuring Internet conditions at a fine enough granularity to attain high-probability estimates of end-user performance.
- Integrity control. A server must ensure that each client request receives the correct response.

2.2 Named Data Networking

NDN[9] or Content-Centric Networking [10] is one of the most promising next generation Internet architecture, which replaces *where* with *what*. Packets carry the *name* of the data instead of source and destination addresses.

There are two kinds of datagrams in NDN architecture: Interest and Data. End users send Interest which contains the name of the Data they want, NDN routers forward Interests by their names, which is called name-based routing, and create PIT (Pending Interest Table) entries to record of the Interest, incoming face and outcoming face. If the Interest meets its corresponding Data, the Data travels the reverse path marked by PIT entries back to the originator of an Interest. Data is cached in Content Store by routers, which decouples requests and responses in both time and space: data requester and producers need not to known each other's location, nor need they be online at the same time [11].

The seminal paper by Jacobson et al. [10] presents the blueprint of NDN architecture, and Cheng Yi et al.[12] gives in-depth analysis on forwarding of NDN, which is a highlight (also challenge) of NDN and benefits from its stateful forwarding plane. After forwarding Interest packet, NDN routers maintain the state of every pending Interests that guide Data packets back to the consumers. Thus, routers are aware of the network state by observing the two-way traffic. In this case, immediate routers can explore multiple alternative paths without leaving the task to the end consumer. In other

[1]There are other different terms used to describe those techniques, but functions are basically overlapped

words, NDN, by the maintaining massive states, makes itself adaptive and intelligent.

Cheng Yi et al.[12] introduces Interest NACK and coloring-based adaptive forwarding strategy into NDN architecture in order to make the best of stateful data plane.

In the original sketch of NDN, router starts a timer for the forwarded Interest based on estimated RTT. If the timer expires, the router may try an alternative path. However, timer is such a simple design lacking of sensitivity and promptness. What's worse, the downstream router cannot be acknowledged but only wait until the timer expires. Interest NACK addresses to fix this issue. Interest NACK is designed to be a special kind of Interest. When an NDN node can neither satisfy or forward an Interest (e.g., no available face for the Interest, duplicated Interest, or congestion), it sends back an Interest NACK with the corresponding name and an error code, to the downstream node. Then the downstream node will take proper actions based on the error code. If the downstream node can do nothing for the error and has no alternative face to retry, it may send Interest NACK back to its downstream node recursively. But Interest NACK is different from ICMP messages, NACK is sent to the previous node, which exhibits hop-by-hop feedback, while an ICMP message is sent back to the source host, which is exactly end-to-end flow and congestion control.

Coloring-based adaptive forwarding strategy get the idea from traffic light, marking paths different colors: Green, Yellow or Red. When a new FIB entry is created or a new interface is added to a FIB entry, the interface's initial status is Yellow. It turns Green when Data flows back from that interface. A Green interface turns Yellow when a pending Interest times out (i.e., no Data comes back within the expected time), after Data ceases flowing for a certain amount of time, or upon the receipt of a "No Data" or "Duplicate" NACK. An interface is marked Red if it goes down. A "Congestion" NACK does not change the color of an interface but reduces the rate that Interests can be sent through that interface. Green interfaces are always preferred over Yellow ones; Red interfaces are never used to forward Interests.

Detailed information about coloring-based adaptive forwarding and Interest NACK can be found in the technical report [13].

3. METHODOLOGY

In this section, we focus on nCDN architecture. Comparison between traditional CDN and nCDN is shown in Figure 1. The key difference is, traditional CDN has to build some transmission components, which consist of real-time data collection and monitoring, monitoring agents, etc. The transmission components aim to build high-performance stream and content delivery network. To facilitate this, nCDN introduces NDN and removes the transmission components. As a result, all data plane traffic (excluding control plane traffic) is carried by NDN.

NDN employs name-based routing, which is suitable in this scenario. It also revolutionizes content delivery, by changing it from connection-oriented to content-oriented.

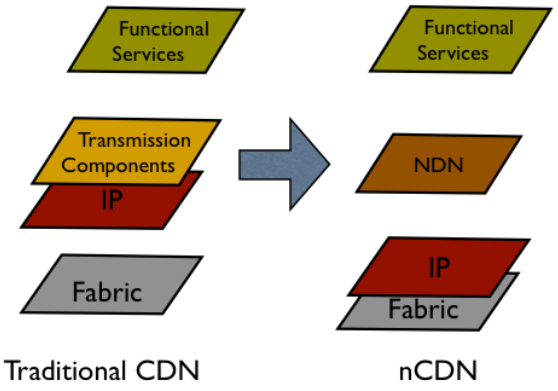


Figure 1: CDN Architecture

3.1 Metadata and Name

Metadata, in the context of CDN, plays key roles in identification, discovery, and management. These functions are exactly what *Name* does in the context of NDN. Therefore, the most fundamental naming requirement of NDN is inherently satisfied by traditional CDN. To simplify the process, we use metadata to construct the name of contents. On the other hand, requests from end users should be translated into Interest, which contains a corresponding name.

