자치영역 네트워크 토폴로지 작성 알고리즘

민 경 훈† · 장 혁 수††

요 약

네트워크 토폴로지는 네트워크 관리에 매우 중요한 정보를 제공하여 왔는데 현재까지 네트워크 토폴로지에 관한 정보를 얻는 방법은 네트워크를 직접 설계한 관리자에 의한 수 작업 또는 관리자만이 알 수 있는 정보의 입력에 의해서만 작동하는 반자동 시스템에 의해 가능해 왔다. 이러한 시스템은 입력 조건이 자치영역내 모든 IP 주소와 서브넷 마스크 정보가 있거나 최소한 IP 주소의 네트워크 식별자에 대한 정보가 있어야 하기 때문에 네트워크 관리자를 제외하고는 네트워크 전체 토폴로지에 대한 정보를 얻어내기에는 한계가 있다. 본 논문에서는 사용자 누구나 관리자 도움 없이 자치영역내 임의의 IP 주소를 알고 있으면 전체 주소 범위를 파악할 수 있고 아울러 자치영역 경계를 구분할 수 있으며 이를 이용하여 네트워크 토폴로지에 관한 정보를 얻을 수 있는 알고리즘을 제안한다. 아울러 네트워크 토폴로지 정보를 이용한 네트워크 지도 작성 알고리즘도 제공한다.

An Intra-domain Network Topology Discovery Algorithm

Kyong-Hoon Min† · Hyuk-Soo Jang††

ABSTRACT

A network topology has been an important factor for an efficient network management. but data collection for the network configuration has been done manually or semi automatically by a network administrator or an expert Requirements to generate an intra-domain network topology are usually either all IP addresses with subnet/supernet mask or the network identification of all IP addresses. The amounts of traffic are generally high in the semi-automatic system due to using large number of low-level protocols and commands to get rather simple data

In this paper, we propose an algorithm which can be executed with only publicly available input It can find all IP addresses as well as the network boundary of an intra-domain by using an intelligent method developed in this algorithm. The collected data will be used to draw a network map automatically by using a proposed network topology generation algorithm

1. 서 론

A network topology, which shows configuration and connection information about networks, sub-networks and network equipments, takes an important role in solving many network problems. Network path, traffic flow, fault, and traffic congestion as well as hardware and software information can be found through the network topology But there are some problems to acquire necessary data for the network topology with currently available approaches. We review existing algorithms for discovering the network topology and

※ The work was supported (in part) by MIC through supporting program of research center for education on information and communication.

† 준 회 원 명지대학교 대학원 정보통신공학과
†† 정 회 원 명지대학교 전자·정보통신공학부 교수

논문접수 2000년 2월 16일, 심사완료 2000년 4월 11일

find some problems in each of those algorithms.

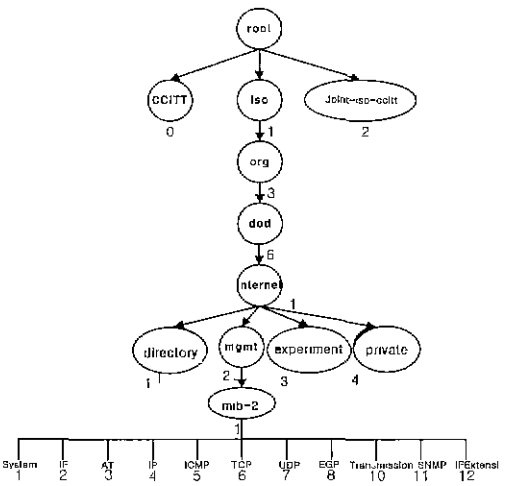
Of the existing algorithms to get network topology information, the algorithm [1, 8] uses multiple low-level protocols and commands including ICMP (Internet Control Message Protocol), "Echo Request / Reply" command, "Address Mask Request / Reply" command, "Traceroute" command, and the DNS (Domain Name System) protocol. It needs all IP addresses in an intra-domain as input conditions. An intra-domain, which is sometimes called an autonomous system [3], is a group of networks and routers controlled by a single administrative authority. The algorithm [2] uses SNMP (Simple Network Management Protocol) with various routing protocols such as BGP (Border Gateway Protocol), OSPF (Open Shortest Path First), and RIP (Routing Information Protocol). It requires the boundary IP of an intra-domain as an input to build an intra-domain network topology. The algorithm [4] also uses SNMP with all IP addresses and subnet masks as input conditions within an intra-domain. Since the algorithms [1, 2, 4, 8] require all IP addresses as input conditions, general public cannot use them without a network administrator's support. Our earlier algorithm [6] can be executed with only small number of input, such as a starting IP naddress, an ending IP address and a community name. But it continues to search the IPs beyond the boundary of an intra-domain in some test site, since the boundary is sometimes not clear with publicly available input. So, an additional mechanism which can identify the boundary between an intra-domain and external ISPs (Internet Service Provider) or other networks is needed.

