An Algorithm for Building an Enterprise Network Topology Using Widespread Data Sources

Anton Andreev, Iurii Bogoiavlenskii
Petrozavodsk State University
Petrozavodsk, Russia
{andreev, ybgv}@cs.petrsu.ru

***Abstract*—A lot of network management tasks depend on the description of the network topology. However, the lack of standard methods of the network elements detection, coupled with the incompleteness and heterogeneity of the available topology data, complicate the network topology discovery process. In these conditions, formal models and methods of topology discovery are required. Contribution of the paper is an algorithm for automated enterprise network topology discovery based on a previously developed graph model and criteria for building graph elements. The proposed algorithm is capable of dealing with incomplete heterogeneous data about network topology as well as with the presence in the network of uncooperative devices. The paper also evaluates the algorithm and provides the testing results.**

I. Introduction

Communication infrastructures of modern network service providers (NSP) are complex multilayer systems that include a large number of devices (switches, routers) and internal links.

Most widespread NSP are midsize enterprises that provide services to their own employees and, possibly, to a limited number of smaller enterprises. Such networks, containing up to a thousand of devices and up to ten thousand of network computers, are the subject of this study.

A lot of network management tasks in the enterprise networks, such as capacity planning and root cause analysis, require a complete and detailed description of both logical and physical network topology. Such description could be used to solve a wide range of problems: performance analysis and topology scaling [1], ensuring reliability of the network topology and reducing the number of internal connection [2], etc.

Due to the large sizes of the network producing and maintaining such description manually is very difficult. Therefore arises the problem of the automation of the topology discovery process [3], [4], [5], [6], [7], [8], [9]. Solving this problem is difficult because of a number of issues. The network topology exploration is complicated by the lack of support of standard tools for detecting elements of the network environment and connections between them (e. g. Link Layer Discovery Protocol) by most of the network equipment. This leads to the need for an analysis of heterogeneous data sources including those that are not specifically designed for network discovery (ARP cache, Address Forwarding Tables). Data collection and analysis is complicated by the diversity in modes of implementation of the network device software by vendors and by the possibility of data incompleteness or incorrectness due to network diversity and data obsolescence. Moreover, lack of access to certain devices (hosts and servers, transparent switches and hubs) leads to a need to make a number of assumptions about their properties and connections.

The network topology description is commonly represented as a graph, vertices of which represent network devices, their ports and end points of data transmission protocols, and edges represent hierarchy and data exchange connections. To solve the problem of network topology discovery automation, the authors previously have developed a generalized graph model of an enterprise network's physical, link, and network layers topology [10]. Based on this model, five criteria of the network element interconnection detection were developed and mathematically proven. These criteria could be used with the most widespread heterogeneous data sources on a network topology, such as Address Forwarding Tables (AFT), cache of ARP, CDP, LLDP, STP and its extension (RSTP, MSTP).

The goal of this paper is development, implementation, testing and evaluation of an algorithm for automated building an enterprise network topology based on the model and criteria from [10] as well as a number of new criteria provided in this paper.

A. Organization of the paper

The rest of this paper is organized as follows. The next section provides a short description of the related work. Section III shortly describes the graph model of an enterprise network topology from [10]. Section IV describes the proposed algorithm of network topology graph building. Section V contains description of the algorithm properties and its testing. Section VI concludes the paper.

II. Related work

Existing algorithms of graph building are based on one data source and describe only a limited set of topology properties at a given network layer. For example, algorithms from [4], [5] build graph using AFT and the algorithms from [6] — using Spanning Tree Protocol data. These algorithms are incapable of working in networks that include IEEE 802.1Q virtual local area networks (VLAN).

Authors of [7], [8], [9] proposed algorithms of network topology graph building in presence of VLAN, however, these algorithms require AFT completeness close to 100 percent and a small number of inaccessible devices. Furthermore, [7] provides no result of testing, and [8] uses empirical criteria for connection detection that have no formal proof. The algorithm

ISSN 2305-7254

in [9] does not differentiate between physical and logical connections, so there is no possibility to consider different VLANs separately in the resulting graph.

III. A GRAPH MODEL OF THE NETWORK TOPOLOGY

Let us provide a short description of the model from [10] for the purposes of using its elements in a description of the network topology graph building algorithm.

All elements of the developed model will be presented using the sample network (Fig. 1), which consists of three IP-subnetworks containing one workstation each. Each subnetwork is implemented within a separate VLAN. The isolation of the broadcast domains by VLANs is done by two switches, to which the workstations are physically connected. The router interfaces providing connections between the subnetworks are also physically connected to the switches.

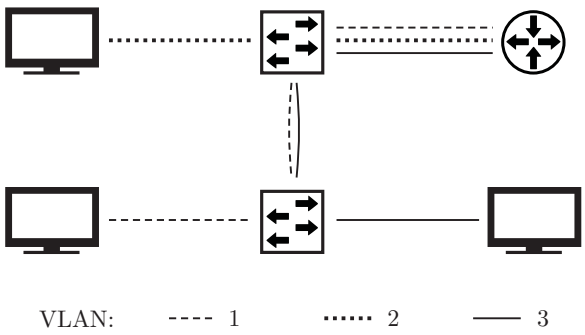

Fig. 1. The layout of the sample network

A. Physical layer

Let us take a nonempty finite set of network devices D. A set of all ports of any device $d \in D$ we will call P_d and a set of all of the ports of all of the devices — P. For any $d \in D$, a set P_d is nonempty and finite. For any two $d_1, d_2 \in D$, $P_{d1} \cap P_{d2} = \varnothing$. Let us define the association relation between the ports and the devices $A^{(1)}$ in such a manner that $(p, d) \in A^{(1)}$ and $(d, p) \in A^{(1)}$ when and only when $p \in P$, $d \in D$, and $p \in P_d$. The relation $A^{(1)}$ is binary, symmetric and irreflexive.

On the given set P, let us take a physical connection relation $L^{(1)}$, which is binary, symmetric, transitive and irreflexive. Two ports $p_1, p_2 \in P$ that are associated with different devices are $L^{(1)}$-related if they are connected by the same data transmission media.

A graph of the physical layer of the sample network (that was introduced in the Fig. 1) is presented in the Fig. 2. On this graph, squares represent devices and circles represent ports.

