International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 3 | ISSN : 2456-3307

Memory Efficient and High Lookup PIT-Named Data Networking

A. Shanmuga Priya*, Anurudh Kumar, W. J. Kalesha, Dr. K. Suresh Joseph
Department of Computer Science, Pondicherry University, Puducherry, India

ABSTRACT

With an increase in growth of internet users, content plays the vital role and the IP address has less importance.Today's Internet architecture deals with IP address which leads to increases in addresses day-by-day which occupies more space.Server dependency is more at present with the current internet architecture.The issue of current internet architecture are memory requirement and server dependency which are overcome by Named Data Networking(Content-Centric Networking) using content name.NDN replaces the Host Centric Networking by Content-Centric Networking.This proposed framework deals with reducing memory and increasing lookup by clustering and encoding the component of a prefix in Pending Interest Table(PIT).

Keywords: Named Data Networking, Content Centric Networking,IP address,PIT.

I. INTRODUCTION

As the user increases with Internet usage day-by-day for communication, exchange of information such as text, video, audio in which they require content only not host address like IP address.The purpose of IP address is the identification of network interface and addressing the location.IPV4 is a protocol which uses 32-bit addressing in the total of 232 addresses and a new addressing system IPV6 uses 2128 addresses.Increases the address requires more space so It gave birth to NDN in 2009 by using a content name.NDN aims to simplify the application development process and the issues of today's internet architecture in terms of security, scalability, and sustainability which leads for the development, implementation, and understanding of NDN architecture face challenges like naming, signing, routing, and forwarding.NDN overcomes the issues of IP address and server need by retrieving the content from in-between routers.

When a request arrives at NDN router for particular content[5], it is sent to the functional data structures Content Store(CS), Pending Interest Table(PIT), Forwarding Information Base(FIB).The content store caches the previously requested content to fulfill a future request.If the requested content is not available in the content store, the name is checked in Pending Interest Table and if the name does not exist, then the requester's interface is added to the interface list to receive the content. Then the interest packet is passed to FIB and a new PIT entry is formed. The longest prefix of content name is found and interest packet is forwarded to the outgoing interfaces of FIB.

NDN has two main challenges.First, storing variable length names occupies more space compared to IP address .second, when a request arrives at the NDN router, then the PIT is made to lookup using an exact matching technique to check whether an entry for the content name already exists or not. A lookup incurs time proportional to the number of PIT

entries and the length of individual names.To solve these problems, an efficient data structures are needed for storing names in the PIT of NDN router.

II. ARCHITECTURE OF NDN

Named Data network is a Data-centric network which replaces Host centric network and driven by the user.NDN architecture is designed based on the basic principle of the Internet.The internet and NDN share the hourglass architecture with similar functions in Fig.1[11].NDN directly uses domain name services (DNS) and inter-domain routing protocols like BGP and OSGF.NDN uses the content name for the data packet which allows a user to track packets.The content name for data packets removes the need for application-specific middleware.Security and Strategy layers are added in NDN architecture.Security layer gives security to all the contents of the channel.Strategy layer gives the stateful forwarding plane and makes a forwarding decision for interest packet.The functions of the transport layer on the internet are inserted with forwarding plane in NDN.

Communication in ndn motivated by the user and achieved by two packets such as Interest packet and Data packet. Interest packet has the name of desired data and routers use this name to forward the Interest toward the data. Data packet encompasses both the name and the content, together with a signature by the provider's key which binds the two.

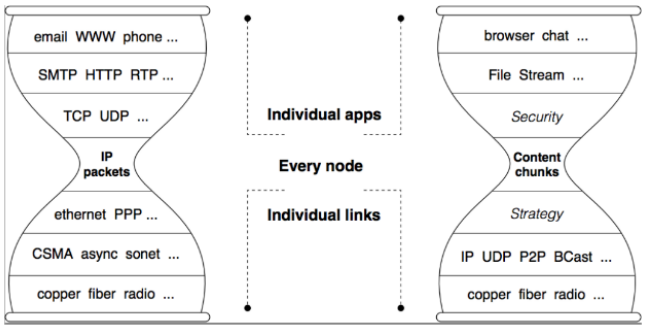

Figure 1. Internet and NDN hourglass architectures

III. NDN DATA STRUCTURES

Routing in NDN maintains three data structures Like Content Store (CS), Pending Interest Table (PIT), Forwarding Information Base (FIB).

A. Content Store (CS)

Content store cache the copy of data packet passing through the router until replaced by new content. Increasing the sharing probability, bandwidth is saved and reducing retrieval time of content through the cached data. Searching content packet done through exact matching that is character by character match of a name.

