Identifying Network Dynamics with Large Access Graph and Case-Based Reasoning

研究生：林義堯撰

中華民國九十一年年六月
Abstract

This study adopts large access graph algorithm and case-base reasoning approach to generalize user access patterns and diagnose network events respectively for facilitating the network management. Large access graph (LAG) algorithm discovers the frequently inter-connections among hosts to provide an overview of network access relation. The case-based reasoning (CBR) system diagnoses the instant network events with the past experience. NetFlow log data collected from the router of the dormitory network of National Sun Yat-Sen University is used for demonstrating these two methods. The evaluation results measured by recall, precision, and accuracy show that these two mechanisms are useful to support the network administer to keep track of network access relations and diagnose the network events.

Keywords: network management, large access graph, case-based reasoning, case similarity, NetFlow.
中 文 摘 要

本篇論文藉由分析路由器上的流量記錄資料以協助網管人員監控並維護網路正常運作，我們利用高頻存取圖形演算法來找出網路上各種存取狀況的模式，網管人員可以由這些結果，對整個網路環境的存取特性有概括性的瞭解，我們並應用案例式推理方法，將過去網管人員的經驗儲存於案例庫裡面，如果再有類似的狀況發生，馬上可以得到過去判斷的結果。此外，我們利用中山大學學生宿舍網路的流量資料（NetFlow），來驗證我們的方法。透過這兩個機制，網管人員可以更容易地控管網路，以提高網路連線品質。

關鍵字：網路管理、高頻存取圖形、案例式堆裡方法、案例相似度、網路流量分析
# Table of Content

CHAPTER 1 INTRODUCTION ...................................................................................1  
1.1 Background .................................................................................................1  
1.2 Research Objectives ....................................................................................4  
1.3 Research Framework ....................................................................................6  

CHAPTER 2 LITERATURE REVIEW ....................................................................9  
2.1 NetFlow and cflowd ....................................................................................9  
2.2 Access Pattern in The Network ................................................................11  
2.3 Case-Based Reasoning (CBR) ....................................................................14  

CHAPTER 3 DISCOVERING NETWORK ACCESS PATTERNS WITH LARGEST ACCESS GRAPH ...................................................................................18  
3.1 NSYSU Dormitory Network ......................................................................18  
3.2 Data Collection ............................................................................................19  
3.3 Data Pre-processing ....................................................................................20  
3.4 LAG Mining ................................................................................................22  
3.5 Pattern generation .......................................................................................24  

CHAPTER 4 DIAGNOSING INSTANT NETWORKS EVENTS WITH CASE-BASED REASONING ...................................................................................33  
4.1 Problem Statement .......................................................................................33  
4.2 Data Processing ..........................................................................................34  
4.3 Case Representation ....................................................................................36  
4.4 Case Retrieval .............................................................................................39  
4.5 Case Repair and Case Retain ......................................................................40  

CHAPTER 5 EVALUATION AND RESULTS .......................................................41  

CHAPTER 6 CONCLUSIONS AND FEATURE RESEARCH .............................46  
REFERENCES......................................................................................................48  
APPENDIX ...........................................................................................................51
List of Figures

Figure 1.1 Representing an Access Pattern by an Access Graph..........................3
Figure 1.2 Research Framework...........................................................................7
Figure 2.1 The cflowd Architecture (McRobb, 1998)...........................................11
Figure 2.2 The Procedure of LAG Algorithm.....................................................12
Figure 2.3 LAG2 Procedure Algorithm...............................................................14
Figure 2.4 The CBR Cycle (Aamodt and Plaza, 1994)........................................16
Figure 3.1 The Network Topology of NSYSU....................................................18
Figure 3.2 The NetFlow Infrastructure (CISCO, 1999).......................................20
Figure 3.3 Transforming Flow-oriented to Link-oriented Data Records..............21
Figure 3.4 Large-two Graphs.............................................................................24
Figure 3.5 An Example: Two Large-2 Graph Cannot Combine into a Large-3 Graph 26
Figure 3.6 Large-5 Graph – Multicast Service.................................................28
Figure 3.7 Large-8 Graph – LFAP Data Collection.............................................28
Figure 3.8 Large-8 Graph – News Exchange.....................................................28
Figure 3.9 Two Large-6 Graphs – IBM NetView................................................28
Figure 3.10 Large-5 Graph – MUD Service.......................................................28
Figure 3.11 Large-4 Graph – DNS Query...........................................................28
Figure 3.12 Large-6 Graph – BBS service.........................................................29
Figure 3.13 On-Line Game................................................................................29
Figure 3.14 Large-2 Graphs and Large 3 Graph - FTP Service............................30
Figure 3.15 Large 3 Graph and Large 4 Graph– Proxy........................................31
Figure 3.16 Large-5 Graph – Cyber University................................................31
Figure 4.1 Transformation from Raw Data to Connection-oriented Records.......36
Figure 4.2 Structure of a Problem.......................................................................38
Figure 4.3 The FTP Client Case.........................................................................38
Figure 4.4 A Solved Problem with State Description.........................................39
Figure 4.5 The DNS Service Case with Weights of Features............................40
Figure 5.1 The Relationship between $R$, $A$, and $A'$.......................................42
Figure 5.2 The Evaluation Results Measured by Precision.................................43
Figure 5.3 The Evaluation Results Measured by Recall......................................43
Figure 5.4 The Evaluation Results Measured by Accuracy.................................44
List of Tables