B. Link layer

Let us define a finite set of labels $VID \subset \mathbb{N}_0$, which correspond to the VLAN identifiers that used in network. In purpose of link layer data transmission using one or more (in case of link aggregation) physical ports each device $d \in D$ creates link layer interface (u, v) where $u \subset P_d$, $v \in VID$. A set of all link interfaces is noted as $I^{(2)}$. For an interface $i = (u, v)$ let us define set $VID_i = \{v\}$. In a case when a

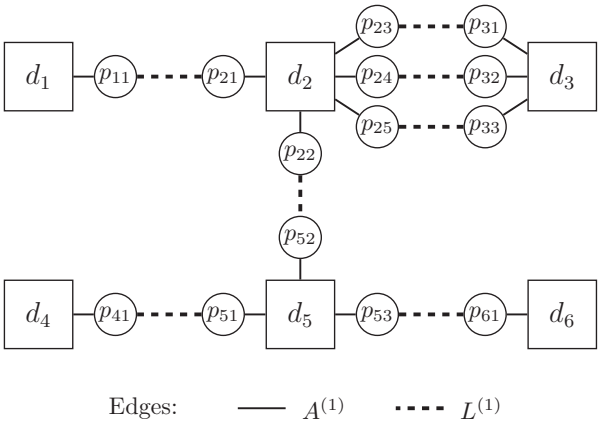

Fig. 2. Graph of the physical layer of the sample network

device associated with the ports from the set u does not support the VLAN technology, $VID_i = \{0\}$.

Let us define the association relation between link interfaces and devices as $A^{(2)}$, so that $(i, d) \in A^{(2)}$ and $(d, i) \in A^{(2)}$ if and only if $i = (u, v)$, $d \in D$ and for all $p \in u$ accomplishes $(p, d) \in A^{(1)}$. The relation $A^{(2)}$ is binary, symmetric and irreflexive. A set of all link interfaces associated with a certain device d we will call $I_d^{(2)}$.

Two device can communicate at the link layer via not blocked (e. g. with a STP) link interfaces, ports of which are connected on a physical layer. On the set $I^{(2)}$, let us define the link layer connection relation $L^{(2)}$ so that two not blocked link interfaces $i_1 = (u_1, v_1) \in I_{d1}^{(2)}$, $i_2 = (u_2, v_2) \in I_{d2}^{(2)}$, associated with different devices, are $L^{(2)}$-related, if $p_1 \in u_1$, $p_2 \in u_2$ exist, where $(p_1, p_2) \in L^{(1)}$. The relation $L^{(2)}$ is binary, symmetric, transitive and irreflexive.

On the set $I^{(2)}$, let us look at the commutation relation $F^{(2)}$, which is binary, symmetric, transitive and irreflexive. The interfaces that are $F^{(2)}$-related to each other must be associated with the same device and be different. Let us interpret the relation $F^{(2)}$ as the following: the configuration of a device provides for a possibility to forward transit data frames between two different link layer interfaces $(i_1, i_2) \in F^{(2)}$.

A graph of the link layer of the sample network is presented in the Fig. 3. Vertices representing link interfaces are depicted as ellipses with ports and VLAN identifiers (refer to Fig. 2) noted inside.

Let us take a look at a graph $\widehat{G}^{(2)} = \langle I^{(2)}, F^{(2)} \cup L^{(2)} \rangle$. The presence of a path between the link layer interfaces in this graph corresponds to the possibility of their communication at the link layer either directly or via a chain of switching devices. As such, the sets of vertices of connected components of the graph $\widehat{G}^{(2)}$ turn out to be the broadcast domains of the network. Partition of the set $I^{(2)}$, each element of which represents one broadcast domain, will be called BD.

For an edge-simple path in the graph $\widehat{G}^{(2)}$ that does not have two consecutive commutation edges, we will use the term *link layer path*. According to the IEEE 801.1D standard, there cannot be more than one link layer path between any two link layer interfaces in the graph.

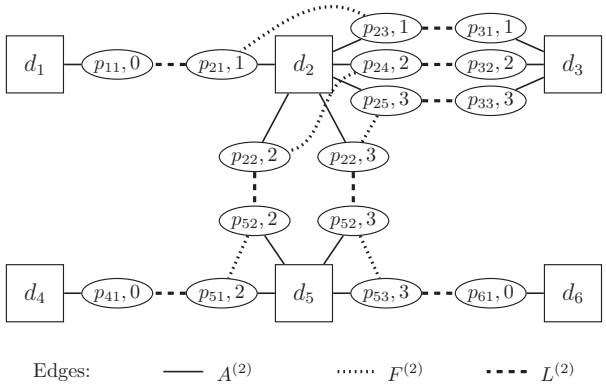

Fig. 3. Graph of the link layer of the sample network

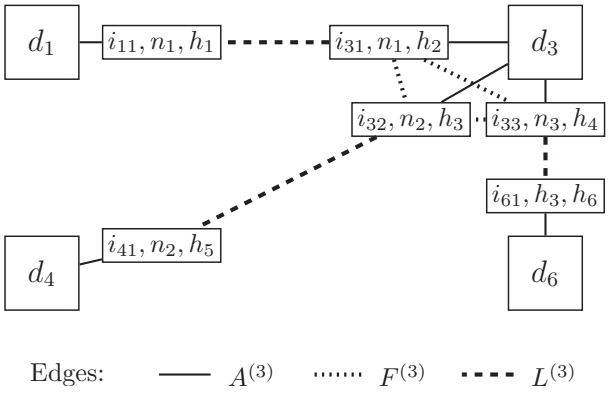

Fig. 4. Graph of the network layer of the sample network

Property 1. *In the link layer graph, between two link layer interfaces, a link layer path exists if and only if the interfaces belong to the same broadcast domain.*

C. Network layer

Let us introduce a set of identifiers of all the network's IP-subnets N. For each subnet $n \in N$, there is a finite set of all possible identifiers of the subnet's hosts — H_n. To transmit and receive data packets at the network layer via link interfaces, the OS of each device $d \in D$ creates a network interface (S, n, h), where $S \subset I_d^{(2)}$, $n \in N$, $h \in H_n$. Let us call a set of all of the network interfaces of all network's devices $I^{(3)}$. Let us define the association relation between network interfaces and devices $A^{(3)}$ so that $(t, d) \in A^{(3)}$ and $(d, t) \in A^{(3)}$ if and only if $t = (S, n, h) \in I^{(3)}$, $d \in D$ and for all $i \in S$, $(i, d) \in A^{(2)}$ is accomplished. The relation $A^{(3)}$ is binary, symmetric and irreflexive.

