Design and Analysis of a Novel Selective Boundary Cut Algorithm over a LAN Network

Rajasekhar Cheeli [#1], A. Hari Kumar [*2]
*M.Tech Scholar [#1], Assistant Professor [*2]*
Department of Computer Science & Engineering,
Visakha Institute of Engineering & Technology,
57th Division, Narava, Visakhapatnam, AP (INDIA) - 530027.

Abstract: **A computer network or data network is defined as a telecommunications network which allows computers to exchange data. In computer networks, networked computing devices exchange data with each other along network links . Generally networks are classified into various types based on the usage and type of configuration. Client-Server network is one among the various types of networks, where a client will always sends a request and server will always generates a response. During the communication between each and every nodes in the network the data is mainly divided into packets of equal sizes. However during communication there may be occur some cuts between nodes like edge cut or node cut, this mainly happens due to inactiveness of some intermediate nodes. If any node during the communication becomes inactive it will lead to a network cut between the consecutive nodes which may sometimes leads to edge cut. As the decision tree based algorithms are very efficient in identifying the cuts they are sometimes involve complicated heuristics for determining the fields and the number of cuts. So in this paper we have implemented a novel packet classification algorithm using boundary cutting is proposed where the proposed algorithm is able to find out the cuts that occur during the data communication between nodes.**

Keywords: **Client-Server Network, Wireless Sensor Networks, Data Exchange, Decision Trees.**

1. Introduction

A wireless sensor network (WSN) are defined as a spatially distributed autonomous sensors to *monitor* physical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to a main location. The more modern networks are bi-directional, also enabling *control* of sensor activity. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance; today such networks are used in many industrial and consumer applications, such as industrial process monitoring and control, machine health monitoring, and so on [1],[2].

In a WSN,each node has an RF transceiver, sensor, memory, powered by battery. Nowadays sensors are widely employed in various research fields since they can monitor temperature and hence whether forecasting can be made easier. They are randomly deployed in areas with sensors attached according to the applications for which they are being used. Since they are being powered up by batteries, energy consumption should be minimized in order to prolong the life of sensor nodes. In a network, sensor nodes communicate with each other so that results are obtained as part of their cooperatively combined work. Since each node needs to communicate with all the other nodes,

wireless links are established between them. A cut is defined as the failure of node. It can separate the network into disconnected paths incapable of communicating with each other. Since they are randomly deployed, loss of connectivity can be quite disastrous as they will lead to the breakdown of entire network [3].

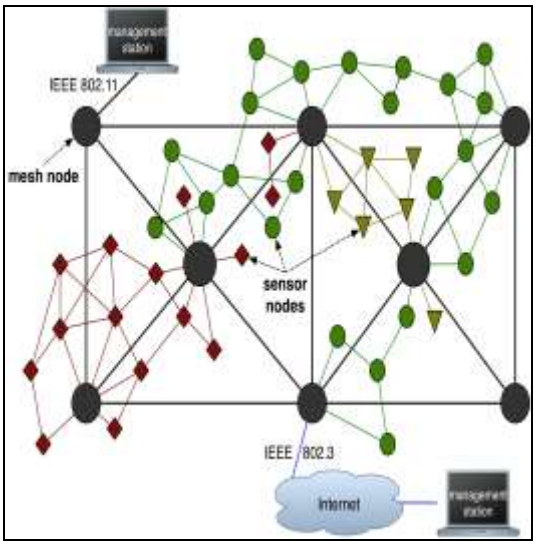

Figure. 1 Represents a Architecture and Working of a Wireless Sensor Network

From figure 1, we can able to find out the working principle of a wireless sensor network, which has a promising technology for monitoring large regions at high spatial and temporal resolution. However, the small size and low cost of the nodes that makes them attractive for widespread deployment also causes the disadvantage of low operational reliability. The node may fail due to different problems. In fact, node failure is expected to be quite common due to the typically limited energy budget of the nodes that are powered by small batteries [2]. Failure of a set of nodes will reduce the number of multi hop paths in the network. Such failures cause a subset of nodes—that have not failed—to become disconnected from the rest, resulting in a"**cut**". Packet classification is an essential function in Internet routers that provides value-added services such as network security and quality of service (QoS) [3]. A packet classifier should compare multiple header fields of the received packet against a set of predefined rules and return the identity of the highest-priority rule that matches the packet header.