Table 1.1 Technology Comparisons (Watson, 1997) .....................................................6
Table 3.1 The Number of Links with Different Flow Size Thresholds.............................22
Table 3.2 The Large Graph Number with Different Flow Size and Thresholds .............23
Table 3.3 The Special Case with Pattern Generation....................................................25
Table 5.2 The Evaluation Results in Precision, Recall, and Accuracy .........................42
Chapter 1 Introduction

1.1 Background

With the rapid diffusion of Internet, it has been an important issue how to effectively and efficiently manage the complex internetworking. Network management involves the process of operation, administration, and maintenance network by extracting network traffic information, and then taking proper actions to maintain network performance (Sugarbroad, 1990). Traditionally, network administrators observe the network status for a period of time and intrude the network operation by tuning routing facilities. The network administrators learned network management domain expertise by solving network problems from time to time. However, a new administrator may spend a great effort to resume the comparative performance of the organizational network management if the organization does not retain its network administration knowledge. Therefore, the retention of network administration knowledge is essential to maintain organizational network performance. Moreover, due to the dramatic change on Internet applications and virus infection, the network traffic patterns may change accordingly. Therefore, it becomes imperative that network management should keep updated information about network traffic in order to react or prevent exceptional events.

Network management involves monitoring and managing network and information systems resources to deliver services to specified service delivery points within specified performance requirements. The International Standard Organization (ISO) has contributed a great effort to network standardization. Its network management model is the primary means for understanding the major functions of network management systems. This model consists of five conceptual areas,
performance management, configuration management, accounting management, fault management, and security management. Among them, the fault management aims to detect, isolate, and correct the problem or the abnormal operation of the network. The past experience can help to detect the fault, isolate, and correct in advance.

The enhanced computation power and the popularized personal computers make the data processing speed up and cost down. Therefore, the technology to transfer data into knowledge is expectable. Research in knowledge discovery in database (KDD) is dedicated for this purpose. KDD is defined as a process of nontrivial extraction of implicit, previously unknown and potentially useful information from data in database (Fayyad, et al., 1996). KDD techniques can be adopted to generalize network traffic data to obtain meaningful patterns for network administrators to operate networks.

Chang (2000) proposed a large access graph method to generalize network access patterns using Kaohsiung Pintung Penghu (KPP) regional network data of Taiwan Academic Netowrk (TANet). Attributed graph was used to represent the relation of common access between hosts, and large access graph (LAG) was used to generalize transaction patterns among hosts. The obtained patterns can bring an overview of the supply chain relationship among hosts, in terms of traffic size, and transaction types. Figure 1.1 shows an example of the large graph discovered by the LAG algorithm.
By virtue of routing log data from routers, theoretically, network administrators can analyze the network performance from these data. However, the abundant log data with tremendously fast growing log data make it infeasible to capture network states from data by human intelligence. LAG makes good use of the log data and discovers user access pattern to provide critical information, which supports network administrators to capture network states.

From the generalized patterns, network administrators can develop specific managerial policies, such as route block or increase network connection bandwidth to improve network performance. Moreover, in order to respond promptly to network state changes, especially in the increase of network attack and virus spray, how to facilitate network administrators to quickly diagnose network problems is getting important. Therefore, in this study, besides obtaining general network access patterns, we also tend to develop network diagnosis mechanisms by making the best use of network traffic data.
1.2 Research Objectives

In this study, we apply the large access graph (LAG) algorithm proposed by Chang (2000) to facilitate the network access pattern generalization, and propose a case-based reasoning approach to identify the network dynamics. In the long-term’s point of view, LAG gives a generalization for a network administrator to swiftly comprehend the access conditions. In the short-term’s point of view, the case-based reasoning method automates the identification of network events within a short time window.