On the set $I^{(3)}$, let us define the network layer connection relation $L^{(3)}$ so that two network interfaces $t_1 = (S_1, n_1, h_1) \in I^{(3)}$ and $t_2 = (S_2, n_2, h_2) \in I^{(3)}$, associated with different devices, are $L^{(3)}$-related to each other if $n_1 = n_2$ and exist $b \in BD$, $i_1 \in S_1$, $i_2 \in S_2$ so that $i_1, i_2 \in b$. The relation $L^{(3)}$ is binary, symmetric, transitive and irreflexive.

In each subnet there may be devices (routers) that can forward transit datagrams between all connected subnets. Let us look at the routing relation $F^{(3)}$ on the set $I^{(3)}$ that is binary, symmetric, transitive and irreflexive. The interfaces that are $F^{(3)}$-related to each other must be associated with the same device. Let us interpret the relation $F^{(3)}$ as the following: the configuration of a device $d \in D$ provides for a possibility of datagram transmission between two network layer interfaces $(t_1, t_2) \in F^{(3)}$.

A graph of the network layer of the sample network is presented in the Fig. 4. Vertices representing network interfaces are depicted as rectangles with link interfaces, subnet identifiers and host identifiers defining each interface noted inside. There $i_{11} = (p_{11}, 1)$, $i_{12} = (p_{12}, 2)$, $i_{13} = (p_{13}, 3)$, $i_{21} = (p_{21}, 0)$, $i_{51} = (p_{51}, 0)$, $i_{61} = (p_{61}, 0)$.

The structure of any given network can be described with a connected undirected graph $G = \langle V, E \rangle$ — the network topology graph — in which the set of vertices is $V = D \cup P \cup I^{(2)} \cup I^{(3)}$ and the set of edges is $E = A^{(1)} \cup L^{(1)} \cup A^{(2)} \cup F^{(2)} \cup L^{(2)} \cup A^{(3)} \cup F^{(3)} \cup L^{(3)}$. The network topology graph does not contain loops and multiple edges, it may, however, contain cycles.

D. Link layer reachability sets

Within the network, one link layer interface is reachable from another link layer interface if the first interface is an endpoint for data transmission at the link layer for the second interface.

We will use the term *reachability path* for a link layer path in which the first and the last edges are not commutation edges. On the set $I^{(2)}$, we will introduce a reachability relation $\leftrightarrow$ so that $i_1 \leftrightarrow i_2$ if from i_1 to i_2 in the graph G exists a reachability path. In this case we will say that i_2 is reachable from i_1. The relation $\leftrightarrow$ is binary, symmetric and irreflexive.

We will say that a port $p \in P$ is reachable from an interface $i_1 \in I^{(2)}$ if exists such an interface $(u, v) \in I^{(2)}$ reachable from i_1 for which $p \in u$.

For each $i \in I^{(2)}$, we will introduce a set $RS_i \subset I^{(2)}$ that includes all interfaces reachable from i. The set of interfaces reachable from the interfaces that are $F^{(2)}$-related to the i, we will define as CRS_i.

Most data sources provide information about reachability of the network's ports from the network's interfaces. Following the definition of reachability and the Property 1, in order for a port p to be reachable from an interface i_1, it is necessary that the interface $i_1 = (u_1, v_1)$ and some $i_2 = (u_2, v_2)$ (where $p \in u_2$) belong to the same broadcast domain. If there are no data as to whether the reachable port belongs to a VLAN, we will assume that either $v_1 = v_2$ (if it is known that the device supports VLAN) or $v_2 = 0$.

IV. An algorithm of network topology graph building

The algorithm of the network topology graph building process based on the dataset received during network device polling consists of four stages:

1) polling of network devices and obtaining from them data on the structure of the network using SNMP (or

other satisfying tool) and storing these data for further processing;
2) building graph fragments that describe the devices (with their ports and interfaces), existence of which follows directly from the results of the input data analysis;
3) building reachability sets for existing link interfaces using the input dataset and inferring missing records in them;
4) building link layer connection edges based on the analysis of the reachability sets, as well as building physical and network layer connection edges based on the definitions of the model. Additionally, this stage includes inference of the devices, data about which are missing from the network input data.

A. Data collection

The data necessary for building a network topology graph is possible to obtain from Management Information Bases (MIB) of the network devices using Simple Network Management Protocol (SNMP). SNMP protocol is a standard data access tool for network equipment and is enabled for use by default.

The data on the network ports is available in standard IF-MIB and includes their assigned MAC-addresses, names, bandwidth and type: Ethernet, virtual, aggregated etc.

Data on VLANs including their names and tables of port-to-VLAN assignment is available, depending on the vendor, in different MIBs. Standard MIBs are Q-BRIDGE-MIB and SMON-MIB. On Cisco Systems devices, there could also be CISCO-VTP-MIB and CISCO-VLAN-MEMBERSHIP-MIB.

Data on IP-subnets and network interfaces is available in IP-MIB or RFC1213-MIB and includes subnet addresses and masks, along with correspondence of IP-addresses to device interfaces.

The source of data on the contacts within broadcast domains is the device's Address Forwarding Table (AFT), which is maintained for every port of the switch and contains every MAC-address that appear as a source of data frames received on this port. AFT are available in BRIDGE-MIB and Q-BRIDGE-MIB.

The source of data on the contacts within IP-network is Address Resolution Protocol (ARP) cache which contains correspondence of IP-addresses to MAC-addresses within broadcast domain along with a number of interface through which these addresses are reachable. Cache is available in RFC1213-MIB and IP-MIB.

During the work of the Spanning Tree Protocol (STP), network devices determine their direct neighbors, data on which (MAC-addresses and port numbers) is being stored in BRIDGE-MIB.

Cisco Discovery Protocol (CDP) and Link Layer Discovery Protocol (LLDP) are designed to be used exclusively for network topology discovery. Devices that support these protocols send frames with data about themselves through all their ports in predetermined time intervals. These data are stored by neighboring devices in CISCO-CDP-MIB and LLDP-MIB.

Data on routing could be obtained from IP-MIB, OSPF-MIB, BGP-MIB etc.

Data on link aggregation is available in IEEE8023-LAG-MIB, CISCO-LAG-MIB, etc. These data include descriptions of aggregated interfaces on the link layer (MAC-address etc.) and sets of aggregated physical ports.

Network device polling is done by using SNMP and IP-addresses on the condition that authorization parameters are known. When all IP-addresses are known then the data collection process is trivial: sequential polling of the devices and saving the collected data. However, if the full set of manageable devices is unknown and the network polling is conducted from a single known node (e. g. the central router), it is necessary to discover foreign IP-addresses and poll them as well. In this case, during polling of some devices, it is necessary to distinguish and poll previously unvisited IP-addresses. Main sources of foreign addresses are cache of ARP, LLDP, CDP and routing tables. The following algorithm 1 polls network devices and accumulates IP-addresses. Input data for this algorithm is the initial list IPs with IP-addresses for polling. Output data is the $CData$ list with the retrieved datasets.

