International Core Journal of Engineering Volume 8 Issue 7, 2022
ISSN: 2414-1895 DOI: 10.6919/ICJE.202207_8(7).0013

A Neural Bloom Filter-based Two-layer Name Lookup Structure for Information-Centric Networks

Junru Shi*, Qingtao Wu

School of Information Engineering, Henan University of Science and Technology, Luoyang 471023, China

Abstract

Information center network (ICN) solved the mass efficient transmission of information problems in the traditional network, it uses the information of the name instead of IP addresses as identification of network transmission, it implements the name information in the network layer analysis, information data in the network routing node cache, multicast transmission of information, and other functions, based on this advantage, This paper designed a kind of information based on neural bloom filter name lookup structure, the structure consists of two layers: nerve bloom filter and backup bloom filter, and proposes two algorithms: the name lookup algorithm and content name insert algorithm, using memory neural network structure as a learning model, in order to reduce memory footprint and the rate of false positives. We analyze the false positive rate generated by the name lookup structure, and construct a sigmoid function to compress the memory space instead of multiple hash functions. The proposed algorithm can be used in name lookup.

Keywords

Information-Centric Network; Neural Bloom Filter; Information Name; Lookup Structure.

1. Introduction

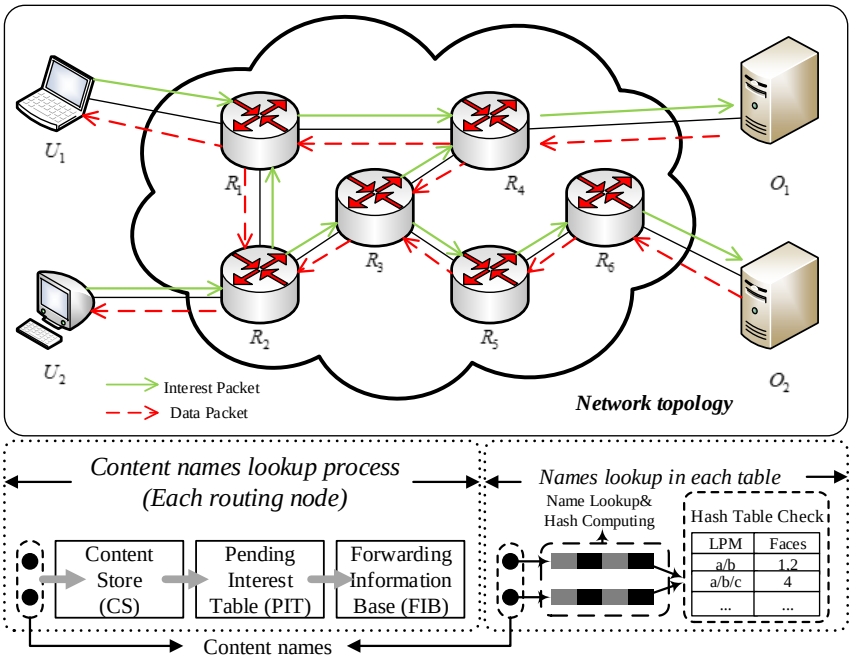

Fig. 1 The overview of content name lookup process in ICN

ISSN: 2414-1895 DOI: 10.6919/ICJE.202207_8(7).0013

In order to solve the problem of efficient transmission of mass information in traditional networks, a new network Information-Centric Network (ICN) is proposed [1]. Unlike traditional IP networks, ICN uses the name of the information instead of the IP address as the identifier for network transmission. This novel network architecture has functions such as enable the resolution of information names at the network layer, cache the information data at routing nodes in the network, and the multicast transmission of information [2]. Therefore, compared with traditional network, users can obtain information more quickly and safely.

In ICN, fast and fast and accurate routing and forwarding can improve the efficiency of information transmission in order to respond quickly to user requested information [3-5]. When a user requests a content information, an interest packet is sent that contains the name of the requested information. If we want to achieve accurate routing of information names, the lookup for information names is very important. Fig.1 shows the entire process of information names lookup at the routing node. At each node, the information name is searched through the Content Store (CS), Pending Interest Table (PIT) and Forwarding Information Base (FIB), and the information name is matched by the longest prefix matching (LPM) [6-8] algorithm. However, due to the diversity of content naming methods, different from IP addresses, the length of information names is not fixed, and the number is huge, how to conduct efficient information name lookup has become a challenge.