A number of works have done with network management in many methods. Most of them use the bytes count as a critical measurement. Bytes count can only determine “how much” they access the network resource, but it cannot definitely specify a network users’ habit to access the resource. We apply the LAG algorithm based on link number rather than on the bytes count. It can really determine “how often” they access the network resource. The more frequent a user accesses resource, the more definite he or she has a regular relation with the resource providers. LAG employs the data mining technology to generate patterns which are previously unknown, and may contribute to the network management. From the diagnosis findings, network administrators can excise network control mechanisms in order to maintain network in good conditions, and have an insight into the regional network quickly. In this thesis, we will apply LAG method to generalize dormitory network patterns in order to provide network access information for dormitory network management.

In this study, we will also develop an intelligent system with case-based reasoning (CBR) capability to examine network access states to detect network access deviation, and to help the administrator to react to network promptly. CBR is the
approach that solves the problem with past experience. Instead of relying solely on
general knowledge of a problem domain, or making associations along generalized
relationships between problem descriptors and conclusions, CBR is able to utilize the
specific knowledge of previously experienced, concrete problem situations (cases).
A new problem is solved by finding a similar past case, and then reusing its problem
solving strategy to deal with the new problem situation (Aamodt, et al., 1994).

A number of computational technologies are implemented in problem solving
domain. Watson (1997) integrated them with comparisons as shown in Table 1.1.
Access conditions are different from network to network. Therefore, CBR specifies
the network events occurring in an organization that a network administrator can receive recommendations from the past experiences.
Table 1.1 Technology Comparisons (Watson, 1997)

<table>
<thead>
<tr>
<th>Technology type</th>
<th>When to use</th>
<th>When NOT to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Well-structure, standardized data and simple precise queries possible</td>
<td>Complex, poorly structured data and fuzzy queries required</td>
</tr>
<tr>
<td>Information retrieval</td>
<td>Large volumes of textual data</td>
<td>Non-textual complex data types, background knowledge, available</td>
</tr>
<tr>
<td>Statistics</td>
<td>Large volumes of well-understand data with a well-formed hypothesis</td>
<td>Exploratory analysis of data with dependent variable</td>
</tr>
<tr>
<td>Rule-based system</td>
<td>Well-understood, stable, narrow problem area and justification by rule-trace acceptable</td>
<td>Poorly understood problem area that constantly change</td>
</tr>
<tr>
<td>Machine learning</td>
<td>Generalized rule are required from a large training set and justification by rule-trace is acceptable</td>
<td>Rules are not required and justification by rule-trace is unacceptable</td>
</tr>
<tr>
<td>Neural network</td>
<td>Noisy numerical data for pattern recognition or signal processing</td>
<td>Complex symbolic data or when a justification is required</td>
</tr>
<tr>
<td>Case-based reasoning</td>
<td>Poorly understood problem area with complex structured data that changes slowly with time and justification required</td>
<td>When case data is not available, or if complex adaptation is required, or if an exact optimum answer is required</td>
</tr>
</tbody>
</table>

1.3 Research Framework

Referring to Figure 1.2, we propose two mechanisms for a network administrator
to facilitate the network management. The LAG algorithm is applied to mine the association rule among connections. It gives an overview of the access conditions. Besides, the CBR method is adopted in the research to support the administrator to diagnose the network events in a timely fashion.

First of all, the raw data is collected from a regional network router by NetFlow with cflowd (CAIDA, 2001). The raw data is processed and transformed from the flow-oriented into the link-oriented, and then stored in the database for the LAG mining. On the other hand, the raw data is processed and transformed to the case-oriented records for network diagnosis.

For generalizing access relation, we discover the association rules among network transactions. The raw data is first pruned with a threshold to clean the noise. The network access patterns discovered by LAG are stored in the pattern base for retrieval.
network administrators to retrieve salient relations.

For diagnosing instant events, the raw data is processed with the CBR cycle. We transform raw data into cases by extracting features from NetFlow logs. New network events are represented as cases. The similarity is measured between the new case and the cases in case base to retrieve the similar case. The administrator consults the retrieved case for the instant event diagnosis. The solved cases can be added into the case base for matching future cases.