Algorithm 1 Cumulative algorithm of data collection

$Visited = IPs$
$CData = \varnothing$
while $IPs \neq \varnothing$ **do**
  $ip =$ first element of IPs
  Remove ip from IPs
  if $ip \in Visited$ **then**
    Continue
  end if
  Put ip into $Visited$
  if Device with address ip is accessible **then**
    $data =$ data from device with address ip
    $oIPs =$ own IP-addresses of device from $data$
    $sIPs =$ side IP-addresses from $data$
    Add elements of $oIPs$ to $Visited$
    Add elements of $sIPs$ to IPs
    Add $data$ into $CData$
  end if
end while

return $CData$

Computational complexity analysis for the algorithm 1 and further algorithms will be provided in section V-A.

B. Graph vertex building

The input data for the vertex building process are the datasets received from the network devices during the first stage of the algorithm. The process can be divided into two steps.

The first step includes creation of vertices describing devices that have provided information about themselves along with information about their environment. From that data is derived the information about the device itself, its ports, link and network interfaces, association, commutation and routing edges.

And the second step includes creation of vertices for devices that have not provided data about themselves but existence of which is directly derived from the data received from other devices. It is possible to discover such devices using data on the link and network layers interactions: AFT, routing tables, cache of ARP, CDP, LLDP, STP. CDP and LLDP cache provide names of neighboring devices and their ports along with their IP and MAC addresses. STP cache provides MAC addresses of neighboring ports and their belonging to VLAN. Using AFT and ARP cache it is possible to find MAC and IP addresses of neighboring and remote devices along with their belonging to VLAN.

During the second step of this stage it is possible to create vertices describing such devices as hosts, servers and devices that do not support SNMP or have denied access to themselves (e. g. devices from foreign administrative domains).

C. Reachability set building

The input data of the stage of building reachability sets is the incomplete graph built during the previous stage and the datasets received during the first stage. Initial filling out of the sets is conducted using data form sources described in the section IV-A. During this process it is required to determine the interface corresponding to the reachability data of the current device as well as a reachable remote interface, using methods that depend on a particular data source. As a result for every existing link layer interface $i \in I^{(2)}$ there will be created a potentially empty reachability set RS_i.

The next step of this stage is the process of inferring data of reachability sets possibly containing incomplete data. The algorithm 2 describes the inference process using the methods from section III-D.

Algorithm 2 Reachability set inference

```
Found = true
repeat
  Found = false
  for all i1 ∈ I(2) do
    for all i2 ∈ RSi1 do
      if i1 ∉ RSi2 then
        Add i1 into RSi2
        Found = true
      end if
      for all i3, for which (i2, i3) ∈ F(2) do
        for all i4 ∈ RSi3 do
          if i4 ∉ RSi1 then
            Add i4 into RSi1
            Found = true
          end if
          if i1 ∉ RSi4 then
            Add i1 into RSi4
            Found = true
          end if
        end for
      end for
    end for
  end for
until Found
```

The algorithm 2 is executed while at least one pair of previously unknown to be reachable interfaces is found. After this algorithm is executed, the reachability sets will contain all records that are possible to detect with the provided dataset and known link layer interfaces and commutation edges.

D. Graph edge building

The input data from the edge building stage is the graph with vertices built during the previous stages and the reachability sets of link layer interfaces.

Let us provide a number of the edge building criteria that were proved in [10].

Criterion 1. *If from an interface $i_1 \in I^{(2)}$ only one interface $i_2 \in I^{(2)}$ is reachable and $RS_{i2} = CRS_{i1} \cup \{i1\}$, then $(i_1, i_2) \in L^{(2)}$.*

Criterion 2. *If two interfaces i_1 and i_2 are reachable from each other, and $RS_{i1} = CRS_{i2} \cup \{i_2\}$ and $RS_{i2} = CRS_{i1} \cup \{i1\}$, then $(i_1, i_2) \in L^{(2)}$.*

Criterion 3. *If two ports p_1 and p_2 are $L^{(1)}$-related, then exist such VLAN identifier $v \in VID_{p1} \cap VID_{p2}$ and $u_1 \subset P$, $u_2 \subset P$ so that $p_1 \in u_1$ and $p_2 \in u_2$, and link interfaces $i_1 = (u_1, v)$, $i_2 = (u_2, v)$ that are not blocked, then $(i_1, i_2) \in L^{(2)}$.*

Using the criteria 1–3 it is possible to search and build the link layer connection edges $L^{(2)}$ with the knowledge of reachability sets or physical layer topology.

Criterion 4. *If two link interfaces $i_1 = (u_1, v_1)$ and $i_2 = (u_2, v_2)$ are $L^{(2)}$-related, then for each $p_1 \in u_1$ exists such $p_2 \in u_2$, so that $(p_1, p_2) \in L^{(1)}$ and vice versa.*

With the knowledge of the link layer topology, the criterion 4 allows to seek and build the physical layer connections $L^{(2)}$.

Criterion 5. *If for two net interfaces $t_1 = (S_1, n, h_1)$ and $t_2 = (S_2, n, h_2)$ exist link interfaces $i_1 \in S_1$ and $i_2 \in S_2$ that belong to the same broadcast domain, then $(t_1, t_2) \in L^{(3)}$.*

With the knowledge of the link layer topology, the criterion 5 allows to seek and build the network layer connection edges $L^{(3)}$.

Let us provide a new criterion of $F^{(2)}$ commutation edge detection in addition to the existing criteria set.

Criterion 6. *If for device $d \in D$ and two reachable from each other link layer interfaces i_1 and i_2 it is true that:*

1) *$i_3 \in I_d^{(2)}$ is reachable from i_1, $i_4 \in I_d^{(2)}$ is reachable from i_2;*
2) *from i_1 and i_2 there are no reachable interfaces from $I_d^{(2)}$ other than i_3 and i_4 respectively;*
3) *from interfaces in $F^{(2)}$ relation with i_1 or i_2 there are no reachable interfaces that are associated with d;*

then i_3 and i_4 are $F^{(2)}$-related.