Table 1. Summary of notations and symbols

Notation	Representation of the symbol
K	Set of prefixes entries stored in memory
Q	Set of requesting names
K^{train}	Distribution over sets to store
Q^{train}	Distribution over queries
D	Set of data structure
M	Memory matrix
A	A learnable address matrix
$o\ \cdot$	A soft max function
$\mathbb{M}(\cdot)$	Memory space
$F\ .$	False positive rate
F_n	False negative rate
$G\ \cdot$	Learned hash function
τ	Threshold

Recently, researchers proposed some different methods to improve the efficiency and accuracy of name lookup. [9] proposed using the Bloom filter as a method of name lookup, which can reduce the false positive rate and the search time. However, the standard Bloom filter cannot delete elements, and the memory footprint increases as the content grows, therefore, [10] proposed to use the counting Bloom filter (CBF), which can dynamically delete the content and reduce the memory footprint. Ref. [11] proposed a Tree-based name lookup method, compared with the Bloom filter it has higher

ISSN: 2414-1895 DOI: 10.6919/ICJE.202207_8(7).0013

accuracy and lower memory space. But, none of these methods solves the problem of conflicts between elements very well, [12] proposed a novel learned index architecture, which can improve the search accuracy by building a lookup model based on machine learning. Due to the large number of content names, compressing memory footprint is very important. [13] proposed a meta-learning neural Bloom filter structure, it can better compress the memory footprint and reduce the false positive rate by using meta-learning.

In this paper, we use a neural Bloom filter combined with a standard Bloom filter to construct a two-layer information name lookup structure in ICN [14]. We use the neural Bloom filter in the first layer to classify and lookup the requested information name, and then use the second layer backup Bloom filter to increase the accuracy of the search [15-18]. The name is addressed by the read operation in the memory augmented neural network and inserted by the distributed write operation. Among them, LSTM is used as the method of learning content collection, and combine meta-learning to select different subsets of the same distribution to train the content. [19] Meta-learning allows us to take advantage of the same redundancy. Therefore, using this method can greatly compress the memory space. However, since the use of neural networks for searching will generate a certain number of false negatives [20], we use a backup Bloom filter to eliminate false negatives and improve the accuracy of the lookup. Our contributions are as follows:

1) We design an information name lookup structure based on the neural Bloom filter, which contains two layers: neural Bloom filter and backup Bloom filter.

2) We use memory-augmented neural network structure as the learned model and construct a sigmoid function to replace multiple hash functions to compress memory space.

3) We analyze the false positive rate generated by the name lookup structure.

4) We propose two algorithms, content name lookup algorithm and content name insertion algorithm, which can reduce the memory footprint and false positive rate.

We formally represent the problems that need to be solved in this paper and propose our model in Section II. In section III, we design the content name lookup algorithm and the name insertion algorithm. We validate our structure via several evaluations in Section IV. Finally in Section V, we give the conclusion about this paper.

2. Problem Statement and Our Model

2.1 Problem Statement

In ICN, the basic method for content name lookup is to use Bloom filter [21]. The Bloom filter is a binary data structure that is often used as a solution to the approximate of elements within a set. It can dynamically insert and search elements and has a very low memory space. The search and insert operation using the Bloom filter is to map the content name to the corresponding bit through multiple hash functions, setting the value of the corresponding bit to 1. For example, give a name set K and a query set Q, inserts members of the set S into the Bloom filter through k hash functions, $b\left[h_i\left(x\right)\right] \leftarrow 1$, where $i = 1,2,...,k$. For a given query content $q \in Q$, find the value of the corresponding bit via hash function in the Bloom filter, if $b\left[h_1\left(q\right)\right] \wedge b\left[h_2\left(q\right)\right] \wedge b\left[h_3\left(q\right)\right] \wedge ... \wedge b\left[h_k\left(q\right)\right] = 1$ this means the content is searched in the Bloom filter. If at least one of the values on the corresponding bit is not equal to 1, the content is not found. However, using the Bloom filter for name lookups produces a certain number of false positives. According to [22], the false positive rate is:

ISSN: 2414-1895 DOI: 10.6919/ICJE.202207_8(7).0013

$$f_p = \left(1 - e^{-\frac{nk}{m}}\right), \tag{1}$$

and given a $\epsilon > 0$, a maximum of false positive of the fraction ϵ is allowed during the lookup process. So we can get a bound of false positive rate, when $f \le \epsilon$ requires:

$$m = n \log_2 e \cdot \log_2 \left(1/\epsilon\right), \tag{2}$$

and the space is within a factor of $\log_2 e \approx 1.44$ of the lower bound.