After motivating these research objectives in this chapter, we review literatures relevant to this research in Chapter 2 including network log data collection with NetFlow and cflowd, large access graph algorithm, and the case-based reasoning method. In chapter 3 the LAG algorithm is applied to discovering NSYSU dormitory network access patterns. In chapter 4, we employ the CBR method in diagnosing network dynamics. The empirical studies and the experimental results are described in chapter 5. Finally, chapter 6 concludes this thesis and points out future research in related issues.
Chapter 2 Literature Review

2.1 NetFlow and cflowd

NetFlow was initially developed by Cisco in its Quality of Service (QoS) program. It is a switching method that allows more efficient switching of packets according to the type of packet. NetFlow services capitalize on the flow nature of traffic in the network to provide detailed data collection with minimal impact on router performance and efficiently process access lists for packet filtering and security services.

A network flow is defined as a unidirectional sequence of packets between given source and destination endpoints. Network flows are highly granular; flow endpoints are identified both by IP address as well as by transport layer application port numbers. NetFlow also utilizes the IP Protocol type, Type of Service (ToS) and the input interface identifier to uniquely identify flows. When a router receives a packet for which it currently does not have a flow entry, a flow structure is initialized to maintain state information regarding that flow, i.e. number of bytes exchanged, IP addresses, port numbers, and etc. Each subsequent packet matching the same parameters of the flow will contribute to the byte count and packet count of the flow structure until the flow is terminated and exported. According to “The White Paper of NetFlow Services and Applications” published by Cisco (1999), NetFlow architecture will terminate and export a flow for four reasons:

(1) Flows, which have been idle for a specified time, are expired and removed from the cache.

(2) Long-lived flows are expired and removed from the cache (flows are not allowed
to live more than 30 minutes by default, the underlying packet conversation remains undisturbed).

(3) As the cache becomes full a number of heuristics are applied to aggressively age groups of flows simultaneously.

(4) TCP connections which have reached the end of byte stream (FIN) or which have been reset (RST).

cflowd was developed to collect and analyze the information available from NetFlow flow-export. It allows a user to store the information and enables several views of the data. It produces port matrices, AS matrices, network matrices and pure flow structures. The amount of data stored depends on the configuration of cflowd and varies from a few hundred Kbytes to hundreds of Mbytes in one day per router. The cflowd system contains four major components (CAIDA, 2001), and the architecture is shown as Figure 2.1 (McRobb, 1998).

(1) cflowdmux. This is the program that acts as the receiver of flow-export data from one or more Cisco routers. It writes raw packets into shared memory, and permits clients to have access to raw flow data. An example client (flowwatch) is included.

(2) cflowd. cflowd takes data from raw flows (collected by cflowdmux) and creates tabular summaries of traffic data (AS matrix, net matrix, port matrix, interface matrix, nexthop table and protocol table). It also acts as a server of tabular data to cfdcollect.

(3) cfdcollect. This is a central collector which collects data from instances of cflowd. It is used to archive the tabular data at regular intervals, producing time-series data for each of the tabular data types.
(4) **Utilities.** There are a handful of utilities included in the package which may be used to examine data on the host(s) where cflowd is running.

![cflowd Architecture](image)

**Figure 2.1 The cflowd Architecture (McRobb, 1998)**

### 2.2 Network Access Pattern

Chang (2000) proposed an algorithm called LAG (Large Access Graph) to describe the access pattern on the network. LAG is based on Apriori, proposed by Agrawal and Srikant (1994), to construct the association rules with graph by mining the network traffic log data.

The procedure of LAG algorithm is shown in Figure 2.2 (Chang, 2000). LAG
algorithm can be decomposed into two cases, $k < 3$ and $k \geq 3$.

1) $LG_1 \leftarrow \{\text{Large-one graphs}\}$  //Generates large-one graphs
2) $CG_2 \leftarrow \text{Gen-Candi2-Graph} (LG_1)$;  //Generates candidate-two graphs
3) $LG_2 \leftarrow \text{Gen-Large-Graph} (CG_2)$;  //Generates large-two graphs
4) for $(k \leftarrow 3; LG_{k-1} \neq \emptyset; k++)$ do begin
   5) $CG_k \leftarrow \text{Gen-Candi-Graph} (LG_{k-1})$;  //Generates candidate-k graphs
   6) $CG_{uk} \leftarrow \text{Gen-Uni-Graph} (CG_d)$;  //Generates unique graphs
   7) $LG_k \leftarrow \text{Gen-Large-Graph} (CG_{uk})$;  //Generates large-k graphs
   end
9) return $\bigcup_k LG_k$;

