The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco, CA, U.S.A., March 1-2, 2013
Doi: 10.7321/jscse.v3.n3.109 e-ISSN: 2251-7545

A Probabilistic Approach to Discover Heterogeneous Network Topologies

Kevin Fuchs
Heilbronn University, Data Center
Heilbronn, Germany
Email: kevin.fuchs@hs-heilbronn.de

***Abstract*—This paper introduces a new approach for the automated discovery of heterogeneous network topologies. The algorithm uses only information that is stored in the Address Forwarding Tables (AFTs) of the network devices. There have been different efforts to find an algorithmic solution using only AFTs. This has always involved the problem that AFTs contain incomplete information, which made it difficult to develop efficient solutions. This paper describes a new probabilistic method, for which the basis is the calculation of degrees to which entries in the AFTs overlap. These overlap degrees are used to calculate interconnection probabilities of network devices. Finally the topology which is the most probable one is selected. Although the search space may become extremely large, the algorithm is pleasantly efficient.**

***Index Terms*—topology discovery, heterogeneous networks, link layer, overlap degrees of address sets, connection probabilities, minimum spanning tree, Kruskal's algorithm**

I. INTRODUCTION

Managing modern computer networks involves many tasks that can only be satisfyingly performed with knowledge about the underlying topology of the network. Amongst others this includes traffic analysis, load balancing and maintenance. Networks often grow independently, which makes it a Sisyphean task to document the network topology manually. For this reason, automatic topology discovery has often been an object of research. Some techniques use end-to-end-measurements. The basic idea behind this is that—depending on the inner structure of the network—there are measurable correlations of network traffic at receivers [1] [2]. Protocols like the Cisco Discovery Protocol (CDP) [3] and the Link Layer Discovery Protocol (LLDP) [4] [5] can be used for layer-2 topology discovery. There has also been work by Yuri Breitbart et al. [6] [7] [8] and Bruce Lowekamp [9] on using the information stored in the Address Forwarding Tables (AFTs). In [8] Breitbart et al. introduce an algorithm that provides an exact solution for topology discovery based on information stored in the AFTs. They also prove that this involves NP-Hardness.

The main objective of this study is the introduction of a new approach using only AFT information. This algorithm does not claim an exact solution. Instead it uses a more efficient probabilistic method, based on overlap degrees between AFTs. A good algorithmic solution should be as independent from particular technologies as possible. Furthermore, it should only use information that is easily accessible on any device. The algorithm introduced in this paper fulfills these requirements.

II. USING AFTS FOR TOPOLOGY DISCOVERY

End-to-end measurements and protocol-based topology inference make clear assumptions about the way network components transmit data through the network. Techniques using end-to-end measurements are commonly based on packet loss rates or delay measurements, detecting correlations of network traffic to infer the topology [1] [2]. Protocol-based techniques require accordant protocols to be implemented and enabled on each network component. Therefore, both end-to-end measurements and protocol-based solutions imply that the network is based on a particular technology. However, an algorithm using only AFT information would infer topologies on a more abstract level.

A switched network can abstractly be described as a tree structure each node of which is able to direct data to one of the network segments that are connected to it. This is done autonomously by each node, meaning that a single node only knows in which network segment a particular destination is located. It does not have any information about the other nodes in the network or the network topology. When data is received by a node it is directed to the network segment where the desired destination is located. The data is sent to the next node which again directs it to its follower node. This way data can be routed from an end point to another one without the single nodes being aware of the network topology. Note that this describes a universal mechanism that is independent from any technology the network is based on. To make this mechanism

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco, CA, U.S.A., March 1-2, 2013
Doi: 10.7321/jscse.v3.n3.109 e-ISSN: 2251-7545

work, every single node must have information about the destinations that are located in the network segments it is connected to. Therefore, the presence of AFTs or something similar to them is always needed, no matter which technology is used to build up the network. Consequently, if we can find an algorithm using only AFT information, this will provide a universal, technology-independent solution for the discovery of switched network topologies.