B. Pending Interest Table (PIT)

Pending Interest Table saves the entry for incoming interest packet interfaces until data packet reached or its entry lifetime terminates. These entries are used to forward data packets to a user. Searching for PIT entries is done over exact matching that is character by character match of names.

C. Forwarding Information Base (FIB)

Forwarding Information Base (FIB) saves the next hops and information for reachable destination and used for forwarding interest packet with name prefix. Searching for FIB entries are done through Longest Prefix Match.

IV. NDN FORWARDING PROCESS

Although a conclusion may review the main points of the paper, do not replicate the abstract as the conclusion. A conclusion might elaborate on the importance of the work or suggest applications and extensions. Authors are strongly encouraged not to call out multiple figures or tables in the conclusion these should be referenced in the body of the paper.

The NDN forwarding process is shown in the Fig.2[11].User sents interest packet to NDN router, the content name is searched in the content store. If the content is found, the NDN router forwards the

contents to the user through a data packet.Else, the CN is looked up in the PIT. If a matching PIT entry is determined, interest packet's incoming interface will be embedded into the list of the interface. So, when the corresponding data packet is available, all interested users will accept a copy of that data packet.If PIT entry is not found for an interest packet, the interest packet is passed to the router's Forwarding Information Base which performs a longest prefix match.Then PIT entry is created for that interest packet.NDN router searches for the PIT entry matching at the arrival of the data packet and forwards data packet to interface in PIT entry.

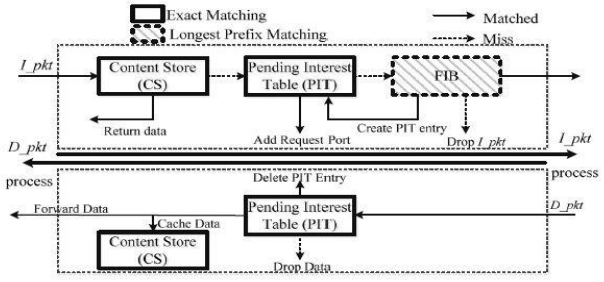

Figure 2. NDN Forwarding process

NDN Forwarding can be classified as scalable forwarding and forwarding strategies. Scalable forwarding with a PIT is stateful forwarding in which routers keep the state of forwarded packets which allows loop detection and flow balance. It deals with memory and time requirement while forwarding packets. Forwarding strategies make all the decision which can be used for forwarding interest and data packets and its plane is control plane.

V. RELATED WORK

Radient is named encoding method for PIT for efficient memory which reduces memory consumption[3].DPEL calculate PIT entry lifetime for each interest using interest satisfaction rate(ISR) and hop count.It requires less PIT size[9].MaPIT is mapping bloom filter which is modified data structure of bloom filter. It minimizes memory consumption and has less execution time[4]. Name lookup engine with Adaptive Prefix Bloom Filter (NLAPB), NDN name or prefix is divided into B-prefix and T-suffix processed by bloom filter and small-scale trie respectively. It increases lookup time, lower false positive rate, decreases memory consumption and reduced processing time[1]. Name label swapping uses fixed-length label swapping replacing unbounded name lookup at core nodes and caches data only at edge nodes.It increases name lookup[2]. Encoded Name Prefix Trie (ENPT) in Name Component Encoding (NCE) scheme encodes the components of the content name.Reduce PIT size.It reduces the access time[6].NRS framework with MDHT and Hskip which the forwarding matching the network topology, exploit request locality, supports domain-specific coping of binding entries, and offers constant hop count.It reduces maintenance overhead and memory requirement[7]. Scalable forwarding is done through secure namespacing by Map and Encap method where mapping solves the scalability issues.It provides Security and reduces memory requirement[8].Scalability is achieved through bloom filter by encoding prefixes in FIB.It reduces memory requirement[10].

Trie is also called as radix tree or prefix tree, a type of search tree which is used to store dynamic set or associative array where the keys are strings.Radix trie or compact prefix tree is space optimized trie in which parent node with a single child is merged.The operations of radix trie take O(k) where k is the maximum length of all strings in the set, where length is the quantity of bits.Compressing tries compress the trie and merge the common branches which give large performance gains.Ctrie is a concurrent hash trie, also called as hash array mapped trie.The concurrent insert and delete operations can be performed and it takes O(1).HAT trie is a type of radix trie which collects individual key-value pair under radix nodes and hash buckets into an associative array.Judy array is a type of associative array with increased performance and less memory requirement and remains efficient even on structures with sizes in the peta-element range with

O(log256n).PATRICA trie, Practical Algorithm To Retrieve Information Coded in Alphanumeric. There is no explicit null link.To search, it uses bit index if a bit is 1 search at right and if a bit is 0 search at left kit it takes O(M) where M is the string[12].