Many algorithms and architectures have been proposed over the years in an effort to identify an effective packet classification solution [4]–[6]. Use of a high bandwidth and a small on-chip memory while the rule database for packet classification resides in the slower and higher capacity off-chip memory by proper partitioning is desirable [7]. Performance metrics for packet classification algorithms primarily include the processing speed, as packet classification should be carried out in wire-speed for every incoming packet. Processing speed is evaluated using the number of off-chip memory accesses required to determine the class of a packet because it is the slowest operation in packet classification. The amount of memory required to store the packet classification table should be also considered.

In this paper, we propose a new efficient packet classification algorithm based on boundary cutting. Cutting in the proposed algorithm is based on the disjoint space covered by each rule. Hence, the packet classification table using the proposed algorithm is deterministically built and does not require the complicated heuristics used by earlier decision tree algorithms [8]-[10]. The proposed algorithm has two main advantages. First, the boundary cutting of the proposed algorithm is more effective than that of earlier algorithms since it is based on rule boundaries rather than fixed intervals. Hence, the amount of required memory is significantly reduced.

Second, although BC loses the indexing ability at internal nodes, the binary search at internal nodes provides good search performance.

2. Related Work

In this section we mainly discuss about the related work which was carried out in order to filter out the packets and identify the boundary cut that occur during data transmission.

2.1 Problem Identification

Consider a sensor network modeled as a undirected graph G=(V,E), whose node set V represents the sensor nodes and the edge set E consists of pair of nodes (u, v) such that nodes u and v can exchange messages between each other. Note that we assume inter-node communication is symmetric. An edge (u, v) is said to be incident on both the u and v. The nodes that share an edge with a particular node u are called the neighbors of u. A cut is the failure of a set of nodes Vcut from G results in G being divided into multiple connected components [6]. Recall that an undirected graph is said to be connected if there is a way to go from every node to every other node by traversing the edges, and that a component Gc of a graph G is a maximal connected sub graph of G. We are interested in devising a way to detect if a subset of the nodes has been disconnected from a distinguished node, which we call the source node, due to the occurrence of a cut[11],[12].

2.2 Cut in Graph Theory

In graph theory, a **cut** is a partition of the vertices of a graph into two disjoint subsets. Any cut determines a **cut-set**, the set of edges that have one endpoint in each subset of the partition. These edges are said to **cross** the cut. In a connected graph, each cut-set determines a unique cut, and in some cases cuts are identified with their cut-sets rather than with their vertex partitions. In a flow network, an **s–t cut** is a cut that requires the *source* and the *sink* to be in different subsets, and its *cut-set* only consists of edges going from the source's side to the sink's side. The *capacity* of an s–t cut is defined as the sum of capacity of each edge in the *cut-set*.

A **cut C=(S,T)** is a partition of V of a graph G=(V,E) into two subsets *S* and *T*. The **cut-set** of a cut C=(S, T) is the set $\{(u,v) \in E | u \in S, v \in T\}$ of edges that have one endpoint in *S* and the other endpoint in *T*. If *s* and *t* are specified vertices of the graph *G*, then an ***s–t* cut** is a cut in which *s* belongs to the set *S* and *t* belongs to the set *T*.In an unweighted undirected graph, the *size* or *weight* of a cut is the number of edges crossing the cut. In a weighted graph the **value** or **weight** is defined by the sum of the weights of the edges crossing the cut. A **bond** is a cut-set that does not have any other cut-set as a proper subset[13],[14].

Generally there are mainly two types of cuts that occur while transferring data from one node to other. They are as follows:

a) Minimum Cut
b) Maximum Cut

Now let us look at each cut in with an example as follows:

a) Minimum Cut

A cut is ***minimum*** if the size or weight of the cut is not larger than the size of any other cut. The illustration on the right shows a minimum cut: the size of this cut is 2, and there is no cut of size 1 because the graph is bridgeless.

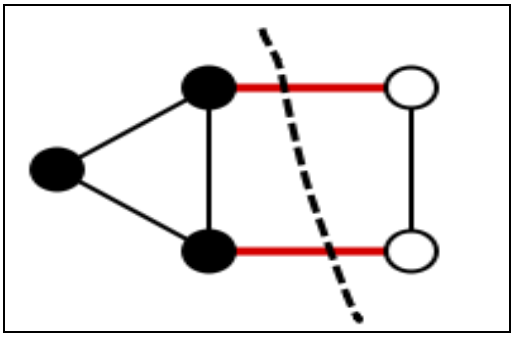

Figure. 2 Represents a Minimum Cut in a Network during Data Transfer

b) Maximum Cut

A cut is ***maximum*** if the size of the cut is not smaller than the size of any other cut. The illustration on the right shows a maximum cut: the size of the cut is equal to 5, and there is no cut of size 6, or $|E|$ (the number of edges), because the graph is not bipartite (there is an odd cycle).