Therefore, our lookup problem can be translated into how to minimize the false positive rate of the lookup and have a lower memory footprint during the name lookup process. Table I shows the main notations in this paper, for any $q_i \in Q$ our problem is formalized as:

$$\begin{aligned} &\min_{Q,K} \mathbb{M}\left(Q,K\right) \le \gamma \\ &\min F\left(Q\right) \le \delta \end{aligned}, \tag{3}$$

where γ is the memory space occupied by the lookup process, and δ is the false positive rate generated by the lookup.

2.2 Our Model

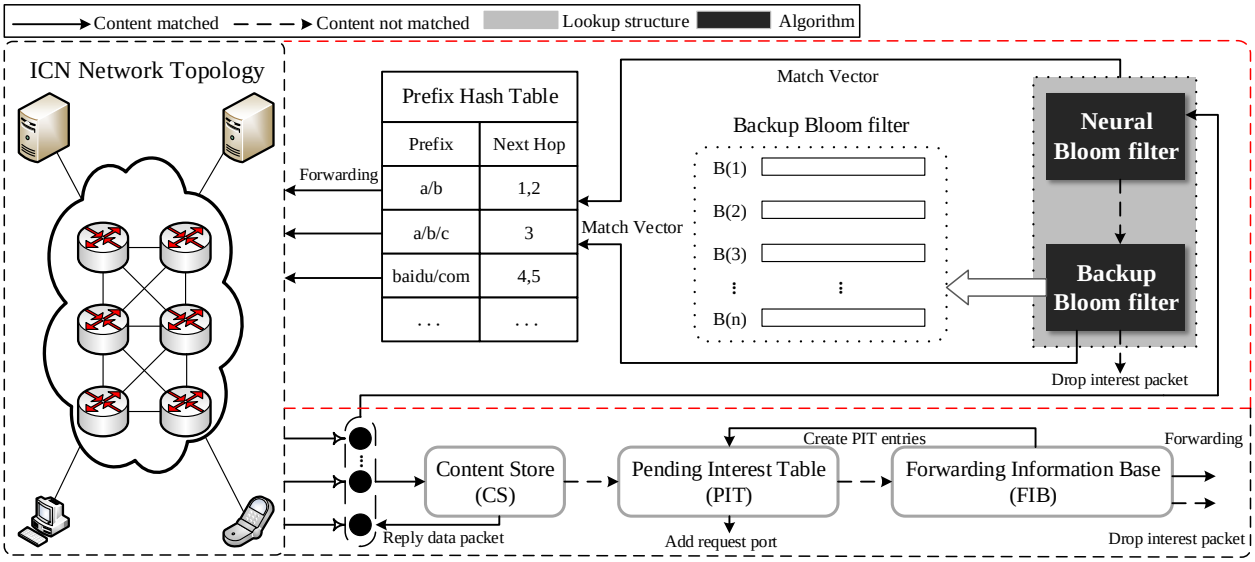

Fig. 2 Framework for name lookups using the learned Bloom filter.

In this section, we propose a novel structure for ICN content name lookup to compress memory and improve the accuracy of search. In order to improve space efficient solution, [23] presents neural Bloom filter, using meta-learning and memory augmented neural networks to compress memory footprint. Our lookup structure mainly includes two layers based on neural Bloom filter and standard Bloom filter.

Our lookup structure is illustrated in Fig.2. When the user requests an interest packet, the content of the request is first looked up in the CS table. If there is a matching content name in the CS, then the

ISSN: 2414-1895 DOI: 10.6919/ICJE.202207_8(7).0013

data packet is returned to the user. If there is no matching content, it is looked up in the PIT table. If there is no matching content name in the PIT, finally the content will lookup in the FIB. When the matching content is found in the FIB, the interest packet is forwarded and the corresponding content name is created in the PIT table, and if the requested content is not found, the interest packet is finally discarded. At each routing node, we lookup through the constructed model. The content name is first searched by the neural Bloom filter. If the corresponding content name is found in the neural Bloom filter, the longest prefix matching algorithm is used to match the corresponding name prefix, and forward the content by the next hop in the corresponding prefix hash table. If the corresponding content is not found in the neural Bloom filter, the content name is passed to the backup Bloom filter for secondary lookup, and then the found content is matched with the longest prefix and sent from the corresponding next hop.