Supposing all AFTs in the network are complete, the topology can be inferred from the relationships between the address sets that are associated with interfaces. Let I_{ij} denote the *j*th interface on the network device S_i. There are two possible ways to discover connections between devices by the use of the AFTs.

a) Parent and child devices: A particular network device S_c can be identified as the child of its parent device S_p if the union of the address sets corresponding to all downlink interfaces of S_c equates to the address set of the particular downlink interface I_{pj} to which S_c is connected.

b) Unions and intersections: Breitbart et al. propose in [6] that two interfaces I_{ij} and I_{kl} of two different devices are directly connected if the union of their address sets equates to the set of all addresses that are present in the entire network and if the intersection of the two sets is empty.

Point *a)* requires knowledge about the uplink ports on the devices. This is because the union of all interfaces—uplink and downlink interfaces—always reflects all addresses in the entire network. Therefore, uplink ports must be excluded and only downlink ports must be considered. On the contrary, point *b)* does not imply any knowledge about uplinks and downlinks.

As switches store addresses that appear frequently, whereas they discard infrequent ones, we cannot rely on the AFTs to be complete. This is the key problem of finding an algorithmic solution using only AFTs. There has been work on this topic for example by Bruce Lowekamp [9] and Yuri Breitbart [6] [7] [8]. In [8] Breitbart et al. describe an algorithm to solve the problem of incomplete AFTs stating it as a NP-hard problem.

This paper introduces a new probabilistic approach to solve the problem of incomplete AFTs. The AFTs may not be complete but in general they overlap to a high degree. The key of the approach is that we calculate this overlap degree and use it as the basis for the calculation of connection probabilities. This way we can search for the most probable topology. Although the search space may be of huge size we will see that the search can be implemented efficiently.

III. Overlap Degrees of AFTs

First of all we must find a measure to express the overlap degree of two address sets. Fuzzy Logic provides a way to calculate the subset degree of fuzzy sets. In the theory of fuzzy sets an element is not necessarily said to be a member or not a member of a set. Instead, such an element is assigned a membership degree between Zero and One [10]. Referring to [11], the degree to which a fuzzy set *A* is a subset of another fuzzy set *B* is calculated with (1).

$$U(A,B) = \frac{[A \cap B]}{[A]} \tag{1}$$

[*A*] is called the *size* of *A* and is due to (2) with $F_A(x)$ denoting the membership function of the fuzzy set *A* and *x* denoting an element of *A*.

$$[A] = \sum_{x \in A} F_A(x) \tag{2}$$

As an AFT either contains an address entry or not, address sets are not fuzzy but classical sets. But the latter ones can be interpreted as fuzzy sets with each member having the membership degree One [12]. Therefore, the size of an address set equates to the cardinality of it so that we can simplify (1) to (3).

$$U(A,B) = \frac{|A \cap B|}{|A|} \tag{3}$$

Equation (3) describes the overlap degree in the sense of the *subset* degree. But what we actually need is the overlap degree in the sense of the *equality* of two sets. As *equality* means that one set is a subset of the other one and vice versa, we can get the overlap degree by calculating the subset degree in both directions, which equates to (4).

$$\begin{aligned} O(A,B) &= U(A,B) \cdot U(B,A) \\ &= \frac{|A \cap B|}{|A|} \cdot \frac{|B \cap A|}{|B|} = \frac{|A \cap B|^2}{|A| \cdot |B|} \end{aligned} \tag{4}$$

In section II, point *a)* describes a way to identify parent and child devices. Based on this, we can calculate the overlap degree of the address set associated with all downlink interfaces of the device S_c and the address set of the interface I_{pj} , to which S_c—or, more precise, the uplink interface of S_c—is connected. In the following, we use this approach for the calculation of overlap degrees and connection probabilities. Obviously this implies that we know the uplink port of the device S_c. Therefore, in addition, section IX will explain the calculation of overlap degrees using the approach of point *b)* in section II, which does not imply any knowledge about uplinks and downlinks.