Traditional CDN selects a best surrogate to satisfy users' requests and keeps a connection-oriented schema. In comparison, nCDN uses a content-orientated schema, translating requests into Interests and retrieves the corresponding Data with name-based routing in a content multihoming scenario. Thus, data transmission evolves from IP to NDN.

3.2 Request Routing Algorithm and Mechanism

For traditional CDN, goals such as load balancing, traffic engineering or lowering cost etc, are integrated into routing-request algorithm, which attracts a lot of research attention. With nCDN, we can achieve those goals with NDN, since NDN provides a flexible and powerful forwarding strategy. Thus, routing-request algorithm becomes very simple, directing the request to the nearest surrogate. We call this surrogate the entry point, since it is the ingress point for request to the

nCDN network. We discuss this topic in Section 4.3

In order to balance loads on entry points, non-adaptive routing request algorithm such as random and round-robin can be applied. Adaptive routing also works, but it seems to be complex and insignificant.

As for the request routing mechanism, existing solutions also work under our nCDN architecture: For example, the most widely used DNS-based request routing mechanism.

3.3 High-Performance Transport System: Content Routing & Delivery

Cisco predicts that by 2014, video traffic will take over 90% of the overall Internet traffic [14]; YouTube announced that it now receives 2 billion views per day in 2010[15]. In light of this significance we use video transmission as the case to show how NDN improves CDN.

Video streaming is bandwidth consuming and time intensive. CDN applies different approaches to reduce the number of requests back to the original server. Firstly, CDN tries its best to direct video requests to a surrogates if possible, intra- or inter- cluster caching approaches are used. Furthermore, CDN also creates special manner to handle cache miss, for instance, Akamai uses a tier distribution platform to achieve this goal. With tiered distribution, a set of Akamai "parent" clusters of surrogates are utilized. These clusters are typically well-provisioned clusters, chosen for their high degree of connectivity to edge clusters. When an edge cluster does not have a piece of requested content in cache, it retrieves that content from its parent cluster rather than the original server. But from the outside, the parent clusters looks capable of originally providing those contents, the parent clusters can be treated as normal surrogates.[2]

For nCDN, the problem is solved with name-based routing. If a surrogate (including parent clusters) can originally provide content set with the name prefix, then it announces the prefix into NDN's routing system. All nodes that satisfy this will announce the prefix. Routing protocol (such as OSPF) will generate routing tables according to these announcements, thus, a specific prefix may have several alternative paths. IP will choose the best path for a specific IP prefix, based on a relatively static routing table; While NDN can make full use of multihoming, detecting the real-time state of alternative paths, and choosing the immediate best path based on latency, congestion etc. NDN architecture even includes a "Forwarding Strategy Layer", which can take full advantage of content multihoming. Here we demonstrate a single content routing and delivery case, using a toy topology to explain the process. In the Figure 2, all circles represents nCDN surrogates, which are NDN

[2]Their routing metrics should get special care

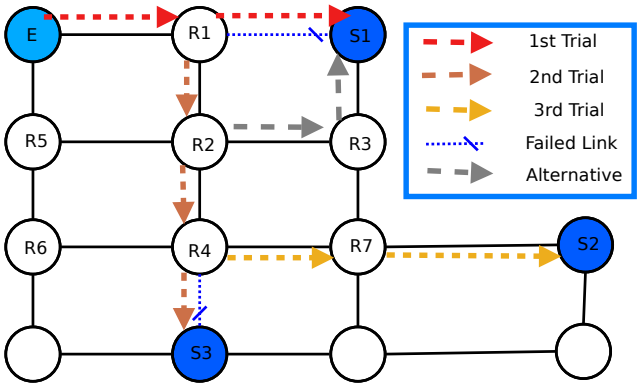


Figure 2: Toy Case to Show Content Routing and Delivery

capable. The node labeled "E" is the entry point for the data transmission in our case. Requests will be translated into NDN packets. Nodes labeled "S1", "S2", and "S3" are the three CDN surrogates, they hold the same data set and announce the same name prefix, which can originally provide the contents requested in this case, which we call *content hosting surrogates* of the request. In our case, all the other nodes cannot originally provide the requested content, but can forward Interest to the right surrogates. Links are represented by black solid lines, connecting adjacent nodes forming a specified topology. We assume that the routing metric and delay of each link is the same. In our case, links R1-S1[3] and R4-S3 are failed, which may be due to physical damage or temporary congestion, therefore they are unable to serve the request in our case. By default, NDN nodes try different alternative surrogates one by one. In addition, we also assume Interest is not satisfied by cached data and the corresponding PIT entries does not timeout.