In ICN, we treat x as the content name in the request interest packet. [24] proposed a memory architecture named neural Bloom filter, it combines memory-augmented neural networks and meta-learning to construct the data structure. We denote a set of elements $K = \left\{x_1, x_2, x_3, ..., x_n\right\}$, and a set of names requested $Q = \left\{q_1, q_2, q_3, ..., q_u\right\}$, now we denote K^{train} as the distribution of the storage sets and Q^{train} as the distribution over queries. Then we use meta-learning for training, first we get sample set K from K^{train} to store and sample $Q_t = \left\{q_1, q_2, ..., q_t\right\}$ from Q^{train}, for any y that satisfies distribution $D = \left\{\left(q_j, y_j = 1\right) \mid q_j \in K\right\} \cup \left\{\left(q_j, y_j = 0\right) \mid q_j \notin K\right\}$ with $j = 1, 2, ..., t$. Our names lookup process can be seen as a content-based addressing mechanism for meta-learning, implemented using the read operation. According to [24], we read from memory use the read vector r_j which is a sum of memory records:

$$r_j = \sum_{j=1}^{t} soft\max\left(\frac{f\left(x\right) \cdot M\left(j\right)}{\left\|f\left(x\right)\right\| \cdot \left\|M\left(j\right)\right\|}\right), \tag{4}$$

where $M\left(j\right)$ is the $j-th$ row in the memory matrix M, $f\left(x\right)$ is a function of the input x by the controller.

We write the contents of the storage set K to the memory matrix M using the sigmoid function:

$$f_{write}\left(x\right) = \frac{1}{1+e^{-x}}, \tag{5}$$

then we calculate the logistic with queries:

$$g_j = f_\theta^r\left(M, q_j\right), \tag{6}$$

and we get the loss L is:

$$L = \sum_{j=1}^{t} y_j \log g_j + \left(1 - y_j\right) \log\left(1 - g_j\right). \tag{7}$$

ISSN: 2414-1895 DOI: 10.6919/ICJE.202207_8(7).0013

The neural Bloom filter structure shows in Fig.3. This architecture can look up and insert content names at the same times. As shown in Fig.3, determining whether a content name is to be searched or inserted is determined by classification through a controller. At each routing node, the requested content is sorted through a soft max. This step is equivalent to inserting or addressing the content through the hash function in the Bloom filter. For any names $x_i \in K$, the content name first use LSTM as a learned model to learning set membership and encodes the input to z, then we use the MLP model to transform the content z to look up and insert. Since in the memory-augmented neural network, read operations and write operations can be performed simultaneously, we can find the corresponding position a of the content in the memory by computing the searched content q and a learnable address matrix A via soft max:

$$a = o(qT A), \tag{8}$$

utilizing address a, if the contents can be found in the cache, the address a is component-wise multiplying with the memory matrix M to read the content, if the content is not found in the cache, a write operation is performed:

$$Mt+1 = Mt + waT, \tag{9}$$

insert the content into the matrix M and weighted it by the address a, where Mt is the memory matrix at time t.

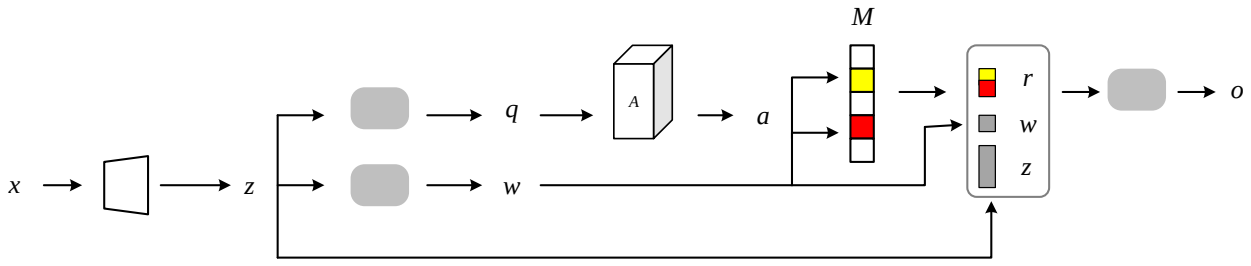

Fig. 3 Architecture of neural Bloom filter.