There are two different types of address sets to consider: sets according to single interfaces and sets corresponding to network devices. In the following we use the same notation for address sets as for the network devices and interfaces they correspond to. In example, let S_i denote the address set associated with a network device and let I_{ix} represent the

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco, CA, U.S.A., March 1-2, 2013
Doi: 10.7321/jscse.v3.n3.109 e-ISSN: 2251-7545

address set of the *x*th downlink interface of the device S_i. The address set S_i is the same as the union of all its downlink address sets I_{ix}. In (5) X_i denotes the set of downlink interfaces of the device S_i.

$$S_i = \bigcup_{x \in X_i} I_{ix} \quad (5)$$

This way we can calculate the degree to which the address set of a device S_i overlaps with the address set of the interface I_{jk} of another device S_j. Referring to (4), the overlap degree $O(S_i, I_{jk})$ is calculated with (6).

$$O(S_i, I_{jk}) = \frac{|S_i \cap I_{jk}|^2}{|S_i| \cdot |I_{jk}|} \quad (6)$$

IV. Connection Probabilities

In order to compute the probability with which a network device S_i is connected to an interface I_{jk} of another device S_j all possible connections for the device S_i must be considered. Therefore, the connection probability $P(S_i, I_{jk})$ is calculated by normalizing the overlap degree $O(S_i, I_{jk})$ by the sum of the overlap degrees of all possible connections to this device. Connections connecting a device with itself are excluded. Let M be the total number of network devices, let N_m be the number of downlink interfaces on a specific device S_m and let I_{01} be the root of the network—treated as virtual interface. Then the connection probability is calculated with (7).

$$P(S_i, I_{jk}) = \frac{O(S_i, I_{jk})}{\sum_{m=0}^{M} \sum_{n=1}^{N_m} O(S_i, I_{mn})} \qquad m \neq i \quad (7)$$

Figure 1 shows a possible topology of a network consisting of three switches each of them having two downlink interfaces. I_{01} represents the virtual root interface.

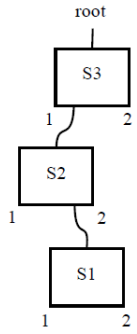


Fig. 1: network with three switches

According to (7), the probability of S_1 being connected to I_{22} equates to (8).

$$P(S_1, I_{22}) = \frac{O(S_1, I_{22})}{O(S_1, I_{01}) + O(S_1, I_{21}) + O(S_1, I_{22}) + O(S_1, I_{31}) + O(S_1, I_{32})} \quad (8)$$

V. Topology Probabilities

Let M denote the number of all network devices in the entire network, and let $I_{k_m l_m}$ be the target interface of the parent device, to which the child device S_m is connected. According to (9), the probability for a complete topology equates to the product of all connection probabilities belonging to this topology.

$$P_{top} = \prod_{m=1}^{M} P(S_m, I_{k_m l_m}) \quad (9)$$

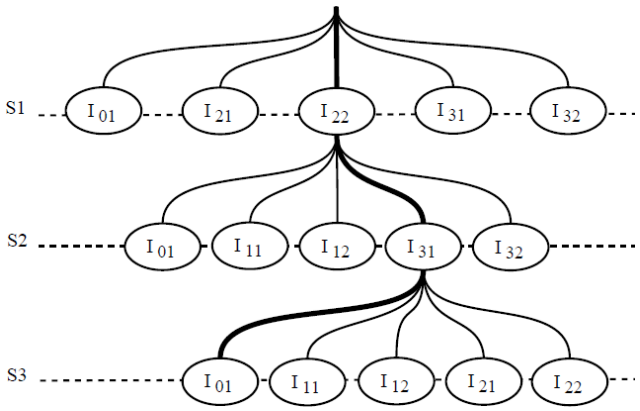
In example, the topology probability of the network shown in figure 1 is equal to (10).