Proof: Following to proposition 3 from [10], the interfaces i_3 and i_4 are located on the reachability path

$w = (i_1, \ldots, i_3, \ldots, i_4, \ldots, i_2)$. Following to the definition of reachability, subpaths $w_1 = (i_1, \ldots, i_3)$ and $w_2 = (i_4, \ldots, i_2)$ of path w have $L^{(2)}$ edges from both ends. As i_3 and i_4 are associated with the same device d, they cannot be $L^{(2)}$-related with each other. Let us suppose that they are not $F^{(2)}$ related. Then subpath $w_3 = (i_3, \ldots, i_4)$ of path w must pass through one more interface $i_5 \in I_d^{(2)}$ because i_3 and i_4 are already included in $L^{(2)}$ edges in subpaths w_2 and w_3 and they could be $F^{(2)}$-related only with interfaces from $I_d^{(2)}$. To be certain, let us say that $(i_3, i_5) \in F^{(2)}$. Then following to the definition of a link layer path, the subpath $w_4 = (i_5, \ldots, i_2)$ of path w could not start with $F^{(2)}$ edge. But then path w_4 satisfies the definition of reachability path and $i_5 \leftrightarrow i_2$ which contradicts the conditions of the criterion. So then, $(i_3, i_4) \in F^{(2)}$. ■

Detection of a new commutation edge may lead to inference of new reachability data between link layer interfaces. This may lead to detection of new connection edges. Taking this into account let us provide the following algorithm 3 of graph edge building.

Algorithm 3 Edge building

```
Found = false
repeat
  Found = false
  Call algorithm 2
  for all i1 ∈ I(2) do
    for all i2 ∈ I(2) \ {i1} do
      if For i1 and i2 one of criteria 1,2,3 is true and
      (li1, li2) ∉ L(2) then
        Add (i1, i2) to L(2)
        Found = true
      end if
      if ∃ d ∈ D such that (i1, d), (i2, d) ∈ A(2) and
      (i1, i2) ∉ F(2) then
        for all i3 ∈ RSi1 do
          for all i4 ∈ RSi2 do
            if For i3 and i4 criterion 6 is true then
              Add (i1, i2) to F(2)
              Found = true
            end if
          end for
        end for
      end if
    end for
  end for
  for all p1 ∈ P do
    for all p2 ∈ P \ {p1} do
      if For p1 and p2 criterion 4 is true and (p1, p2) ∉
      L(1) then
        Add (p1, p2) to L(1)
        Found = true
      end if
    end for
  end for
until Found
```

Detection of network layer connection edges will not impact on the search for other graph edges and vertices, so the search for these edges could be conducted with the algorithm 4 after the algorithm 3 is executed.

Algorithm 4 $L^{(3)}$ edge building

```
for all t1 ∈ I(3) do
  for all t2 ∈ I(3) \ {t1} do
    if For t1 and t2 criterion 5 is true and (t1, t2) ∉ L(3)
    then
      Add (i1, i2) to L(3)
    end if
  end for
end for
```

Once the algorithms 3 and 4 finish their work, the graph will contain all edges between the existing vertices that are possible to detect using the input dataset on the condition that there are no devices that have not been found during the vertex building process.

E. Border device building

Possible incompleteness of the input data or presence in the network of uncooperative devices that are do not provide data about themselves may lead to ambiguous situations during the network topology graph building process. In such situations reachability sets indicate the presence of previously undirected graph edges or vertices but it is impossible to build such elements using any of the criteria provided earlier.

Fig. 5 provides an example of network topology where ambiguous situation may appear. In the ambiguous situation of the first type, device d_2 (gray) is uncooperative and unknown and existence of the link interfaces i_2, i_3, i_4 is unknown. Reachability set of the interface i_1 contains two elements: i_5 and i_6. Devices d_3 and d_4 are also uncooperative and it is known only that $RS_{i5} = RS_{i6} = \{i_1\} \cup RS_{i0}$ following the reachability relation properties. Such situation could appear if d_2 is a transparent switch or hub and d_3 and d_4 are hosts.

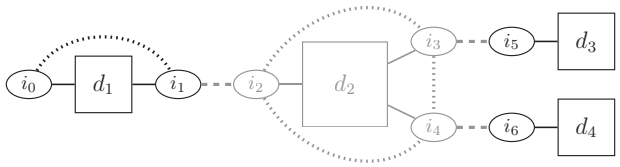

Fig. 5. An example of an ambiguous situation

In the ambiguous situation of the second type, device d_2 is uncooperative but known and it is unknown if the link interfaces i_3, i_4 exist. Reachability set of the interface i_1 contains three elements: i_2, i_5 and i_6. Devices d_3 and d_4 are also uncooperative and it is known only that $RS_{i5} = RS_{i6} = \{i_1\} \cup RS_{i0}$ following the reachability relation properties. Such situation could appear if SNMP-access to d_2 is unavailable and d_3 and d_4 are hosts.

Let us propose a method of resolution of such ambiguous situations. Let us note that data sources used for reachability set construction should not be viewed as equal. Data of CDP, LLDP and STP describes connections with direct neighbors. AFT and ARP cache provides data on reachability that does not necessarily describe direct connections. Herewith AFT and STP cache provides data on VLAN while others do not.

To categorize the reachability data, let us introduce the concept of reachability record priority. Let us define set RSS of ordered pairs of link layer interfaces so that $(i_1, i_2) \in RSS$, if $i_1, i_2 \in I^{(2)}$ and $i_1 \leftrightarrow i_2$. For each interface $i \in I^{(2)}$ let us define set $RSS_i \subset RSS$ so that $\forall (a, b) \in RSS_i$ it is true that $a = i$, $b \in RS_i$. Let us also define a function PR: $RSS \to \mathbb{N}$ using the following rule with dependence on certain data source:

$$PR((i_1, i_2)) = \begin{cases} 0, & \text{source – RS inference} \\ 1, & \text{source – ARP} \\ 2, & \text{source – AFT} \\ 3, & \text{source – CDP or LLDP} \\ 4, & \text{source – STP} \end{cases}$$

Let us name function PR as a reachability priority. Knowing this priority allows us to differentiate reachability data sources and use this knowledge to resolve ambiguous situations.

If data on reachability between two interfaces is found using several data sources then the source with highest priority is chosen.

Let us take the first type of ambiguous situations. We assume that for all $r \in RSS_{i1}$ it is true that $PR(r) < 3$. Let us name device d_2 as a border hub and define criterion of such situation detection.