The following steps show the process of content routing and delivery:

Step 1. Requests from end users enters the NDN network from E, for example, our previous work [16] tried to generate NDN Interest according to HTTP request. E chooses R1 assuming its intent is to direct the request to S1, as its first trial, because the minimized path cost of choosing R1 is 2 (E-R1-S1) while that of R5 is 4 ($C-R5 \sim S1$, or $C-R5 \sim S3$). Thus, node E sends Interest packet to R1.

Step 2. From view of R1, S1 is its next best hop. So Interest packet is forwarded to S1.

Step 3. Normally, S1 sends back the requested data and

[3]Note here "–" means direct link connection and "∼" means logic connection, which may have several real paths

the data packet traces the reverse path of Interest packet to arrive at node E and the transmission succeeds. However, link R1-S1 is failed, R1 gets no data back. In this case, R1 tries another path and forwards the Interest to R2 from where the Interest may arrive at S1 or S3. This is the second trial.

Step 4. There are two best choices for R2: R2-R4-S3 and R2-R3-S1. Forwarding strategy implies that nodes try alternative paths one by one. In our case, we assume R2 tries path R2-R4-S3 first, thus, Interest is sent to R4.

Step 5. Since link R4-S3 is failed, Interest is forwarded to R7, then to S2. S2 provides Data, which follows the path S2-R7-R4-R2-R1-E.

Step 6. E gets the Data, then encapsulates the digital media to IP packet(s) and sends it back to the end user.

Different forwarding can be employed here. For example, another forwarding strategy is to try the best paths simultaneously. In this situation, R2 get Data from R2-R3-S1 (the alternative path), while Data from R2-R4-R7-S2 is ignored, since the PIT entry is removed once the first Data comes. This strategy minimizes the request latency at the price of more resources, since it requires more bandwidth, creates more entries and trials, while only one path will be utilized.

However, in the above case the best forwarding strategy for R2 is to try path R2-R3-S1 and get the Data back immediately without trying path R2-R4-S3. This should, and can, normally happen during a continuous content delivery, because NDN can detect the real-time network state. Once failed links R1-S2 and R4-S3 are detected through past experience, these corresponding trials will be blocked. At R2 delay of choosing R3 and R4 is measured, in our case R3 is preferred. Coloring-based forwarding strategy realizes this approach.

As we can find from the above case, if needed, NDN would try to select the best surrogates from all surrogate sets (S1, S2 and S3) for every transmission. This NDN-based content routing and delivery has the following advantages:

- Simple: Do not need the support from application layer information, NDN can handle congestion, flow balance, and cache miss.
- Adaptive and Accurate: NDN makes decisions based on real-time network state. Thus, it is accurate and works better under a dynamic environment.
- Effective and Robust: NDN gets rid of congested paths and distributes its flow traffic to free paths. This remits congestion and produces positive effects for the whole network.

3.4 Content Selection & Outsourcing

Full-Site and Partial-Site content selection are supported. Partial-Site approach could announce longer prefix in order to decrease cache miss, what is more, the original server or parent clusters of surrogates should announce the longer prefix, in case of cache miss on edge surrogates.

There are some contents on original server which need to be distributed into some specified surrogates. This is content outsourcing. Current commercial CDN uses non-cooperative pull-base approach, which means, every surrogate sends requests to the original server. Cooperative pull-based approach, which reply on complex DHT (Distributed Hash Table) is under research. While for nCDN, surrogates naturally cooperate:

- Surrogate Level Cooperation: If one surrogate can provide a data set, the name based routing directs the requests to the best surrogates, which do not have to be the original server.
- Request Level Cooperation: Same requests from multiple surrogates aggregate together, leading to multicast, which is more efficient than multiple point-to-point transmission.

3.5 Techniques Irrelevant to Data Transmission

Since NDN only changes the data transmission of CDN, techniques which are irrelevant to data transmission would work on nCDN architecture directly, or with little change. Traffic generated by these functions, referred to as control traffic, can be carried by IP as outbound channel, since the control traffic may be beyond CDN infrastructure.