$$P_{top} = P(S_1, I_{22}) \cdot P(S_2, I_{31}) \cdot P(S_3, I_{01}) \quad (10)$$

VI. Search Algorithm

In the previous sections, we have explained how the probability of a potential topology can be calculated. Discovering the most probable topology means that we have to search for it among all possible topologies. Depending on the number of network devices and active interfaces, the search space is expected to become extremely large. Nevertheless, the following remarks show, that even the use of a simple backtracking algorithm is surprisingly efficient. Additionally, in section VIII we discuss the use of Kruskal's algorithm which provides a far more efficient solution.

A. Search Tree The search space is represented by a search tree, the construction of which is illustrated by the means of the example network shown in figure 1. A partial view of the according search tree is shown in figure 2.



The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco, CA, U.S.A., March 1-2, 2013
Doi: 10.7321/jscse.v3n3.109 e-ISSN: 2251-7545

Fig. 2: seaarch tree

The layers of the tree represent the network devices the ordering of which may be arbitrary. The nodes are each allocated an interface I_{ij} the device may be connected to. This way, every possible connection between two devices is equivalent to an edge in the search tree. Furthermore, every single possible topology is represented by a certain branch in the tree, so is the example topology in figure 1 which is represented by the marked branch in figure 2.

B. Backtracking Algorithm Let M be the number of network devices and let N be the number of all downlink interfaces on all devices including the virtual root interface. Then the number of tree nodes is calculated with (11).

$$\sum_{m=1}^{M} N^m \quad (11)$$

The example network in figure 1 leads to a search tree with 399 nodes (M = 3, N = 7). For example, a network consisting of seven devices and overall 70 interfaces will create a search tree with the size of about 8000 billion nodes. Therefore, the use of heuristics is inevitable in order to narrow the search space. Despite the huge size of the search space, we implemented a heuristic search based on a simple backtracking algorithm [13]. This implied using a heuristic condition to tell the search algorithm, when to leave the current search direction and try another one. This condition was easy to find as the search tree has some characteristic properties that can be utilized. The probability of a topology is due to the product of the connection probabilities corresponding to the edges of the branch representing this topology. Because all connection probabilities are smaller than or equal to one, the product of them will decrease the deeper a branch is traversed. We stored the highest probability we had found in a previously traversed branch. When we recognized at a particular level of the current branch that the product of connection probabilities had become smaller than the previously best result we skipped the search for the current branch and continued with another one.

VII. Testing

We used SNMP requests [14] [15] to retrieve AFT information and stored it in a database. The address data might also be retrieved using the telnet protocol [16]. Both SNP and Telnet are supported by almost all network devices which satisfies the condition that an algorithm should only use information that is easily available.

The experimental arrangement consisted of twelve switches and 30 to 60 hosts. Seven to eight of the twelve switches were used to form the topology of the test network. The other switches were used to bundle several hosts to networks. Overall we made thirty different tests comprising flat and deep topologies as well as mixed structures. This included topologies which cannot be guaranteed to be discoverable. These are topologies with several network devices connected in series with none of them having more than one active downlink interface. See figure 3, the devices S_2 and S_3 are connected in series, both of them having only one active downlink interface. The address sets of the interfaces I_{11}, I_{21} and I_{31} as well as the sets corresponding to S_2, S_3 and S_4, are nearly the same ("nearly" means that the AFT of a switch also contains the network address of its neighbor switch). In such cases, in the discovered topology, the devices connected in series may be swapped.