Due to the use of a neural Bloom filter, a certain number of false negatives will be produced. So we use a backup Bloom filter as the second layer of the lookup structure to reduce the number of false negatives to 0. As shown in [25], using a threshold τ to determine whether a content name x is in the set K. We set a threshold τ, if $g\ x\ < \tau$ but $x \in K$, this means that content lookup produces false negatives in the first layer structure, so pass the content to the second layer of the backup Bloom filter for a second lookup. We define a function:

$$G\ x\ = \left\lfloor mg\ x\ \right\rfloor \tag{10}$$

where $g\ x$ is a sigmoid function. We use $G\ x$ as a hash function to replace multiple hash functions to map the content name to a $m-$bit array in the backup Bloom filter. This method is used to reduce the conflict between elements and compress the memory footprint.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202207_8(7).0013

Since the structure uses neural network and meta-learning, learning with a small number of samples reduces the memory usage to a certain extent, and the neural network can improve the accuracy of the lookup. In the following sections, we will analyze the accuracy of the search and demonstrate the performance of our architecture in ICN.

2.3 Analyze the False Positive Rate

The accuracy of search is an important index of content names lookup. In the process of the content routing, it is very important to accurately lookup and forward the content name requested by the user. In this section, we will analyze the false positive rate generated by our proposed architecture.

Our name lookup architecture is divided into two layers, and the false positive rate generated by the architecture consists of two parts: neural Bloom filter and backup Bloom filter. In the neural Bloom filter, the content name insertion using the write operation in the memory-augmented neural network can be regarded as the insertion operation by multiple hash functions in the standard Bloom filter. Therefore, during the lookup process, the name content addresses found will generate conflicts and false positives. Unlike the Bloom filter, using the neural Bloom filter for name lookup also produces a certain number of false negatives, so we will set the backup Bloom filter size equal to the number of false negatives. For the false positive rate generated by the architecture, we give the following definition.

Definition 1: Give an ICN name set K, for any name $x_i \in K$ and $g(x_i) \geq \tau$, a false positive rate FN is generated from the neural Bloom filter, and a false positive rate FB is generated form the backup Bloom filter:

$$FQ = FN + FB. \tag{11}$$

When the content name satisfies $g(x) < \tau$ but $x \in K$, this means that the content name has a false negative through the lookup of the neural Bloom filter, and we define the false negative rate as F_n. The above definition can be expressed as:

$$FQ = P(g(x) \geq \tau) + P(1 - g(x) < \tau) FB. \tag{12}$$

3. Content Name Lookup Algorithm Design

In this section, we design the name lookup algorithm and the name insertion algorithm to detail the content name lookup process in ICN.

3.1 Content Name Lookup Algorithm

In the ICN, the user requests an interest packet, the interest packet contains the requested content name, and the name lookup is performed through the two-layer lookup structure at each routing node. Algorithm 1 shows the operation of the entire lookup process.

When a content name is waiting to be looked up, as shown in Algorithm 1, the name is first searched through the neural Bloom filter, the content name x is embedded into the neural network through LSTM, and then the content name is transformed to a search q via the MLP model for classification. We set a threshold τ if the name x satisfies the condition $x \in K$ and $g(x) \geq \tau$, this means that the content name can be searched in memory. In order to get the position of the name in memory, we calculate the query q and a learnable address matrix A by a soft max function. Then we read the location information searched in the memory, and output the content name to the longest prefix match

ISSN: 2414-1895 DOI: 10.6919/ICJE.202207_8(7).0013

in the hash table, and finally send the matched content name from the corresponding next hop. However, due to use the neural Bloom filter for lookup, a certain number of false negatives are generated. So when x satisfies $x \in K$ but $g\ x\ < \tau$, we use the backup Bloom filter for another further lookups to improve the accuracy of the name lookup. The standard Bloom filter uses multiple hash functions to reduce conflicts between elements. To compress memory and reduce conflicts between elements, we use a learned function instead of multiple hash functions. The content name is searched by $G\ x$ in the backup Bloom filter to lookup whether the corresponding bit is 1 or not. If the corresponding bit is all 1, then match the longest prefix in the hash table and send the matched content from the next hop. If at least one bit of the corresponding bit is not 1, the interest packet is discarded.