In this paper, we propose an algorithm which can determine a searching boundary intelligently based on the bit pattern of the given IP address. It uses the SNMP to collect MIBs (Management Information Base) on the network configuration and the collected MIBs will be used to find connection information between routers as well as network equipments. Optionally, it can use the "PING" command to check if subnet information is valid as well as the DNS protocol to enhance human understanding.

The remainder of this paper is organized as follows. Section 2 explains the SNMP protocol, the MIBs, the "PING" command, DNS protocol and how they are used in the algorithm. Section 3 describes the algorithm logic. Section 4 explains how to determine the boundary of an intra-domain and find a network topology from the collected MIBs. Section 5 shows an actual implementation of the algorithm and the test result of the case study from a real site. Section 6 concludes the paper.

2. SNMP, MIBs, PING and DNS used in Algorithm

Most of the NMSs (Network Management Systems) in IP networks focus on collecting information from MIBs which can be obtained by sending the "get-request", "get-next-request", "set-request", "get-response" and "trap" messages of the SNMP. The collected information is usually modified and processed according to the users' need [5]. The standard MIB has a hierarchical structure as in the (Fig. 1) with OIDs (Object Identifier) distinguishing different objects. An OID can be represented either alphabetically like iso.org.dod.internet.mgmt.mib-2.system or numerically like 1.3.6.1.2.1.1. This paper only uses some MIBs of the sub-group of the mib-2 hierarchically and those MIBs are described as follows.



(Fig. 1) MIB structure

2.1 System group

System group has overall information about the system. The objects in this group are *sysDesrc, sysObjectID, sysUpTime, sysContact, sysName, sysLocation, sysService*. The *sysDescr* is a description of the entity, such as hardware, operating system, etc. The *sysObject* is the vendor's authoritative identification of the network management system contained in the entity. The *sysUpTime* is the time since the network management portion of the system was finally reinitialized. The *sysContact* is the identification and contact person for this managed system. The *sysName* is an administratively assigned name for this managed system. The *sysLocation* is the physical location of this system [7]. The *sysServices* is a value that indicates the set of services this entity primarily offers. In the algorithm, this system group information is used to describe a hardware and vendor information of a system and included in the SYSTEM LIST, which will be explained later.

2.2 IF (Interface) group

The interface group contains generic information about the physical interfaces of the entity, including configuration information and statistics on the events occurring at each interface. Each interface is thought as being attached to a sub-network, although an interface to a point-to-point link is also allowed. Implementation of this group is mandatory for all systems [7].

The algorithm uses this group to find only the index number, type, speed, status, and the physical address of the interface of a router and then store this group information in the SYSTEM LIST. The index number is the indexed value, which is simply an integer in the range between 1 and the value of *ifNumber* object, with each interface being assigned a unique number. The algorithm uses this value to distinguish the interface among all interfaces. The number in front of IP address of the (Fig. 5) is the index number. The type is the *ifType* object record which describes the type of interface, such as serial, ethernet, etc., in the algorithm. The speed is the value of *ifSpeed* object which estimates the current capacity of the interface in bit per second and uses to inform the speed of the interface in the algorithm.

2.3 AT (Address Translation) group

The address translation group consists of a single table. Each row in the table corresponds to one of the physical interfaces of the system. The row provides a mapping from a network address to a physical address. Typically, the network address is the IP address for this system at this interface. The physical address depends on the nature of the sub-network. For example, if the interface is a LAN, then the physical address is the MAC address for that interface. If the sub-network is an X.25 packet- switching network, then the physical address may be an X.21 address. The table is indexed by *atIfIndex*, whose value matches that of *ifIndex* for one of the entries in the interface group. The table is also indexed by network address. The table contains an entry only for each interface that uses a translation table [7].