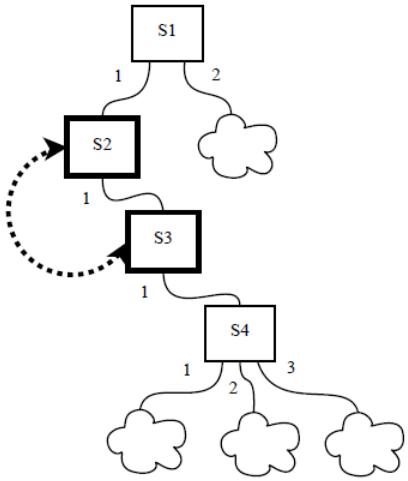


Fig. 3: ambiguous topology

Except for these special topologies 100 per cent of all tests succeeded. Moreover, the "naive" approach to use a simple backtracking algorithm was pleasantly efficient. The backtracking algorithm included counters for nodes that were processed and nodes that were skipped. Although the size of the search tree grew massively with increasing number of interfaces, the search algorithm had to process just a few thousand nodes compared to billions or even 1000 billions of nodes the entire search tree consisted of. The reason for this is the way the overlap degrees are distributed. Addresses that are located in a particular branch of the network will not appear in another branch. Therefore, the majority of the computed overlap degrees and the corresponding connection probabilities are zero or nearly zero, which makes the backtracking algorithm skip large subtrees.

VIII. Kruskal's Algorithm

The backtracking algorithm was implemented with an optional cycle detector in order to predict connections that

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco, CA, U.S.A., March 1-2, 2013
Doi: 10.7321/jscse.v3.n3.109 e-ISSN: 2251-7545

would cause a cycle in the topology. This way, the corresponding connection probabilities could be assigned zero during the backtracking process. The tests showed that the topology discovery worked in both cases—with or without cycle detection. But it emerged that cycle detection has a positive impact on reliability. While constructing the search tree, this increases the belief in non-cyclic topologies as the weights of their probabilities become higher.

If cycles are to be eliminated right from the beginning, the underlying problem amounts to what is known as the Minimum Spanning Tree Problem (MSTP) which is a subject matter of algorithmic graph theory and belongs to the most discussed problems in computer science [17]. The MSTP is about finding a path connecting a set of nodes such that the path forms a spanning tree with minimum cost. The edges between the nodes are given weights. A spanning tree of minimal cost then equates to a graph with the sum of its edge weights being as small as possible. The first one to find an efficient algorithmic solution for the MSTP was Otakar Boruvka in 1926 [18]. However, the two most famous algorithms were published by Joseph B. Kruskal [19] and Robert C. Prim [20], who referenced Boruvka. The following focuses on Kruskal's algorithm, proposing an idea how to use it in order to find the topology with the highest probability.

Boruvka's, Prim's and Kruskal's algorithm belong to the group of greedy algorithms. *Greedy* means that subsequently choosing a local optimum also leads to a global optimum [21]. In the case of the MSTP, the global optimum is a subset of edges connecting all nodes, whereby the sum of the edge weights is minimal. This condition can be satisfied by sorting all edges by their weights and selecting incrementally the smallest remaining edge until all nodes are connected. This is the basic approach of Kruskal's algorithm. In addition, Kruskal adds a new edge only if it does not create cycles in the graph. Accordingly, the complexity of the algorithm is determined by the sorting function and the cycle detection algorithm. An efficient way to implement cycle detection for Kruskal's algorithm is the union-find data structure [22], with which the complexity of Kruskal's algorithm is $O(m\ log(n))$ with m denoting the number of edges and n being the number of nodes [21]. In our case the edge weights are the connection probabilities. The global optimum therefore does not equate to the minimum sum but to the maximum product of the edge weights which in turn is equivalent to the maximum topology probability. The following is a suggestion how a slight modification of Kruskal's algorithm might be used instead of backtracking.

1) Create a trivial graph consisting only of nodes representing the network components S_i and no edges.
2) Sort all possible connections by their probabilities in descending order.
3) As long as not all network components have been connected, repeat the following steps:
   a) Take the connection (S_i, I_{jk}) with the highest probability and delete it from the list.
   b) Discard the connection if
      - it creates a cycle or
      - S_i has already been connected or
      - I_{jk} has already been connected

   Otherwise, add the connection to the graph.