Algorithm 1 The lookup of an entry in the architecture

Input: Element set K, Memory matrix M, Learnable address matrix A, Threshold τ
Output: Element x
1: Invoke LSTM construction to input an embedding ***z***;
2: Invoke MLP construction to get a query word ***q***;
3: **for** $x, y\ \in D$ **do**
Calculate the memory address ***a*** according to (8);
end for
if $x \in K$ and $g\ x\ \geq \tau$ **then**
Read the query address ***r*** $\leftarrow flatten\ M \odot a\ ;$
Output x and match the longest prefix in Hash Table;
Send x to next hop;
else if $x \in K$ but $g\ x\ < \tau$ **then**
Search x in backup Bloom filter;
Calculate $i \leftarrow G\ x\ ;$
while Search $b\left[i\right] = 1$ in backup Bloom filter **do**
Output x and match the longest prefix in Hash Table;
Send x to next hop;
end while
else
Drop the element x;
end if

3.2 Content Name Insertion Algorithm

The purpose of setting up the backup Bloom filter is to reduce the false negative generated by the neural Bloom filter to 0, so the size of the backup Bloom filter is the number of false negatives. The memory-augmented neural network can be used to locate or separately input to improve network performance. Therefore, when the content name $x \notin K$, it means that the lookup content name is not in the content set. As shown in Algorithm 2, we cache the content name x into the memory by the write operation in the memory-augmented neural network, and record its position into the memory

ISSN: 2414-1895 DOI: 10.6919/ICJE.202207_8(7).0013

matrix M, if the same content name x is requested at the next moment, then it can be searched in this routing node.

When the name x satisfies the condition $x \in K$ but $g\ x\ < \tau$, insert the name into the backup Bloom filter using the learned function $G\ x$ as the hash function, where $g\ x$ is a sigmoid function and $g\ x\ \in \left[0,1\right]$, m is the size of the backup Bloom filter, which is equal to nF_n. The names get corresponding position through hash function, then change the value of bit for the corresponding position to 1 in the backup Bloom filter, if different content is inserted into the same location, then the value of bit is still 1 and does not change. In this way, the names in the content set will be mapped via $G\ x$ to higher bit position. Our proposed algorithm reduces the false positive rate and the memory space at the same time. We will prove the performance of our algorithm in the experimental section.

Algorithm 2 The insertion of an entry in the backup Bloom filter

Input: Element set K, Size m, Element x, Threshold τ

1: Invoke LSTM construction to input an embedding ***z***

2: Invoke MLP construction to get a write word ***w***

3: **for** $x \notin K$ **do**

4: Calculate the memory address ***a*** according to (8);

5: Insert x in memory matrix M;

6: Update memory matrix M according to (9);

7: **end for**

8: **if** $x \in K$ and $g\ x\ < \tau$ **then**

9: Calculate $G\ x$ according to $G\ x\ = \left\lfloor mg\ x\ \right\rfloor$;

10: Position in backup Bloom filter: $i \leftarrow G\ x$;

11: Set bits in backup Bloom filter: $b\left[i\right] \leftarrow 1$;

12: **else**

13: Set bits in backup Bloom filter: $b\left[i\right] \leftarrow 0$;

14: **else if**

4. Conclusion

In this paper, we design a kind of information based on neural bloom filter name lookup structure. In addition, we propose two algorithms: the name lookup algorithm and content name insert algorithm, using memory neural network structure as a learning model to reduce memory footprint and the rate of false positives. In the future, we will make more studies for the information center network name search method.

Acknowledgments

This work was supported by the Scientific and Technological Innovation Team of Colleges and Universities in Henan Province under Grants No. 20IRTSTHN018.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202207_8(7).0013

References

[1] G. O. Young, J. Peters. Synthetic structure of industrial plastics, in Plastics. Vol. 3 (1964), pp. 15-64.