Criterion 7. *If $|RS_{i1}| > 1$ and $\forall r \in RSS_{i1}$ it is true that $PR(r) < 3$ and also $\forall i \in RS_{i1}$ it is true that $RS_i = \{i1\} \cup CRS_{i1}$ then interface i_1 along with every interface $i \in RS_{i1}$ are connected with border hub.*

This situation will be resolved as follows (based on the example from Fig. 5): creation of device d_2, interface $i_2 \in I^{(2)}_{d2}$ connected with i_1, interfaces i_3 and i_4 connected with i_5 and i_6 respectively.

Let us take the second type of ambiguous situations. We assume that there exists $r_1 \in RSS_{i1}$ for which $PR(r_1) >= 3$ and for all $r \in RSS_{i1} \setminus \{r_1\}$ it is true that $PR(r) < 3$. Let us name device d_2 as a border switch and define criterion of such situation detection.

Criterion 8. *If $|RS_{i1}| > 1$, there exists $r_1 \in RSS_{i1}$ for which $PR(r_1) \geq 3$ and $\forall r \in RSS_{i1} \setminus \{r_1\}$ it is true that $PR(r) < 3$ and also $\forall i \in RS_{i1}$ it is true that $RS_i = \{i_1\} \cup CRS_{i1}$, then interface i_1 along with every interface $i \in RS_{i1}$ are connected to border switch.*

This situation will be resolved as follows (based on the example from Fig. 5): connection of interface $i_2 \in I^{(2)}_{d2}$ to i_1, creation of interfaces i_3 and i_4 connected with i_5 and i_6 respectively.

The algorithm 5 describes a method of resolving ambiguity of both types with possibility that the connection to border device is passing through several VLANs.

The algorithm 5 returns an indicator of success of ambiguity resolution. After this algorithm is executed, every known ambiguous situation of the first and the second types will be resolved. Missing $F^{(2)}$, $L^{(1)}$ and $L^{(3)}$ edges could be discovered and built with use of algorithms 3 and 4

Algorithm 5 Border ambiguity resolving

```
Found = false
for all p ∈ P for which ∄ p_1 ∈ P : (p_1, p) ∈ L^(1) do
  LEAF_1 = {i_1 = (u, i) ∈ I^(2) : p ∈ u and for i_1 criterion 7 is true}
  LEAF_2 = {i_1 = (u, i) ∈ I^(2) : p ∈ u and for i_1 criterion 8 is true}
  if LEAF_1 ≠ ∅ or LEAF_2 ≠ ∅ then
    Found = true
  end if
  if LEAF_1 ≠ ∅ then
    if LEAF_2 ≠ ∅ then
      leaf = FIRST(LEAF_2)
      i_2 = i ∈ RS_leaf for which PR((i_1, i)) ≥ 3
      d_2 = d ∈ D that is associated with i_2
    else
      Create d_2 and add to D
    end if
  end if
  for all i_1 ∈ LEAF_1 ⋓ LEAF_2 do
    i_2 = i ∈ RS_i1 for which PR((i_1, i)) ≥ 3
    if i_2 = NULL then
      Create and add to I^(2) interface i_2, for which VID_i2 = VID_i1
    end if
    Add (i_1, i_2) to L^(2) and (i_2, d_2) to A^(2)
    for all i ∈ RS_i1 do
      Create and add to I^(2) interface i_3, for which VID_i3 = VID_i
      Add (i, i_3) to L^(2) and (i_3, d_2) to A^(2)
    end for
  end for
end for

return Found
```

For simplicity of the algorithm 5 the following moment is omitted from it. During the creation of new ports and interfaces of border devices, it is necessary to take into account connections within different VLANs: if the device is connected to a border device through several link layer interfaces then for new interfaces of this border device the set of ports should be equal.

F. General algorithm

The algorithm 6 describes a general process of building a network topology graph and includes all four stages. The input data for this algorithm are sets of IP-addresses for data collection process and the output is a network topology graph.

V. Evaluation and testing

A system for automated network topology graph building based on the proposed algorithm has been implemented in Java platform with use of Java, Kotlin and Groovy languages. It has 4591 lines of code.

Algorithm 6 General algorithm of network topology graph building

Call algorithm 1 and put result to $CData$
Build graph vertices using $CData$
Initialize reachability sets using $CData$
repeat
  Call algorithm 3 for edge building
  Call algorithm 5 and put result to r
until $r =$ **true**
Call algorithm 4 for $L^{(3)}$ edge building

A. Computational complexity

To estimate the (asymptotic) time complexity of this entire developed algorithm of the network topology graph building let us consider each of its stages individually.

During the first stage (algorithm 1) the poll of unique IP-addresses is done with purpose of data obtaining. Each unique IP-address corresponds to a single network interface and must be visitied mandatory. So the time complexity of the algorithm 1 could be estimated as $\Theta(|I^{(3)}|)$.

The second stage (vertex building) includes handling the dataset obtained during the first stage and creating corresponding vertices. The complexity of this could be estimated as $\Theta(|CData|)$, where $CData$ is the output set of the data collected with algorithm 1.

Construction of the reachability sets using data on remote devices is done during the third stage of the algorithm. For every element of data about remote devices every interface is being checked to establish a conformity. Thus the time complexity of initial reachability set construction could be estimated as $\Theta(N * |I^{(2)}|)$ where N is an amount of data items about remove devices.

The algorithm 2 is used for reachability set inference. During each of its iterations every link layer interface is handled, for which, in turn, reachable interfaces are handled. Then all interfaces $F^{(2)}$-related to the latter are handled, and then all reachable from them interfaces are handled. Therefore, the time complexity of each iteration could be estimated as $O(|I^{(2)}|^4)$. The iterations are run until at least one previously unknown record in reachability set is found. The maximum amount of such records is $C^2_{|I^{(2)}|}$ (a number of two-element combinations). Note that $C^2_n = \frac{n!}{2!(n-2)!} = \frac{n^2-n}{2}$, which is asymptotically close to n^2. Further we will use n^2 as an asymptotic equivalent of C^2_n. Therefore the time complexity of the algorithm 2 could be estimated as $O(C^2_{|I^{(2)}|} * |I^{(2)}|^4) = O(|I^{(2)}|^6)$. However, in most cases most of the previously unknown records are found during first or second iteration so time complexity could also be estimated as $\Omega(|I^{(2)}|^4)$.