IX. Excluding Uplink Interfaces

Until now we have connected network devices S_c with the interfaces I_{pj} of potential parent devices S_p. This has been done silently implying that the uplink interface of S_c is known. Searching for the interface containing the address of the network root (e.g. the central router) has worked well in practice to identify the uplink port of a device. However, Yuri Breitbart et al. propose in [6] that two interfaces I_{ij} and I_{kl} are directly connected if the union of their address sets contains all addresses occurring in the entire network and if the intersection of I_{ij} and I_{kl} is empty. But the address tables of the interfaces must be complete to make this approach work. In [6] they design a solution based on forcing the AFTs to update. Furthermore, they collect data from the AFTs over a long time period. In [8] they propose an algorithmic solution for topology inference with incomplete AFTs, stating it as a NP-hard problem. But they do not use any probabilistic methods. Instead of calculating overlap degrees based on the relationship between parent and child devices, the following could be done: Let I_{ij} and I_{kl} be two interfaces of interest, and let Q denote the set of addresses occurring in the entire network, which equates to the union of all interfaces on all devices. Then, in reference to (4) we can calculate the overlap degree of $I_{ij} \cup I_{kl}$ and Q with (12).

$$O(I_{ij} \cup I_{kl}, Q) = \frac{|(I_{ij} \cup I_{kl}) \cap Q|^2}{|I_{ij} \cup I_{kl}| \cdot |Q|} \quad (12)$$

Equally—for the intersection of I_{ij} and I_{kl}—we can calculate the non-overlap degree for the interfaces I_{ij} and I_{kl} with (13).

$$\overline{O}(I_{ij}, I_{kl}) = 1 - \frac{|I_{ij} \cap I_{kl}|^2}{|I_{ij}| \cdot |I_{kl}|} \quad (13)$$

Combining these two overlap degrees leads to (14).

$$O_{comb}(I_{ij}, I_{kl}) = O(I_{ij} \cup I_{kl}, Q) \cdot \overline{O}(I_{ij}, I_{kl}) \quad (14)$$

The overlap degree O_{comb} now builds the new basis for the construction of connection probabilities. Finally these probabilities again can be used to search for the most probable

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco, CA, U.S.A., March 1-2, 2013
Doi: 10.7321/jscse.v3.n3.109 e-ISSN: 2251-7545

topology in the same way described in the previous sections. This means that only the address sets used for the calculation of overlap degrees are changed and the basic approach is not affected. The difference is that it is no longer necessary to distinguish uplink interfaces from downlink interfaces.

X. LIMITATIONS AND FUTURE WORK

This paper does not introduce a complete topology discovery mechanism for multi-subnet networks. It only considers topologies of a single subnet in a switched domain. Future work may therefore include the question how this algorithm can be extended to multi-subnets.

Even though the author drafted the use of Kruskal's algorithm, the purpose of this paper does not include providing the most efficient search algorithm. For this reason, future work should also include the implementation of a more efficient search algorithm whereby Kruskal's algorithm might be an obvious solution as explained in section VIII.

As outlined in section IX the calculation of connection probabilities should also be improved so that uplink interfaces do not need to be excluded. Furthermore, future work should contain testing the algorithm on larger networks.

XI. CONCLUSION

In contrast to end-to-end measurements and protocol-based solutions the use of AFT information can provide algorithmic solutions that are independent from the technologies that networks are based on. The incompleteness of AFTs is a key problem of this approach. This paper introduces a probabilistic algorithm considering incomplete information as a natural part of the problem. Instead of solving the problem exactly we content ourselves with probabilistic results.

The algorithm works with heterogeneous networks, it is technology-independent, it produces low error rates and it can be implemented efficiently. The main purpose of this paper is to solve the problem of incomplete AFTs by the calculation of overlap degrees. The author has shown that this approach can be used to implement a reliable and efficient algorithm for the discovery of heterogeneous network topologies. Therefore, this paper provides a small but meaningful piece in the field of topology discovery.