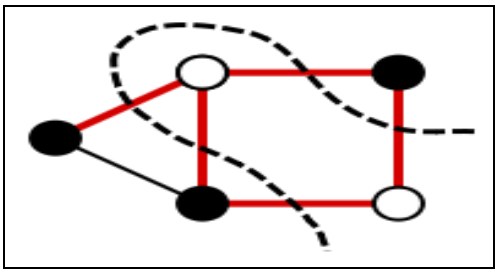

Figure. 3 Represents a Maximum Cut in a Network during Data Transfer

3. A Novel Selective Boundary Cut (NSBC) Algorithm

In this section we will mainly define the novel selective boundary cut algorithm for identifying the packets which are lost during data transmission. For this we have used decision tree along with binary codes for identifying the cut occurred or not in the network.

3.1 Problem Motivation

HiCuts and HyperCuts algorithms perform cutting based on a fixed interval, and hence the partitioning is ineffective in reducing the number of rules belonging to a subspace. Moreover, when the number of cuts in a field is being determined, complicated preprocessing should be made to balance the required memory size and the search performance. In this study, we propose a deterministic cutting algorithm based on each rule boundary, termed as boundary cutting (BC) algorithm. Now let us discuss about the construction of a decision tree based on boundary cut mechanism.

3.2 Building a Novel Boundary Cut Decision Tree Algorithm

When the cutting of a prefix plane according to rule boundaries is performed, both the starting and the ending boundaries of each rule can be used for cutting, but cutting by either is sufficient since decision tree algorithms generally search for a subspace in which an input packet belongs and the headers of the given input are compared for entire fields to the rules belonging to the subspace (represented by a leaf node of the decision tree).

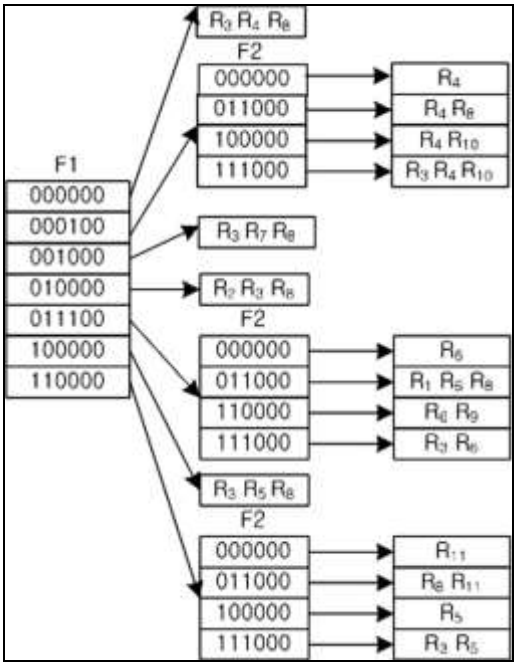

Figure. 4 Represents the decision tree for a Boundary Cut Algorithm

Comparison of the proposed decision tree to the HiCuts decision tree shows that the BC algorithm does not cause unnecessary cutting. Clearly, no two leaves have the same set of rules. Since cutting does not occur in the absence of a boundary, unnecessary cutting is avoided. Hence, rule replication is reduced in the BC algorithm. Additionally, the depth of the decision tree is always less than or equal to six including a leaf since each field is used once at the most.

For example, there are three internal nodes in the second level of the decision tree shown in Fig. 4. For the first internal node, the middle boundary, which is 100000, separates the area covered by the internal node into two partitions, which are [0, 31] and [32, 63] in the F2 interval. Since the first partition has two rules, R4 and R8, there is no need to use the other boundary 011000 included in [0, 31], and since the second partition has three rules (which does not exceed *binth*), R3, R4, and R10, there is no need to use the other boundary 111000 included in [32, 63]. Hence, the internal node results in only two partitions. Similarly, in the second internal node, the middle boundary (110000) separates the area covered by the internal node into two partitions, each of which has three rules; hence, the other boundaries do not have to be used. The same thing happens in the last internal node.