The fourth stage of the algorithm is building of graph edges. It starts with the call of the algorithm 3 so let us estimate the complexity of its iterations which start with the call of the algorithm 2. Next the search for $L^{(2)}$ edges is done using criteria 1, 2, 3 during which a comparison of reachability sets of two interfaces is done. The time complexity of each criterion test could be estimated as $O(|I^{(2)}|)$ since complexity of set comparison is $O(N)$ where N is the set size. Next, for each interface, a search for $F^{(2)}$ edges is done using criterion 6. The complexity of this criterion test is $O(|I^{(2)}|)$ since it requires handling reachability sets of taken interfaces. The whole complexity of the stage of $F^{(2)}$ edge search could be estimated as $O(|I^{(2)}|^3)$ since it requires handling pairs of reachable interfaces and testing criteria for them. Therefore the time complexity of this joint stage of search for $L^{(2)}$ and $F^{(2)}$ edges could be estimated as $O(|I^{(2)}|^5)$. However, if most of the $F^{(2)}$ edges were found during input data handling or reachability sets are near to be complete (then $F^{(2)}$ edge is found during first iteration) then this stage complexity could be estimated as $\Omega(|I^{(2)}|^3)$.

Next the search for $L^{(1)}$ edges is done within the algorithm 3. Every pair of ports is examined with criterion 4. The time complexity of testing this criterion is $O(|VID|)$ because every interface based on the current port is being considered and $\Omega(1)$ since criteria could be proved during the first iteration. The entire stage of $L^{(1)}$ edge search is could be estimated as $O(|P|^2 * |VID|)$ and $\Omega(|P|^2)$.

Iterations of the algorithm 3 are executed while at least one $L^{(1)}$, $L^{(2)}$ or $F^{(2)}$ edge is found. Considering that the amount of $L^{(2)}$ edge is significantly larger than $L^{(1)}$ (in case when VLAN spans whole network $|L^{(2)}| = |L^{(1)}| * |VID|$) and not discovered previously $F^{(2)}$ edges (most of them are discovered on vertex building stage) the maximal number of iterations could be estimated as $C^2_{|I^{(2)}|}$ since edges are indirected. And the minimal number of iterations is 1 if no edges were found and 2 if most edges were found during the first iteration.

Therefore the complexity of the entire algorithm 3 is could be estimated as $O(C^2_{|I^{(2)}|} * (|I^{(2)}|^6 + |I^{(2)}|^5 + |P|^2 * |VID|)) = O(|I^{(2)}|^8 + |I^{(2)}|^7 + |I^{(2)}|^2 * |P|^2 * |VID|) = O(|I^{(2)}|^8)$ and $\Omega(1 * (|I^{(2)}|^4 + |I^{(2)}|^3 + |P|^2)) = \Omega(|I^{(2)}|^4)$.

The next step of the algorithm is discovery and resolution of ambiguous situations using the criteria 7 and 8. The complexity of each of these criteria could be estimated as $O(|I^{(2)}|)$ because during testing with these criteria it is required to examine reachability sets of a given interface.

In algorithm 5, for every port, the search for interfaces matching the criteria 7 or 8 is done. New devices and edges are created after that. Therefore, the time complexity of this algorithm could be estimated as $\Theta(|P| * |VID| * |I^{(2)}|)$.

As a result, the time complexity of a single iteration of the main cycle of the algorithm 6 could be estimated as $O(|I^{(2)}|^8 + |I^{(2)}| * |P| * |VID|) = O(|I^{(2)}|^8)$ and $\Omega(|I^{(2)}|^4 + |I^{(2)}| * |P| * |VID|) = \Omega(|I^{(2)}|^4)$. Iterations of the main cycle of the algorithm 6 are done while at least one ambiguous situation is discovered and resolved. Such situations are resolved on the base of ports, therefore the maximal amount of possible ambiguous situations could be estimated as $|P|$ and the minimal amount is 0.

After resolving all ambiguous situations, the search for $L^{(3)}$ edges using the criterion 5 and the algorithm 4 begins. With criterion 5, every pair of link layer interfaces is examined to see whether they are in the same broadcast domain. Following to the property 1, two interfaces are in the same broadcast domain if there is a link layer path between them. It is possible to find such path with depth-first search that has the complexity

$O(|I^{(2)}| + |L^{(2)}| + |F^{(2)}|)$, which, due to the link layer graph properties, could be rewritten as $O(|I^{(2)}|)$. Thus, the time complexity of testing the criterion 5 could be estimated as $O(|I^{(2)}|)$, and the overall complexity of the algorithm 4 is $O(|I^{(2)}|^3)$.

Due to the low time complexity of the data collection, vertex building, reachability set construction and $L^{(3)}$ edges building, the general complexity of the algorithm 6 could be estimated by the complexity of the main cycle of this algorithm, which could be estimated as $O(|P| * |I^{(2)}|^8)$ and $\Omega(|I^{(2)}|^4)$.

B. Data presence influence

The algorithm uses data from network devices which could have only a limited set of information about other devices: their names, MAC-address and name of one of its ports, one IP-address. Therefore, to work successfully the algorithm requires to have an SNMP access to all possible service devices. If this condition is not achieved, the graph will not be totally accurate. Moreover, as has been noted in [4], [10], the data obtained from devices could be incomplete or inaccurate. Despite the ability of the algorithm for data inference, it is not possible to obtain a completely accurate graph in some cases.

In the best case scenario, when a network contains no uncooperative service devices and data between all devices are consistent and complete, then the graph built by the algorithm exactly matches the true network topology graph. But this case is unlikely.

To explore the impact of data incompleteness on the graph building process, we used a network topology graph generator as well as imitated data from the graph devices. Generated data was used to build a graph with the algorithm and after that the newly built graph was compared with the original graph. During this research the following parameters were changed: the size of the generated graph, the number of VLANs, and the AFT completeness. All service network devices were considered as cooperative during the graph generation process.

The AFT completeness for a certain interface is considered to ba a containment of every record on every foreign interface with which it is possible to exchange data on link layer from the current interface. During their data generation, the desired completeness was set in the range from 0 to 1. This rate was used as a possibility of certain record inclusion for AFT of a certain interface. For border switches that have directly connected hosts, the records about these hosts have been included anyway.

The diagram on Fig. 6 was built as a result of 2250 tests with a data generator. It shows dependency of the average rate of correctly discovered physical layer connection edges on the rate of the AFT completeness. Brighter columns here represent an average part of discovered connections between service devices, and darker columns represent the part of total set of connections.