ACKNOWLEDGMENT

This work was created at the data center of the Constance University of Applied Sciences. The author would like to give special thanks to the data center staff for their support.

REFERENCES

[1] M. Coates, M. Rabbat, R. Nowak, *Merging Logical Topologies Using End-to-End Measurements*. 2003
[2] N. G. Duffield, J. Horowitz, F. Lo Prestis *Adaptive Multicast Topology Inference*. In *Infocom 2001. Twentieth Annual Joint Conference of the IEEE No. 3*, pp 1663–1645, 2001
[3] S. R. Rodriguez, *Topology Discovery Using Cisco Discovery Protocol*. 2009
[4] V. Attar and P. Chandwadkar, *Network Discovery Protocol LLDP and LLDP-MED.*
[5] I. Schafer and M. Felser, *Topology Discovery in PROFINET*. Engineering and Information Technology, Berne University of Applied Sciences
[6] Y. Breitbart, M. Garofalakis, B. Jai, C. Martin, R. Rastogi, A. Silberschatz *Topology Discovery in Heterogeneous IP Networks: the NetInventory System.* In *IEEE/ACM Transactions on Networking No.3*, pp. 401–414, 2004
[7] Y. Breitbart, H. Gobjuka *Discovering Network Topology of Large Multisubnet Ethernet Networks*. In *32nd IEE Conference on Local Computer Networks No.3*, pp. 428–435, 2007
[8] H. Gobjuka, Y. Breitbart, *Ethernet Topology Discovery for Networks with Incomplete Information*. In *16th International Conference on Computer Communications and Networks*, pp. 631–638, 2007
[9] B. Lowekamp, D. R. O'Hallaron, T. R. Gross *Topology Discovery for Large Ethernet Networks SIGCOMM '01 Proceedings of the 2001 Conference on Applications, Technologies, Architectures and Protocols for Computer Communications*, New York, 2001
[10] L. A. Zadeh, *Fuzzy Logic*. University of California, Berkeley, 1989
[11] B. Demant, *Fuzzy-Theorie oder die Faszination des Vagen*. Braunschweig, Vieweg, 1993
[12] H. Zimmermann, *Fuzzy Sets, Decision Making, and Expert Systems*. Boston, Kluwer, 1993
[13] R. J. Walker, *An Enumerative Technique for a Class of Combinatorial Problems*. In *Proc. AMS Symp. Appl. Math., Vol. 10*, p. 91–94, 1960
[14] J. Case et al., RFC 1157: *A Simple Network Management Protocol (SNMP)*. 1999
[15] K. McCloghrie and M. Rose, *RFC 1213: Management Information Base for Network Management of TCP/IP-based Internets: MIB-II*. 1991
[16] J. Postel and J. Reynolds, *RFC 854: telnet protocol specification*. 1983
[17] R. L. Graham, P. Hell, *On the History of the Minimum Spanning Tree Problem*. Burnaby: Simon Fraser University, School of Computing Science, 1982
[18] O. Boruvka J. Nešetřil, E. Milková, H. Nešetřilová, *Otakar Boruvka on Minimum Spanning Tree Problem (Translation of the Both 1926 Papers, Comments, History)*. Praha, Charles University
[19] J. Kruskal, *On the Shortest Spanning Subtree of a Graph and the Travelling Salesman Problem* In *Proc. Amer. Math. Soc. No. 7(1956)*, pp. 48–50,1955
[20] R. Prim, *Shortest Connection Networks and Some Generalizations*. In *Bell System Technical Journal No. 36*, pp. 1389–1401, 1957
[21] J. Kleinberg and E. Tardos, *Algorithm Design*. Boston, Pearson/Addison-Wesley, 2006
[22] D. Z. Pan, and Z. B. Liu, X. F. Ding, Q. Zheng, *The Application of Union-Find Sets in Kruskal Algorithm*, In *International Conference on Artificial Intelligence and Computational Intelligence AICI '09*, pp. 159–162, 2009