The algorithm uses this group to acquire the IPs and the physical addresses of sub-network which is connected to a router.

2.4 IP (Internet Protocol) group

The IP group contains information relevant to the implementation and operation of IP at a node. Since IP is implemented in both end systems (hosts) and intermediate systems (routers), not all of the objects in this group are relevant for any given system. Objects that are not relevant have null values.

The *ipRouteTable* contains information used for Internet routing. The information in the route table is a relatively general nature and could be extracted from a number of protocol-specific routing tables, such as those for *RIP, OSPF,* and *IS-IS (IGP specified and* standarized by ISO). There is one entry for each route presently known to this entity. The table is indexed by *ipRouteDest*. For each table route, the local interface for the next hop is identified in *ipRouteIfIndex*, whose value matches that of *ifIndex* for one of the entries in the interfaces group. Each entry also indicates, in *ipRouteProto*, the method by which this route was learned [7].

The algorithm uses this group to acquire a network address of a interface and determine connection information between routers. The network address of a interface can be derived from the bitwise-AND operation between the interface IP and the mask address.

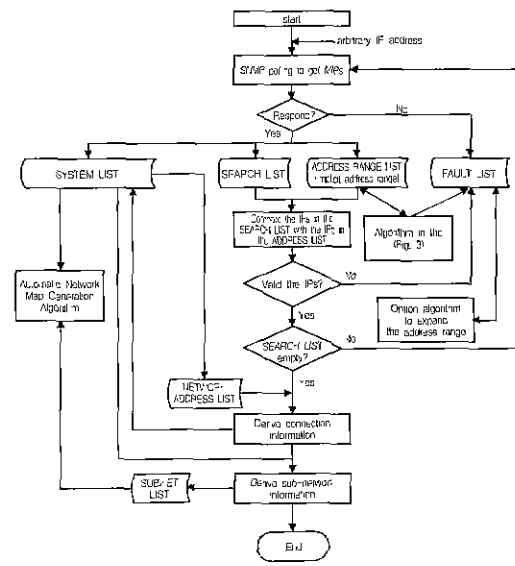
Of the MIBs of the sub-group of the mib-2 in the (Fig. 1), "ICMP", "TCP", "UDP", "EGP", "Transmission", "SNMP". and "IFExtensions" groups are not used in the algorithm, since the algorithm can fully discover the network topology with collected information of the MIB groups, "System", "IF", "AT", and "IP", which belong to the layer 3 and under. The MIB values of those excluded groups belong to the layer 4 and above and some network devices do not support them.

In addition, the algorithm uses the "PING" command and the DNS protocol optionally beside SNMP. Sub-network information which was collected from "AT" group is in the ARP cache. Since the cached data does always not reflect the most current information, the "PING" command is optionally added to solve the inaccuracy problem. The "PING" command can also test if the host is alive and return the round trip time delay. The DNS protocol is used to convert the IP address into the host to enhance human understanding.

3. Algorithm Logic

The algorithm generates six different lists which collect network information for later use to produce a network map. Those lists are as follows ; a SYSTEM LIST for each router information, a SEARCH LIST for the next interesting target IP addresses, a ADDRESS RANGE LIST for the implicit address range, a SUBNET LIST for the sub-network information, a FAULT LIST for the addresses beyond either the address or implicit address range, and a NETWORK ADDRESS LIST for the network addresses of the interface points in the router table.

The algorithm works as depicted in the (Fig. 2) It starts polling with an arbitrary IP to get SNMP MIBs. If it receives response with System, IF, AT and IP MIBs, then the SEARCH LIST, the SYSTEM LIST, and the ADDRESS RANGE LIST are added/ updated according to the MIB values. The FAULT LIST is added if there are no response. The SEARCH LIST contains next interesting IP addresses to be searched. The ADDRESS RANGE LIST keeps track of network IP portion of the IP and holds the current address range to be searched. The procedure of determining the address range is represented in the (Fig. 3).