VI. PROPOSED SYSTEM

A. Scalable-PIT Framework

The scalable PIT is memory efficient encoding framework for prefixes in Fig.3.It includes decomposition, Name Clustering, Token allocation, PATRICA trie.

1) Decomposition: Given request is decomposed or split into components based on the delimiter (/).

2) Name Clustering: The components organized in nodes and levels.Then the components are grouped in each level.

3) Token Allocation: The components in each level are encoded with the token. If the same component is present the same level, then the component is merged and re-assign the unique token based on previously assigned token which will efficiently reduce the memory.

4) PATRICA Trie: A compact trie in which any node that has only one child is merged with its parent.

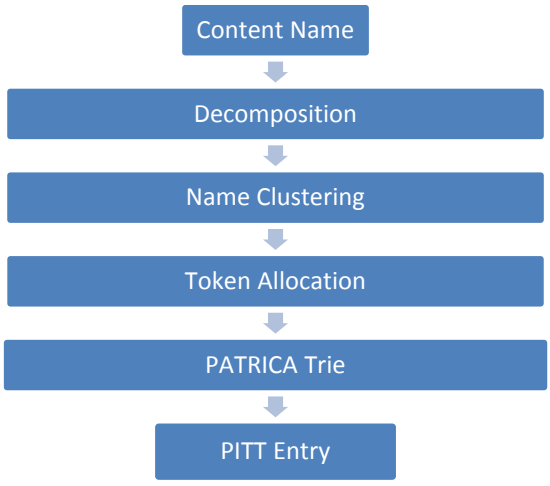

Figure 3. Scalable-PIT Framework

When a request arrives at NDN router, checks whether the requested content is present in the CS or not using the content name.If it is present in CS, it will return the content to a user.If it is not present, it will go to the PIT. Here, the request is decomposed into components based on the delimiter (/).Then the token is assigned to each component based on both the level it is present and other components present in that level Fig.4.If the same component is present the same level, then the component is merged and re-assign the unique token based on previously assigned token. For example, In Level2 the components starting from node 2 and node 9 are yahoo, google and google, Sina, Baidu, respectively.They are encoded as <yahoo, 1>, <google, 2>and <baidu, 1>, <sina, 2>, <google, 3>. Here,the same components are present in the same level, then merge that components<google,2>,<google,3> and produce <google,4>. If the component is found in PATRICA trie, then the token is fetched for that component. Else if it is not found in a trie, then the token will assign to the new component-based previously assigned tokens and inserted to the trie.

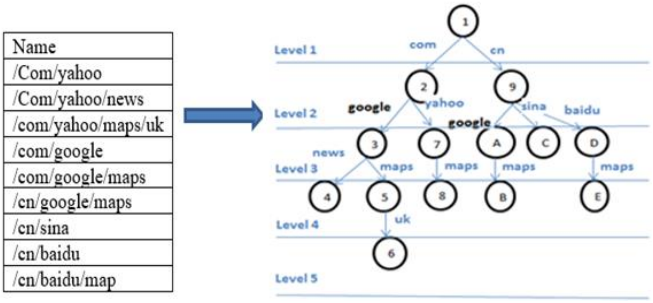

Figure 4 . Level Structure for Token allocation.

B.Algorithm

The algorithm for inserting component into trie and deleting a component from trie using Scalable N-PIT is given in fig.5.When a request reaches for content, first it decomposes the prefixes into components using the delimiter (/) and cluster the components each level and assign the token based on the level and the other component present in the same level using decompose (prefix).The prefix key and its value are assigned to component length. If the component length is equal to both node with value and key and

k depth of trie. Then node should be empty so that new component can be inserted. If the node with value is equal to the component of a prefix.Then merge the component with its parent and remove the children for that name prefix.To delete a component from trie, decompose the prefix into components using the delimiter(/) and cluster the components each level and assign the token based on the level and the other component present in the same level using decompose(prefix).Then check requested content name which is to be deleted and node prefix are same.Both are same, the invoke delete(Cni).If time is over then drop the data packet otherwise sent the data packet to incoming interfaces.

Algorithm 1:Insert component into trie

1. Decompose(prefix)
2. C_l=prefix(k,v)
3. If (C_l==$node_{vk}$.length .&& C_l==k.length)
4. If ($node_{vk}$==null)
5. $node_{vk}$=v
6. $node_{vk}$=true
7. End
8. if($node_v$==C_k)then
9. Merge(P,remove(Cp))

//merge prefix to the node & remove children for that name prefix.