3.5.1 Accounting

Accounting and Billing is an extremely important function for commercial CDN. Since every piece of data is named, every request is received and responded at entry points; and latency, packet loss, average bandwidth, etc. can be measured, making accounting easier. As to access control, original authentication and authorization mechanisms still works and NDN provides extra data-oriented security to improve data integrity and stop those illegal readers, which makes accounting more credible.

3.5.2 Surrogate Placement

nCDN can support both flexible surrogates placement strategy, including single-ISP and multi-ISP approaches. Surrogates, running intra-domain routing protocol, construct an organized network automatically, which limits the necessity for human management.

3.5.3 Cache Organization

NDN itself is a cache capable protocol. Cache is inherently supported. Cache update policies and caching techniques are widely researched in the NDN community, which can be combined with existing CDN research.

3.6 Platform Components

Since routing plane of NDN holds the information of content distribution and forwarding plane detects the real-time network state and adjusts forwarding choice dynamically,Some components can be removed from CDN architecture, while some others can be simplified. Basically, the components mapping what to where or detecting network state, can be removed, while components which monitor the network state or is related to traffic engineering, can be simplified or removed.

To make it clear, we take Akamai's design[7] for instance. The following components probably can be removed or simplified: Real-time Mapping, Mapping Scoring, Global Traffic Manager, Communication and control system, Monitor Agent.

4. DISCUSSION & FUTURE WORK

4.1 Cache Miss Ratio v.s. Routing Scalability

Cache miss can happen even when the surrogate announces the corresponding prefix. This is because not all the contents within the announced name prefix are hosted in the surrogate, due to inappropriate announced prefix, dataset update or cache update.

According to analysis in Section 3.3, the request will finally try all the content hosting surrogates and the original server in order to retrieve the content. Thus, logical content routing and delivery system can handle cache miss, but the process is resource consuming and time intensive. So we should try best to decrease cache miss ratio. The following rules may help:

- Announce the prefix only if you can originally provide almost all the contents whose name beginning with the prefix, special care on naming should be taken into consideration.
- Prefer to announce longer prefix, in order to get close to the above rule.

However, these two rules increase the number of entries in the FIB (Forwarding Information Base) of NDN and lead to routing scalability problems. Van Jacobson et al [17] proposes custodian-based routing. In their scenario, different nodes contain different subsets of data, with the same name prefix, and all announce the prefix. A node's ID and interfaces are taken into consideration in order to get better scalability and efficiency. We believe this helps to decrease cache miss ratio and limit the routing table to a reasonable size.

4.2 CDN Interconnecting

As a result of the significant growth in content delivered over IP networks, existing CDN providers are scaling up their infrastructure and many Network Service Providers and Enterprise Service Providers are deploying their own CDNs. Subject to the policy of the CSP, it is generally desirable that a given item of content can be delivered to an end user regardless of that end user's location or attachment network. This creates a need for interconnecting (previously) standalone CDNs so they can interoperate and collectively behave as a single delivery infrastructure. An IETF Working Group named CDNi (Content Delivery Network Interconnecting) [18] is established to solved this problem.

The goal of the CDNi Working Group is to allow the interconnection of separately administered CDNs in support of the end-to-end delivery of content from CSPs through multiple CDNs and ultimately to end users (via their respective User Agents).

In our nCDN context, this problem is translated to inter-domain routing. Core IP routing protocols, BGP, IS-IS and OSPF, can be used as-is to deploy NDN [9], and OSPF-based Intra-domain routing is implemented and proved to work well in NDN scenario [19]. The remaining BGP-based protocol has yet to be proven, however, we believe inter-domain routing protocol of NDN helps to solve CDN interconnecting.

4.3 Forwarding Strategy v.s. Request-routing Algorithm

Lots of potential area can be explored to optimize the performance of CDN according to different metrics. Traditional CDN request routing improves application performance metrics such as network delay [8], server load [20] and system throughput [2], but incurs significant ISP cost. In contrast, CDN request routing in [21] reduces ISP cost by exploiting the concave nature of ISP charging functions. Request routing in [22] reduces CDN server charging volume as a means to reduce ISP costs for CDN. All these solutions can be adopted in nCDN as they are, however, in this paper we argue that forwarding strategy is a more optimal method.

NDN holds information on content distribution and detects network state in real-time, providing details for optimization solutions. We choose Forwarding Strategy because it is flexible, especially when combined with intra-domain traffic engineering solutions. The control capabilities are powerful enough to achieve different optimizing goals from inside the infrastructure directly, but not outside. This problem needs further exploration and is one of our future works.