[2] W.-K. Chen, Linear Networks and Systems. World Scientific. Vol. 3 (1990). pp. 123-135.

[3] J. U. Duncombe. Infrared navigation Part I: An assessment of feasibility, IEEE Trans. Electron Devices. Vol 11 (1959), pp. 34-39.

[4] E. P. Wigner, Theory of traveling-wave optical laser, Phys. Rev., vol. 134 (1965), pp. 635-646.

[5] H. Kurss, Kahn. W. A note on reflector arrays. IEEE Transactions on Antennas and Propagation. Vol. 15 (1967). pp. 692-693.

[6] E. E. Reber, R. L. Michell, and C. J. Carter, Oxygen absorption in the earth's atmosphere, Aerospace Corp. (1968). pp. 4230-4246.

[7] J. H. Davis, J. R. Cogdell. Calibration program for the 16-foot antenna, Elect. Eng. Res. Lab., Univ. Texas, Austin, TX, USA, Tech. Memo, Nov, (1987). pp. 15.

[8] Llewellyn, F. B. Some fundamental properties of transmission systems. Proceedings of the IRE. Vol. 3 (1952). pp. 271-283.

[9] B. Roberts, J. Sirkka Jarvenpaa and B. Cherie. Evolving at the Speed of Change. Mastering change readiness at Motorola's semiconductor products sector. Vol. 2 (2008). pp. 3.

[10] G. O. Young, Synthetic structure of industrial plastics, in Plastics. Vol. 3, (1964). pp. 15-64.

[11] Markowitz, M Harry. The elimination form of the inverse and its application to linear programming. Management Science. Vol. 3. (1957), pp. 255-269.

[12] J. Zhang, Jing, T. Nelson. Optical gain and laser characteristics of InGaN quantum wells on ternary InGaN substrates. IEEE Photonics Journal. Vol. 5 (2013), pp. 2600111-2600111.

[13] Azodolmolky, Siamak. Experimental demonstration of an impairment aware network planning and operation tool for transparent/translucent optical networks. Journal of Lightwave Technology. Vol. 4 (2011), pp. 439-448.

[14] J. S. Turner, New directions in communications, IEEE J. Sel. Areas Commun. Vol. 13 (1995), pp. 11-23.

[15] W. P. Risk, G. S. Kino, and H. J. Shaw, Fiber-optic frequency shifter using a surface acoustic wave incident at an oblique angle. Opt. Lett. Vol. 11 (1986), pp. 115-117.

[16] P. Kopyt. Electric properties of graphene-based conductive layers from DC up to terahertz range. IEEE Transactions on Terahertz Science and Technology. Vol. 6 (2016), pp. 480-490.

[17] L. Levitt, Internet Technologies Deployed Behind The Firewall for Corporate Productivity. Internet Society INET'96 Annual Meeting. (1996).

[18] R. J. Hijmans, J. van Etten. Raster: Geographic analysis and modeling with raster data, R Package Version 2.0-12, Jan. 12, (2012).

[19] R. Fardel. Fabrication of organic light-emitting diode pixels by laser-assisted forward transfer. Applied Physics Letters. Vol. 6 (2007). pp. 61-103.

[20] Fulton, A. Sarah. The sense of a woman: Gender, ambition, and the decision to run for congress. Political Research Quarterly. Vol. 2 (2006), pp. 235-248.

[21] Y. C. Chang. Playful toothbrush: ubicomp technology for teaching tooth brushing to kindergarten children. Proceedings of the SIGCHI conference on human factors in computing systems. Vol. 2 (2008), pp.206-221.

[22] D. B. Payne and J. R. Stern, Wavelength-switched pas- sively coupled single-mode optical network, in Proc. IOOC-ECOC, Boston, MA, USA, (1985), pp. 585-590.

[23] D. Ebehard and E. Voges, Digital single sideband detection for interferometric sensors, presented at the 2 nd Int. Conf. Optical Fiber Sensors, Stuttgart, Germany, Jan, (1984). pp. 2-5.

[24] G. Brandli and M. Dick, Alternating current fed power supply, U.S. Patent, Vol. 4 (1978). pp. 084-217.

[25] J. O. Williams, Narrow-band analyzer, Ph.D. dissertation, Dept. Elect. Eng., Harvard Univ., Cambridge, MA, USA, (1993), pp. 439-448.