10. End
11. End

Figure 5.a. Insertion algorithm for Scalable-PIT Framework

Algorithm 2: Delete component from trie

1. decompose(prefix)
2. If (Cn=node prefix)
3. K_{cn}=K_{cn}-1
4. If K_{cn}==0
5. delete(Cn_i)
6. Cn=Cn U K_{cn}
7. End
8. If time is zero
9. Drop D_{pkt}
10. Else
11. Sent D_{pkt} to incoming interface.
12. End
13. End

Figure 5.b. Deletion algorithm for Scalable-PIT Framework

C.Complexity Analysis

Assume, N name requests reach an NDN router. Each name has M components. Here, Memory complexity and Time complexity for storing a set of names and their encoding forms are discussed.

1)Time Complexity

Time complexity can be achieved by clustering the same components and by reducing the depth of the trie Therefore, the lookup complexity can be O (M/k).The operation in PATRICA trie occurs at leaf node with updation time complexity of O (M/k).

2)Memory Complexity

Memory complexity defines the total amount storage space required to store all the N names.The space occupied by O (N) by compressing single-child nodes together in PATRICA trie.

D. Performance metrics

The total amount of memory consumed and lookup time consumed can be calculated by the following matrics.Memory Used is the total memory used to store names and its encoded names in the PIT in MB.

Frequency is the number of names updated per second in million per second.Interest Satisfaction Rate is the difference between request generated and data received.

VII. CONCLUSION

In this paper, Memory efficient key allocation method is proposed, called Scalable PIT framework, which reduces the storage needed by the PIT data structure in NDN router and improved lookup time. The proposed system facilitates the efficient lookup and reduces the memory requirement using PATRICA trie. It can handle names with more than seven or eight components.

VII. REFERENCES

[1]. Quan,W.,Xu,C.,Guan,J.,Zhang,H.,& Grieco,L. A. 2014. Scalable Name Lookup with Adaptive Prefix Bloom Filter for Named Data Networking,18(1),102–105.

[2]. Luo,J.,Wu,C.,Jiang,Y.,& Tong,J. 2014. Name Label Switching Paradigm in Named Data Networking,7798(c),1–4. https://doi.org/10.1109/LCOMM.2014.2387344

[3]. Saxena,D.,& Raychoudhury,V.2016. Radient : Scalable ,memory efficient name lookup algorithm for named data networking,Journal of Network and Computer Applications,1–13. https://doi.org/10.1016/j.jnca.2015.12.009

[4]. Li,Z.,Liu,K.,Zhao,Y.,& Ma,Y. 2014. MaPIT: An enhanced pending interest table for ndn with mapping bloom filter. IEEE Communications Letters,18(11),1915–1918. https://doi.org/10.1109/LCOMM.2014.2359191

[5]. Saxena,D.,Raychoudhury,V.,Suri,N.,& Becker,C. 2016. ScienceDirect Named Data Networking : A survey,ScienceDirect ,19,15–55.

[6]. Wang,Y.,He,K.,Dai,H.,Meng,W.,Jiang,J.,Liu,B., & Chen,Y. 2012. Scalable name lookup in NDN using effective name component encoding. Proceedings - International Conference on Distributed Computing Systems,(20100002110051),688–697. https://doi.org/10.1109/ICDCS.2012.35

[7]. Dannewitz,C.,D'Ambrosio,& Vercellone,V. 2013. Hierarchical DHT-based name resolution for information-centric networks,Computer Communications,36(7),736–749. https://doi.org/10.1016/j.comcom.2013.01.014

[8]. Afanasyev,A.,Yi,C.,Wang,L.,Zhang,B.,& Zhang,L. 2015. SNAMP: Secure namespace mapping to scale NDN forwarding. Proceedings - IEEE INFOCOM,281–286. https://doi.org/10.1109/INFCOMW.2015.7179398

[9]. Bouk,S. H.,Ahmed,S. H.,Yaqub,M. A.,Kim,D.,& Gerla,M. 2016. DPEL: Dynamic PIT entry lifetime in Vehicular named data networks.IEEE CommunicationsLetters,20(2),336–339. https://doi.org/10.1109/LCOMM.2015.2508798

[10]. Alzahrani,B. A.,Reed,M. J.,Riihijärvi,J.,& Vassilakis,V. G. 2015. Scalability of information centric networking using mediated topology management. Journal of Network and Computer Applications,50,126–133. https://doi.org/10.1016/j.jnca.2014.07.002.

[11]. https://named-data.net/project/archoverview/

[12]. https://wiki2.org/en/Trie