5. EVALUATION

CDN is designed for scalability, reliability, performance[7]. In this section, we will evaluate these fea-

tures.

We use ndnSIM[23] to evaluate nCDN. A commercial CDN is a very large distributed system which may consist of thousands of globally deployed servers that run sophisticated algorithms to enable the delivery of highly scalable distributed applications. We have tried our best to make our simulation closest to reality. But due to lack of public CDN topology, we use Rocketfuel's AT&T Point of Presence (PoP) topology[24] instead, which contains 625 nodes, including 221 backbone routers, 108 gateway routers and 296 leaf routers, and about 2,101 links with bandwidth, ospf metric, transmission delay and queue capacity. All the nodes can be treated as surrogates of CDN in reality, which interconnect together by IP. In our nCDN scenario, the surrogates are NDN capable.

The dataset of CDN comes from a latest sigcomm paper[25].[4]. The dataset contains more than 100,000 objects, whose request frequency distribution follows the zipf law, with parameters of 0.99 in US, 0.92 in Europe and 1.04 in Asia, respectively. We choose 100,000 objects and 0.99 as our zipf parameter.

We randomly pick some gateway routers to serve as surrogates that can originally provide the data in our simulation. All leaf routers serve as entry points.

nCDN is built with a coloring-based forwarding strategy and NDN routers try available ranked faces one by one. Traditional CDN is built with a GSLB. nCDN relies on NDN to find the best copy and retrieve the content; while traditional CDN replies on the GSLB to select best surrogate and get the data with end to end channel. If a request from an entry point fails to get data back, GSLB will select anther surrogate for the entry point in traditional CDN scenario. A filter mechanism is also introduced to stop frequent and unnecessary re-selections, since congestion or failed node/link leads to continuous unsatisfied requests; while nCDN totally leave requests failure to NDN. By default, entry points do not re-issue the Interest if the Interest does not get Data back.So we can check the result at granularity of every request, excluding retransmission.

5.1 Reliability

In this scenario, suppose that node down and link fail events will be found and solved by CDN within 10 seconds. So we only need to monitor the performance during 10 seconds. In the experiment, after 5 seconds of warming up, we let a key node or link go down, then monitor the unsatisfied requests. We collect the accumulative number of unsatisfied requests every 0.5 seconds. Figure4a and Figure 4b are drawn under the settings that number of content hosting surrogates is 15. As we can find from Figure 4a, traditional CDN line shows instability and some extraordinary peaks (at 8.5th and 12th second). nCDN line just waves at a low quantity. While in Figure 4b, nCDN line has a peak (at 6.5th second) while traditional CDN shows 3 peaks (7th, 8.5th and 12th second). NDN makes full use of stateful forwarding plane and adjusts the forwarding path adaptively and dynamically when a node or link is down. That is why NDN lines show smaller waves and cope with the failed node or links gradually (by marking some faces yellow or red). However, traditional CDN makes decisions by entry points and selects a new surrogate after a round-trip-delay. All the requests sent during the round-trip-delay get no contents back. Thus, the unsatisfied requests will be gathered and form peak points. During these times, all the requests distributed to the node or link will be influenced. The failed link or node will also be ferreted out, but it needs longer time and leads to more unsatisfied request.