Following the provided data, we can conclude that for the test conditions used, the number of correctly discovered physical layer edges is close to 100% when the AFT completeness is at 60% and more. Also, when the AFT completeness is low

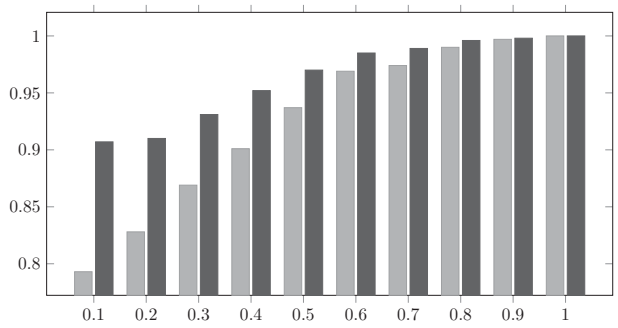

Fig. 6. Dependency of the average number of discovered edges on the AFT completeness

it is still possible to discover more than 80% of edges between service devices due to data inference.

C. Testing in real enterprise network

This algorithm was tested in the network of Petrozavodsk State University (PetrSU) which provided some of its service devices for discovery. There was one router Cisco 7600, four layer 3 switches (which is able to route data between VLANs) Cisco 3750 and Cisco 3850, six different Cisco layer 2 switches. The network had 101 VLANs configured. For every VLAN there was an individual IP-subnet. Moreover, the network contained a lot of uncooperative switches, hosts and servers.

The primary data sources on interconnections in this network appeared to be CDP, STP and AFT. There was no support for LLDP in this network while ARP cache was impossible to use in presence of VLAN for a reachability set construction. Data about uncooperative service devices was obtained using CDP. During the discussion with the network administrators of PetrSU it was confirmed that the built graph completely matched the structure of the explored network segment and its surroundings.

Influence of particular data sources on the accuracy of a graph building process was also studied during tests in the network of PetrSU. Table I provides the number of discovered graph elements depending on the data sources used. First column contains a list of considered sets of graph elements. Here Leaf-1 and Leaf-2 denote ambiguous situations of first and second type respectively. The second column represents the amount of discovered elements when all available data sources were used. Since it has been confirmed that the built graph is correct, the second column represents the actual amount of graph elements. The third column represents the number of discovered elements using all data sources but without reachability set inference using algorithm 2. Columns 4–6 represent the number of discovered elements using one particular data source each: CDP, STP, AFT. The seventh column represents the number of discovered elements with the use of AFT but without reachability set inference.

Based on the data collected during this study, we can conclude that most of the connections could be found using only AFT. However, to obtain a more accurate graph it is necessary to use other data sources as well. CDP and STP data provide an ability to discover connections only with

TABLE I. ELEMENTS FOUND DEPENDING ON THE SOURCES USED

Data sources	All	All without RS inference	CDP	STP	AFT	AFT without RS inference
D	945	907	907	907	915	907
P	1923	1151	1151	1151	1814	1151
$L^{(1)}$	832	63	63	15	796	1
$I^{(2)}$	4483	3512	3452	3452	4337	3452
$L^{(2)}$	1570	672	612	560	1399	101
$F^{(2)}$	14738	8229	8229	8229	14071	8229
$I^{(3)}$	962	962	962	962	962	962
$L^{(3)}$	9679	672	767	103	9061	1
$F^{(3)}$	3717	3717	3717	3717	3717	3717
Leaf-1	6	0	0	0	5	0
Leaf-2	33	0	0	0	29	0

service devices and do not provide any information about hosts. Additionally, using reachability set inference is highly important for building accurate graphs.

VI. CONCLUSION

The description of the logical and physical topology of the enterprise network is required for many network management tasks. The problem of automation of building such description in form of a graph is complicated due to the size and complexity of the network as well as the incompleteness and heterogeneity of available data about network topology. Thereby, to solve this problem, utilization of formal models and methods of the network topology building is required.

The algorithm for building an enterprise network topology graph provided in this paper is based on the formal graph model of the network topology as well as the mathematically proven criteria of graph element discovery and methods of missing data inference from [10]. During the process of developing this algorithm, additional criteria for the discovery of commutating devices, not provided in [10], were designed and proven.

The algorithm tests described in this paper show high accuracy of the network topology graph building even when data is incomplete. This, as well as the polynomial time complexity of the algorithm proven in this paper, shows that the algorithm is suitable for use in practice.

In the future the authors plan to evaluate the actual time complexity and performance of the algorithm when building the topology of real life networks.

REFERENCES

[1] S. Ravindran, C. Huang, K. Thulasiraman, “A dynamic managed VPN service: architecture and algorithms”, *2006 IEEE International Conference on Communications*. 2006, Vol. 2, pp. 664–669.

[2] L. Sivakumar, J. Balabaskaran, K. Thulasiraman, S. Arumungam, “Virtual topologies for abstraction service for IP-VPNs” *2016 17th International Telecommunications Network Strategy and Planning Symposium (Networks)*. 2016,pp. 213–220.

[3] Y. Breitbart, M. Garofalakis, C. Martin, R. Rastogi, S. Seshadri, A. Silberschatz, “Topology discovery in heterogeneous IP networks”, *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*. 2000, Vol. 1. pp. 265–274.

[4] H. Gobjuka, Y. Breitbart, “Ethernet topology discovery for networks with incomplete information”, *Networking, IEEE/ACM Transactions on*. 2010. vol. 18, no. 4. pp. 1220–1233.

[5] Y. Sun, Z. Shi, Z. Wu, “A discovery algorithm for physical topology in switched ethernets”, *Local computer networks, 2005. 30th anniversary. the IEEE conference on*. IEEE, 2005. pp. 311–317.

[6] M.-H. Son, B.-S. Joo, B.-C. Kim, J.-Y. Lee, “Physical topology discovery for metro ethernet networks”, *ETRI Journal*. 2005. vol. 4, no. 27. pp. 355–366.

[7] H. Gobjuka, “Topology discovery for virtual local area networks” *INFOCOM, 2010 Proceedings IEEE*. 2010, pp. 1–5.

[8] L. Zichao, H. Ziwei, Z. Geng, M. Yan, “Ethernet topology discovery for virtual local area networks with incomplete information”, *Network infrastructure and digital content (ic-nidc), 2014 4th IEEE international conference on*. 2014. pp. 252–256.

[9] M. Xiaobo, Y. Tingting, “An algorithm of physical network topology discovery in multi-VLANs”, *TELKOMNIKA*. 2016. vol. 14, no. 3A. pp. 375–379.

[10] A. A. Andreev, A. S. Kolosov, A. V. Voronin, I. A. Bogoiavlenskii, “A graph model of the topology of physical, link and network layers of an enterprise network”, *Proceedings of the 19th Conference of Open Innovations Association FRUCT*. 2016, pp. 3–9.