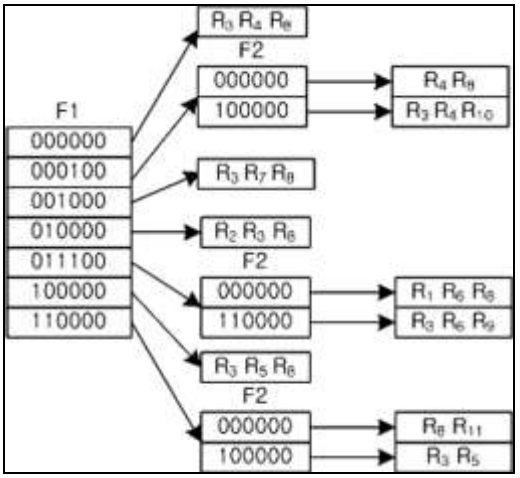

Figure. 5 Represents the decision tree for a Selective Boundary Cut Algorithm

The refined structure of the BC algorithm, called selective BC (SBC), is shown in Fig. 5. At the root node, the middle boundary of each partition is recursively considered to determine whether it should be activated or inactivated, but each partition has more rules than *binth*; hence, all boundaries are used. Upon comparison of the SBC and the BC processes, the number of leaves is reduced from 16 to 10; consequently, the number of stored rules is reduced from 34 to 27.

4. Implementation Modules

Implementation is the stage where theoretical design is converted into programmatically way. Generally in the implementation stage we will divide the application into number of modules in order to make the application develop very easily. We have implemented the proposed concept on Java Platform in order to show the performance this proposed novel selective boundary cut algorithm on a network. The front end of the application takes java swings and socket programming and Back-end takes RMI as a main source for allocating data to the nodes. The application is divided mainly into following 4 modules. They are as follows

1. Building a BC Decision Tree
2. Searching in the Boundary Cutting
3. Selective Boundary Cutting
4. Data Structure

1. Building a BC Decision Tree

When the cutting of a prefix plane according to rule boundaries is performed, both the starting and the ending boundaries of each rule can be used for cutting, but cutting by either is sufficient since decision tree algorithms generally search for a subspace in which an input packet belongs and the headers of the given input are compared for entire fields to the rules belonging to the subspace (represented by a leaf node of the decision tree).

2. Searching in the Boundary Cutting

The cuts at each internal node of the BC decision tree do not have fixed intervals. Hence, at each internal node of the tree, a binary search is required to determine the proper edge to follow for a given input. During the binary search, the pointer to the child node is remembered when the input matches the entry value or when the input is larger than the entry value. Consider an input packet with headers (000110, 111100, 19, 23, TCP), for example; since is used at the root node, a binary search using the header of the given input is performed. The header 000110 is compared to the middle entry of the root node, which is 010000. Since the input is smaller, the search proceeds to the smaller half and compares the input to the entry 000100. Since the input is larger, the child pointer (the second edge) is remembered, and the search proceeds to a larger half. The input is compared to 001000, and it is found to be smaller, but there is no entry to proceed in a smaller half. Hence, the search follows the remembered pointer, the second edge. At the second level, by performing a binary search, the last edge is selected for the header 111100. The linear search, which is the same as that in the HiCuts or HyperCuts algorithm, is performed for rules stored in the leaf node.

3. Selective Boundary Cutting

In this module we propose a refined structure for the BC algorithm. The decision tree algorithms including the BC algorithm use *binth* to determine whether a subspace should become an internal node or a leaf node. In other words, if the number of rules included in a subspace is more than *binth*, the subspace becomes an internal node; otherwise, it becomes a leaf node. In the BC algorithm, if a subspace becomes an internal node, every starting boundary of the rules included in the subspace is used for cutting. We propose a refined structure using the *binth* to select or unselect the boundary of a rule at an internal node. In other words, the refined structure activates a rule boundary only when the number of rules included in a partition exceeds the *binth*.

4. Data Structure

There are two different ways of storing rules in decision tree algorithms. The first way separates a rule table from a decision tree. In

this case, each rule is stored only once in the rule table, while each leaf node of a decision tree has pointers to the rule table for the rules included in the leaf. The number of rule pointers that each leaf must hold equals the *binth*. In searching for the best matching rule for a given packet or the list of all matching rules, after a leaf node in the decision tree is reached and the number of rules included in the leaf is identified, extra memory accesses are required to access the rule table. The other way involves storing rules within leaf nodes. In this case, search performance is better since extra access to the rule table is avoided, but extra memory overhead is caused due to rule replication. In our simulation in this paper, it is assumed that rules are stored in leaf nodes since the search performance is more important than the required memory.