[4]The dataset seems to be collected by Akamai according to the context

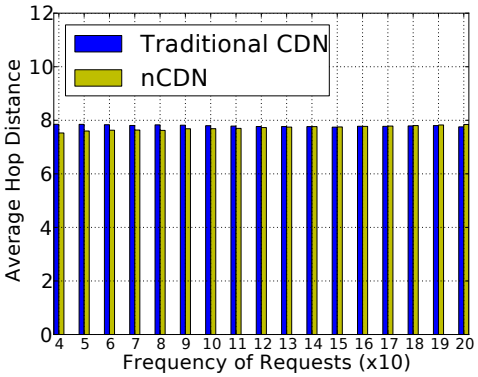


Figure 3: Average Hop Distance

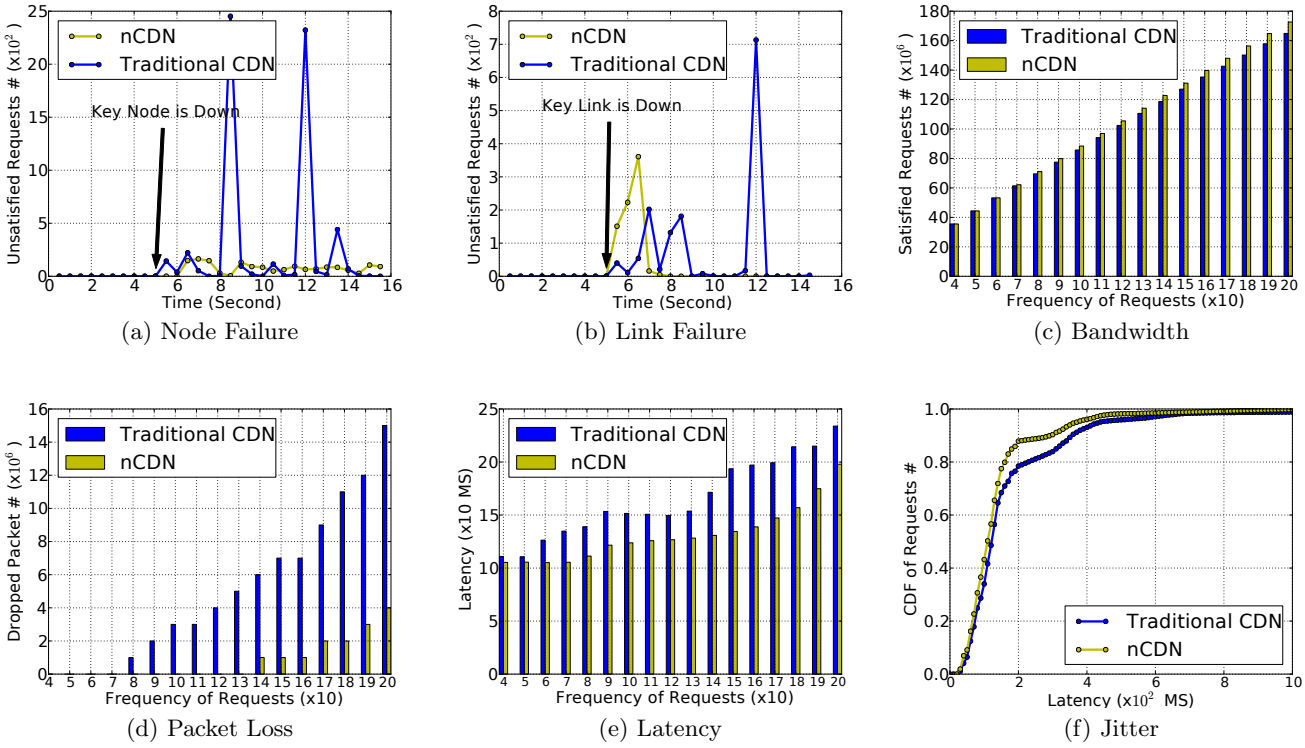
5.2 QoS

We explore QoS by checking bandwidth, packet loss, latency and jitter. As shown in Figure 4c, we measure the bandwidth by number of satisfied requests. When the traffic is not heavy, all the requests sent by consumers get Data back. As frequency of sending requests increases, the bandwidth of nCDN is always more than that of traditional CDN.

As to packet loss, nCDN shows more advantages. There is no packet loss until frequency reaches 140 per second and even after that, the number of lost packets is significantly less than that of traditional CDN, as shown in Figure 4d.

As to latency, average latency of nCDN is lower than that of traditional CDN, especially when the network traffic is as heavy as shown in Figure 4e. This is because NDN adaptively forwards requests to the most suitable links and surrogates.

All of the above three figures share a similar feature: when the network traffic is light, performances between nCDN and traditional CDN are very close.



(a) Node Failure (b) Link Failure (c) Bandwidth

(d) Packet Loss (e) Latency (f) Jitter

Figure 4: QoS and Reliability

However, when the network traffic is heavy, nCDN performs better. The fundamental reason lies in their forwarding path. nCDN and traditional CDN have the best forwarding choice when network traffic is light, since there is no congestion, nodes just pick the shortest path. However, when congestion happens, traditional CDN relies on entry points to adjust routing path, while nCDN can detect the congestion in time and adjust forwarding path accurately and dynamically. This analysis is supported by average hop distance shown in Figure 3

When the traffic is light, nCDN tries different paths to different nodes though some paths are not the shortest (but maybe of least latency or house no congestion) while traditional CDN tries from nearest to furthest. Thus, traditional CDN chooses the shortest paths, that is why the average hop distance of nCDN is slightly longer than that of traditional CDN under light traffic (but satisfy slightly more requests and less latency); when traffic is heavy, traditional CDN turns to longer paths to avoid congestion, but nCDN makes full use of shortest path whenever there is availability. This is why nCDN gets a shorter average hop distance when traffic is heavy.