(Fig 2) The Algorithm Logic Flow

We need to search all IPs in the SEARCH LIST and checks each of the IPs if it is a valid IP within the ADDRESS RANGE LIST. The above searching and checking procedure continues until the SEARCH LIST is empty. If it is not valid, the FAULT LIST is updated and the address range may need to be expanded. Our algorithm allows the address range to be expanded optionally. The NETWORK ADDRESS LIST and the SUBNET LIST are derived from the SYSTEM LIST. Those two lists will be later used to produce a network map.

The algorithm is consisted of two mechanisms, in which the first mechanism collects necessary network information and puts them in the LISTs and the second mechanism generate a network topology usings the LISTs. The first mechanism follows from step 1 to step

6 as in the <Table 1>. The second mechanism is in the <Table 2>.

3.1 The first mechanism for six LISTs

〈Table 1〉 The First Mechanism

Step 1:

If	Poll the starting IP address and receive response
Add collected MIB values to the SYSTEM LIST, the ADDRESS RANGE LIST, and the SEARCH LIST Determine matched prefix bit pattern, address and implicit address range in the ADDRESS RANGE LIST	
Go to step 2	
Else	Add the IP address to the FAULT LIST and stop searching

Step 2:

If	IP address of the SEARCH LIST is included in the address range of the ADDRESS RANGE LIST
Mark flag A(available) at the IP address in the SEARCH LIST and go to step 3	
else if	IP address of the SEARCH LIST is not included in any list
Add IP address to the FAULT LIST	
else if	IP address of the SEARCH LIST is include in the implicit address range
Mark flag A at the IP address in the SEARCH LIST Update address range and implicit address range (details example in the fig 3)	
Go to step 3	

Step 3:

While	Not flagged IP address in the SEARCH LIST
Loop	
If	Poll the IP address of the SEARCH LIST and receive response
Add collected MIB values to the SYSTEM LIST and the SEARCH LIST Go to step 2	
Else	Add the IP address to the FAULT LIST
If	The IP address of the SEARCH LIST appears several times
Remove the IP address from the SEARCH LIST	
End loop	
Go to step 4	

Step 4:

Compute network address by bitwise-AND operation between the IP and its mask and then add it to the NETWORK ADDRESS LIST

Step 5.

Loop	Total number of addresses in the NETWORK ADDRESS LIST
If	There is a duplicate network address in the NETWORK ADDRESS LIST
If	It is found in the different interface IP address of the SYSTEM LIST
Add link information to the SYSTEM LIST	
End loop	
Go to step 6	

Step 6

If	Poll the IP address of the SYSTEM LIST and receive Response
Add AT group values of MIB to the SUBNET LIST	
End of the first mechanism	

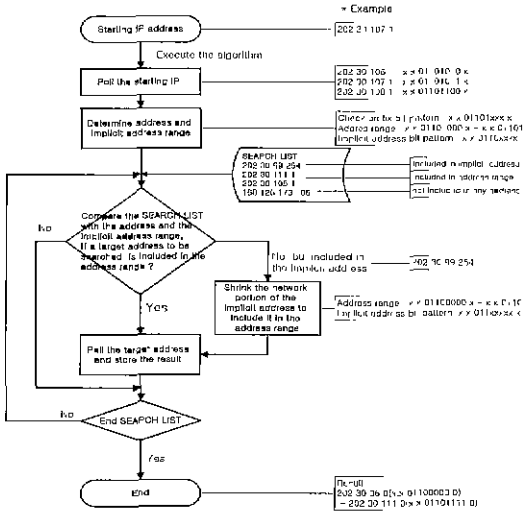
3.2 The second mechanism for network topology

〈Table 2〉 The Second Mechanism

if	All LISTs are generated from the first mechanism
Loop	Total number of routers in the SYSTEM LIST
plot network devices, sub-networks, and their connections	
End loop	
End of the second mechanism	
End of the algorithm	

4. Determining the boundary of an intra-domain and finding a network topology

Our algorithm requires an arbitrary IP address within an intra-domain as a starting IP address to execute the algorithm. The "Community" name should be known prior to run the SNMP to read the MIB values. The starting IP address would be an arbitrary router or a default gateway IP address in the intra-domain. The algorithm determines the IP address range of an intra-domain intelligently from the given IP, since the network address portion of an organization has been generally allocated in a concatenated fashion. The IP address is decided as shown in the (Fig. 3).