5. Conclusion

In this paper we have implemented a new method like Boundary Cut Detection based on Packet Classification by constructing a binary tree for identifying the packets that was lost during data transmission. While the cutting in the earlier decision tree algorithms is based on a regular interval, the cutting in the proposed algorithm is based on rule boundaries; hence, the cutting in our proposed algorithm is deterministic and very effective. Furthermore, to avoid rule replication caused by unnecessary cutting, a refined structure of the proposed algorithm has been proposed. The proposed algorithms consume a lot less memory space compared to the earlier decision tree algorithms, and it is up to several kilobytes per rule except for FW50K and FW100K. The proposed algorithms achieve a packet classification by 10–23 on-chip memory accesses and 1.0–4.0 off-chip memory accesses in average.

References

[1] .F. Akyildiz and I.H. Kasimoglu, "Wireless Sensor and Actor Networks: Research Challenges,"; Ad Hoc Networks, vol. 2, no. 4, pp. 351-367, Oct. 2004.

[2] "Environmental and Temperature Monitoring", Centrak.

[3] Dargie, W. and Poellabauer, C., "Fundamentals of wireless sensor networks: theory and practice", John Wiley and Sons, 2010 ISBN 978-0-470-99765-9, pp. 168–183, 191–192

[4] G. Dini, M. Pelagatti, and I. M. Savino, "An algorithm for reconnecting wireless sensor network partitions," in *European Conference on Wireless Sensor Networks*, 2008, pp. 253–267.

[5] N. Shrivastava, S. Suri, and C. D. T´oth, "Detecting cuts in sensor networks," *ACM Trans. Sen. Netw.*, vol. 4, no. 2, pp. 1–25, 2008.

[6] H. J. Chao, "Next generation routers," *Proc. IEEE*, vol. 90, no. 9, pp. 1518–1588, Sep. 2002

[7] F. Yu, T. V. Lakshman, M. A. Motoyama, and R. H. Katz, "Efficient multimatch packet classification for network security applications," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 10, pp. 1805–1816, Oct. 2006.

[8] B. Lampson, B. Srinivasan, and G. Varghese, "IP lookups using multiway and multicolumn search," *IEEE/ACM Trans. Netw.*, vol. 7, no. 3, pp. 324–334, Jun. 1999.

[9] S. Dharmapurikar, H. Song, J. Turner, and J. Lockwood, "Fast packet classification using Bloom filters," in *Proc. ACM/IEEE ANCS*, 2006,pp. 61–70.

[10] H.Lim,H.Chu, andC. Yim, "Hierarchical binary search tree for packet classification," *IEEE Commun. Lett.*, vol. 11, no. 8, pp. 689–691, Aug.2007.

[11] P. Gupta and N. Mckeown, "Classification using hierarchical intelligent cuttings," *IEEE Micro*, vol. 20, no. 1, pp. 34–41, Jan.–Feb. 2000.

[12] P. Panda, N. Dutt, and A. Nicolau, "On-chip vs. off-chip memory: The data partitioning problem in embedded processor-based systems,"*Trans. Design Autom. Electron. Syst.*, vol. 5, no. 3, pp. 682–704, Jul.2000.

[13] D. E. Taylor and J. S. Turner, "ClassBench: A packet classification benchmark,"

[14] C. R. Meiners, A. X. Liu, and E. Torng, "Topological transformation approaches to TCAM-based packet classification," *IEEE/ACM Trans. Netw.*, vol. 19, no. 1, pp. 237–250, Feb. 2011.

About the Authors

Rajasekhar Cheeli is currently pursuing his 2 Years M.Tech in Computer Science and Engineering at Visakha Institute of Engineering & Technology, Narava, Visakha District. His area of interests includes Networks and Information Security.

A. Hari Kumar is currently working as an Assistant Professor, in Computer Science and Engineering at Visakha Institute of Engineering & Technology, Narava, Visakha District. His highest qualification is M.Tech in Computer Science and Engineering. He has more than 3 years of experience in teaching field. His research interest includes Data Mining, and Networks.