Jitter is also an important factor of QoS, as we can find in Figure 4f (here frequency of requests is 150 per second and retransmission is enabled). 90% of the requests in our nCDN scenario had latency less than 200ms, while the ratio decreases to 80% in traditional CDN scenario. When we want to meet 96% of requests, in the nCDN scenario the latency to achieve this is less than 400ms, while the latency is increased to 500 ms in traditional CDN scenario.

The above four figures are drawn when there are 15 surrogates. For Figure 4f and Figure 3, the frequency of requests is 100 per second. We ran the simulation several times with different random number seeds and zipfs parameters. There are few cases under specified settings (including frequency of request, seed, number of content hosting surrogates), where IP performs better than NDN, but generally, all the simulation results show the same trend as the above figures.

5.3 Scalability

In this scenario, we measure the bottleneck throughput given some specified surrogates. The bottleneck throughput is measured with frequency of consumer sending requests. In order to measure the bottleneck, we set an upper bound for number of unsatisfied requests (no retransmission is allowed here). For some given set of surrogates, We increase the frequency of request by 10 every time. If the number of unsatisfied requests is less than the bottleneck, we increase the frequency until unsatisfied requests are more than the bottleneck, then we choose the previous frequency

to become the bottleneck frequency. As we add more surrogates into set of content hosting surrogates, then measure the bottleneck under the new conditions. As

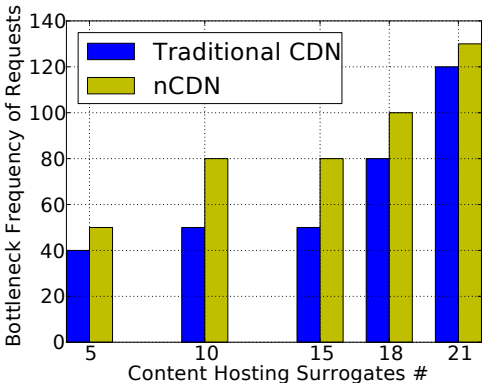


Figure 5: Scalability: Bottleneck throughput

we can find from Figure 5, where upper bound is 1,500 and simulation duration is 300 seconds (5 packets are dropped per second on average). For every specified set of content hosting surrogates, the bottleneck of nCDN is higher than that of traditional CDN. The bottleneck throughput is lower than linear growth with the number of surrogates, this is due to the limitation of other resources, such as link and queue capacity.We use a logarithm to fit the bottleneck throughput, as a result NDN gets a bigger coefficient , which means nCDN shows better scalability.

6. CONCLUSIONS

In this paper, we try to enhance CDN with NDN. Our new architecture can run over IP, Ethernet, etc, so it can be implemented on the current Internet. nCDN has the following advantages over traditional CDN:

1. nCDN architecture greatly simplified CDN design and implementation, since routing plane of NDN holds information of content distribution, and forwarding plane can detect the real-time network state. Thus, CDN no longer needs to keep mapping from "What" to "Where", or create complex modules on the application layer to monitor the network state.

2. nCDN architecture greatly improves CDN almost in all respects, including scalability, reliability and QoS, especially when network state is highly dynamic or traffic is heavy, it owes this to the stateful and adaptive forwarding plane of NDN.

3. nCDN architecture makes it easier to interconnect CDNs and realize different optimization solutions to achieve different goals, such as improving QoS or decreasing the cost.

We realize that CDN is a very complex system, techniques described in this paper are just the skeleton, and some of them need to be further explored. But we believe nCDN does simplify CDN design and improve its efficient. Our future work is to build a prototype of nCDN with CCNx[26], OSFPN [19], and open source or academic CDN, such as CoDeeN[27], a PlanetLab based academic CDN, and GLOBULE[4], an open source CDN. More scientific and engineering problems will appear then.

7. ACKNOWLEDGEMENT

Many Thanks to Alex Afanasyev from ndnSIM team, who helped us a lot on the experiment and implementation.

Supported by the National High-tech R&D Program ("863" Program) of China(No.2013AA010605), the National Science Foundation of China (No.61073172), and National Key Basic Research Program ("973" Program) of China (No.2009CB320501). Jun Bi is the corresponding author.

8. REFERENCES

[1] Al-Mukaddim Khan Pathan and Rajkumar Buyya. A taxonomy and survey of content delivery networks. *Grid Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report*, 2007.