(Fig 3) The Procedure of Determining the IP Address Range

First, the algorithm polls starting IP address and collects the interface IPs of the router, and then roughly determines an initial network address range. The initial network address is determined by comparing the prefix bit pattern of each interface IPs. For example, the interface IPs are "202.30.106(01101010).1", "202.30.107(01101011).1", "202.30.108(01101 100).1", then the matched prefix bit pattern is "202.30.01101xxx.x". So, the initial network address range is roughly from "202.30.104(01101000).0" to "202.30.111(01101111).0".

Since the initial network address range might not be enough to cover all IP addresses within an intra- domain, the matched prefix bit pattern of the initial network address range needs to be modified. If the size of the matched prefix bit pattern would be shrunk, we can include more IP addresses into the address range. As a rule of thumb, we simply shrink one bit of the matched prefix bits by truncating the least significant bit and then call it the implicit address bit pattern. All IPs in the implicit address range will have the same implicit address bit pattern. The algorithm checks the IPs of the SEARCH LIST if they are in the address range or the implicit address range. If the target IP of the SEARCH LIST to be searched is in the address range, the algorithm polls the IP and collects necessary information. But if it is beyond the address and implicit address range, it is stored in the FAULT LIST. Since the address in the FAULT LIST is not in the address range. If it is in the implicit address range, the implicit address range becomes a new address range and a new implicit address range is derived by shrinking one more bit as explained above. For example, if the IP address, "202.30.99(0 1100011).254", was in the implicit address range, the new address range becomes "202.30.111(0 1101111).0" and the new implicit address range is "202.30.011x xxxx.0". The algorithm runs iteratively the above steps until the SEARCH LIST becomes empty. The (Fig. 3) shows the procedure of determining the address range based on the above example.

After searching all IPs in the SEARCH LIST, the algorithm tries to find the connection information between routers. We can get the network address through the bitwise-AND operation between the given IP address and the network mask. We get the net mask value from the MIB, "IP group". If an identical network address is found on different network table, we can conclude that there is a connection between routers. We can decide the network type according to the number of identical network addresses appeared in different tables and the interface type has been already known from the MIB, "IF group". The network type could be either point-to-point, multi-point, LAN bus, etc. Finally, the algorithm generates a network topology based on all the collected information.

5. Algorithm Implementation and Test Result in a Real Site

Implementation can be done as follows ; collecting data for router information, computing connection information to other network in each router, searching sub-network, and displaying network topology information. The (Fig. 4) shows the logic flow of the implementation modules.

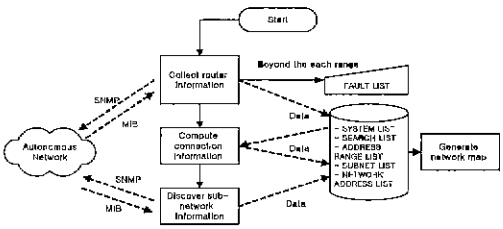
5.1 Collecting router information

This module collects network information such as the

MIB values of the "System", "IF", "IP" group in router by using the SNMP. It also stores the router's configuration and interface information as well as the IP address of the neighboring router as a next target router to be searched. The module also includes the network address range of an intra- domain. So, the module updates the SEARCH LIST, and if there is no IP address in the SEARCH LIST, the algorithm will not continue to search.

5.2 Computing connection information

This module will compute the network address by bitwise-AND operation between IP address of router interface and its mask address and then store the network addresses in the NETWORK ADDRESS LIST. If an identical network address is found on the different network tables, then it has connection among routers. Connection information is stored in the SYSTEM LIST and the NETWORK ADDRESS LIST.



(Fig. 4) The Flow Chart among Implemented

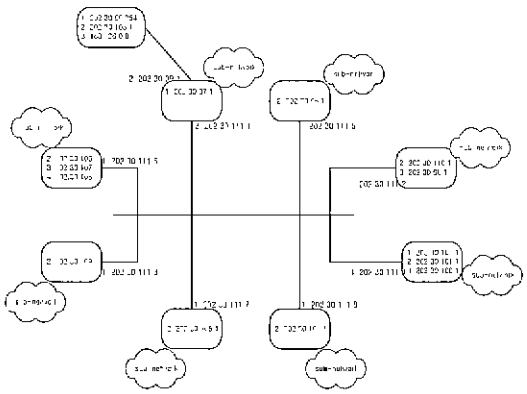
5.3 Discovering sub-network information

Sub-network information can be the MIB values of the "AT" group which is stored in the ARP cache. To solve the cache inaccuracy problem, the module uses "PING" to get accurate sub-network information. To enhance human understanding, the module also uses the DNS protocol to convert IP address into host name. Sub-network information is stored in the SUBNET LIST which is related with the interface of the SYSTEM LIST.