[2] Limin Wang, Vivek Pai, and Larry Peterson. The effectiveness of request redirection on cdn robustness. *ACM SIGOPS Operating Systems Review*, 36(SI):345–360, 2002.

[3] Oscar Ardaiz, Felix Freitag, and Leandro Navarro. Improving the service time of web clients using server redirection. *ACM SIGMETRICS Performance Evaluation Review*, 29(2):39–44, 2001.

[4] Guillaume Pierre and Maarten Van Steen. Globule: a collaborative content delivery network. *Communications Magazine, IEEE*, 44(8):127–133, 2006.

[5] Markus Hofmann and Leland R Beaumont. *Content networking: architecture, protocols, and practice*. Morgan Kaufmann, 2005.

[6] Balachander Krishnamurthy, Craig Wills, and Yin Zhang. On the use and performance of content distribution networks. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pages 169–182. ACM, 2001.

[7] Erik Nygren, Ramesh K Sitaraman, and Jennifer Sun. The akamai network: a platform for high-performance internet applications. *ACM SIGOPS Operating Systems Review*, 44(3):2–19, 2010.

[8] John Dilley, Bruce Maggs, Jay Parikh, Harald Prokop, Ramesh Sitaraman, and Bill Weihl. Globally distributed content delivery. *Internet Computing, IEEE*, 6(5):50–58, 2002.

[9] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J.D. Thornton, D.K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos, et al. Named data networking (ndn) project. Technical report, Tech. report ndn-0001, PARC, 2010.

[10] V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, and R.L. Braynard. Networking named content. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*. ACM, 2009.

[11] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox. Information-centric networking: seeing the forest for the trees. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, page 1. ACM, 2011.

[12] Cheng Yi, Alexander Afanasyev, Lan Wang, Beichuan Zhang, and Lixia Zhang. Adaptive forwarding in named data networking. *ACM SIGCOMM Computer Communication Review*, 42(3):62–67, 2012.

[13] Y Cheng, A Afanasyev, I Moiseenko, B Zhang, L Wang, and L Zhang. Smart forwarding: A case for stateful data plane. Technical report, Technical Report NDN-0002, 2012.

[14] Cisco Visual Networking Index. Forecast and methodology, 2009–2014, 2010. `http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf`.

[15] Youtube blog, may 16, 2010. `http://youtube-global.blogspot.com/2010/05/at-five-years-two-billion-views-per-day.html`.

[16] Sen Wang, Jun Bi, Jianping Wu, Xu Yang, and Lingyuan Fan. On adapting http protocol to content centric networking. In *Proceedings of the 7th International Conference on Future Internet Technologies*, pages 1–6. ACM, 2012.

[17] Van Jacobson, Rebecca L Braynard, Tim Diebert, Priya Mahadevan, Marc Mosko, Nicholas H Briggs, Simon Barber, Michael F Plass, Ignacio Solis, Ersin Uzun, et al. Custodian-based information sharing. *Communications Magazine, IEEE*, 50(7):38–43, 2012.

[18] IETF CDNi Working Group. `http://tools.ietf.org/wg/cdni/`.

[19] Lan Wang, AKMM Hoque, Cheng Yi, Adam Alyyan, and Beichuan Zhang. Ospfn: An ospf based routing protocol for named data networking. *University of Memphis and University of Arizona, Tech. Rep*, 2012.

[20] Richard David Day. Meta content delivery network system, February 27 2007. US Patent 7,185,052.

[21] Varun Khare and Beichuan Zhang. Towards economically viable infrastructure-based overlay multicast networks. In *INFOCOM 2009, IEEE*, pages 1989–1997. IEEE, 2009.

[22] Varun Khare and Beichuan Zhang. Cdn request routing to reduce network access cost. In *Local Computer Networks (LCN), 2012 IEEE 37th Conference on*, pages 610–617. IEEE, 2012.

[23] Alexander Afanasyev, Ilya Moiseenko, and Lixia Zhang. ndnsim: Ndn simulator for ns-3. `http://irl.cs.ucla.edu/ndnSIM.html`.

[24] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring isp topologies with rocketfuel. *ACM SIGCOMM Computer Communication Review*, 32(4):133–145, 2002.

[25] Seyed Kaveh Fayazbakhsh, Yin Lin, Amin Tootoonchian, Ali Ghodsi, Teemu Koponen, Bruce M Maggs, KC Ng, Vyas Sekar, and Scott Shenker. Less pain, most of the gain: Incrementally deployable icn. 2013.

[26] CCNx Project. `http://www.ccnx.org`.

[27] CoDeeN Project. `http://codeen.cs.princeton.edu/`.