5.4 Displaying network topology information

Once the algorithm collects, processes, and stores the data, it generates a network topology and displays that on the screen. This module visualizes all data.

The algorithm is implemented and tested in a real site. The test site is one of the academic sites and its IP network consists of one router, eight switches, and some hubs. The test result is represented in the (Fig. 5). The initial address of the input is 202.30.107.1 and a community name is "xxx". It takes around 180 seconds to check 9 routers and 9 connections and draws a network topology. The entire network address range of the intra-domain is decided from "202.30.96.0" to "202. 30.111.0", and the prefix bit pattern is "x.x.0110 xxxx.x". The real network address range of test site is from "202.30.96.0" to "202.30.111.0" which is identical to the test result. The (Fig. 5) displays the network connection with routers, the interface IP address of the routers, and the connected sub- networks.



(Fig. 5) The Network Map of the Test Site

6. Conclusion

Even though a network topology takes an important role in solving many network problems, the current existing algorithms find it manually or semi-automatically based on the help from a network expert. In this paper, we propose an algorithm to find out all IP addresses within an intra-domain with only publicly known input data. So, it can be used by the general public. We implement and test the proposed algorithm in a real site. The algorithm collects and saves all

necessary data in the six different lists and draws a topology map out of them. The test result shows that the generated network topology is identical to the real network topology in the test site. In addition, the algorithm does not continue to search the IPs beyond the boundary of the autonomous system. This algorithm will be useful to find a network topology and will work well with any network management software in IP networks. We are currently working on the new version of the algorithm to improve the user interface.

References

[1] J. Schonwalder & H. Langendorfer "How to Keep Track of Your Network Configuration," LISA, 1993.

[2] G. Mansfield, M. Ouchi, K. Jayanthi, Y. Kimura, K. Ohta, and Y. Memoto, "Techniques for Automated Network Map Generation Using SNMP," IEEE INFOCOM, 1996.

[3] Daniel O. Awduche, Angela Chiu, Anwar Elwalid, Indra Widjaja, Xipeng Xiao, "A framework for internet traffic engineering, draft-ietf-te-frame work-00.txt," IETF TEWG, Jan 2000.

[4] Hwa-Chun Lin, Shou-Chuan Lai, and Ping-Wen Chen "An Algorithm for Automatic Topology Discovery of IP Networks," IEEE ICC, 1998.

[5] J. D. Case, M. Fedor, M. Schoffstal, J Davin, "Simple Network Management Protocol(SNMP)," RFC 1157, May 1990

[6] Kyong-Hoon Min, Dae-Soo Kim, Hyuk-Soo Jang, "An Automatic Map Discovery and Generation Algorithm in IP Networks," I-COIN 14, pp.1c3.1-3 5, 2000.

[7] William Stallings, "SNMP, SNMPv2, SNMPv3, and RMON 1 and 2," 3rd Ed, Addison Wesley, 1999.

[8] Hwa-Chun Lin, Hsing-Liand Lai, and Shou-Chuan Lai, "Automatic Link Layer Topology Discovery of IP Networks," IEEE ICC, 1999.

민 경 훈

e-mail : neominga@wh.myongji.ac.kr

1998년 명지대학교 정보통신 공학과(공학사)

1998년~현재 명지대학교 정보통신 공학과 석사 과정

관심 분야 : 인터넷, 컴퓨터통신

장 혁 수

e-mail : jang-h@wh.myongji.ac.kr

1975년~1983년 서울대학교 산업 공학과(공학사)

1986년 미국 Ohio State University 산업 및 시스템 공학 석사

1990년 미국 Ohio State University 전산학 박사

1990년~1992년 경북대 전자공학과 교수

1998년 미국 Ohio State University 전산학과 교환 교수

1992년~현재 명지대 전자·정보통신 공학부 교수

관심 분야 : 인터넷, 컴퓨터통신, 컴퓨터구조