Figure 2.2 The Procedure of LAG Algorithm

When $k < 3$, it generates the large-one graphs by scanning the database. Then it generates candidate-two graphs from union of two large-one graphs, which share at least one vertex. Then, it generates large-two graphs fitting with minimum support. When $k \geq 3$, it iterates the following three steps. First, the large-$(k-1)$ graphs found in $(k-1)$th pass are used to generate the candidate graphs $CG_k$. Next step is used to prune the duplicated candidate graphs and meet non-subset closure condition. In the final step, the network database is scanned to count the support of these candidate-$k$ graphs, and then large-$k$ graphs are obtained. Repeat the process until no new large graphs are found.

As $k \geq 3$, we need spend additional efforts on generating candidate-$k$ graphs (Chang, 2000).

(1) **Pruning the duplicated candidate graphs.** In generating candidate itemsets, if $k-1$ items out of a large-$k$ itemset have the same items with the other large-$k$ itemset, they can be combined into a candidate- $(k+1)$ itemset. In such condition,
we can only check if first \((k-1)\) items are the same between two itemsets to generate the candidate itemset. To prevent from missing candidate graphs, we must search possible \((k-1)\) links between two linksets. However, this may result in duplicating candidate graphs, and we need the extra effort to prune duplicate candidates.

(2) **Relaxing the subset closure constraint.** The subset closure property says that all the \((k-1)\)-itemsets of a large \(k\)-itemset should be large \((k-1)\)-itemset. In other words, if any of these subsets are found not to be large, the candidate itemset cannot be large and it is discarded. The subset closure property is implemented to prune candidate itemsets during mining iterations. In our problem to find large graphs, there is no subset closure property resulted from the connectivity property.

To improve the performance of LAG, Tzeng (2001) proposed LAG\(_2\) algorithm to reduce the number of database scans and the counting of irrelative candidate itemsets. When the database table is wide, the column-wise approach to access data is often more efficient than standard row-wise approach (Dunkel and Soparkar, 1999). The table of the network log data is wide because it has a great number of different links in a short interval. Park, Chen and Yu (1995) propose an effective hash-based algorithm for mining association rules. Building hash-table reduce the number of scans or one scan. Searching hash-table is faster than the table of database for counting.

LAG\(_2\) changes the data structure from row-wise to column-wise hash-table after the first scan. In LAG, it needs the number of scans. LAG\(_2\) only scans table once. To merge and to intersect the rows of hash-table makes the performance better for counting candidate itemsets. The LAG\(_2\) algorithm is shown in Figure 2.3 (Tzeng,
2001). It has three major procedures: changing data structure, generating candidate graphs, and generating large graphs. First, LAG$_2$ changes the data structure from row-wise to column-wise after the first database scan. The candidate graph is generalized by two large graphs by the Gen-Candidate-Graph step. The Large graph is generalized so that the number of occurrence in the time interval is greater than the expected minimum support at one of candidate graph by the Gen-Large-Graph step. This loop continues until large graph set is empty.

2.3 Case-Based Reasoning (CBR)

Case-based reasoning basically is a solution “to solve a new problem by remembering a previous similar situation and by reusing information and knowledge of that situation” (Aamodt and Plaza, 1994). It is a problem solving paradigm that is fundamentally different from other major AI approaches in many respects. Instead of relying solely on general knowledge of a problem domain, or making associations along generalized relationships between problem descriptors and conclusions, CBR is able to utilize the specific knowledge of previously experienced, concrete problem situations (cases). A new problem is solved by finding a similar past case, and by reusing it in the new problem situation. A second important difference is that CBR also is an approach to incremental and sustained learning since a new experience is
retained each time a problem has been solved, which makes it immediately available for future problems.

The CBR cycle is comprised of four processes, as a mnemonic, "the four REs" (Aamodt and Plaza, 1994): (1) retrieve the most similar case(s) from case-library or case-base, (2) reuse the solution from the most similar case if appropriate, (3) revise or adapt the proposed solution if necessary, and (4) retain the new case and its solution for future use.

Each process involves a number of more specific steps, which will be described in the task model. In Figure 2.4, CBR cycle is illustrated. An initial description of a problem (at top of Figure 2.4) defines a new case. This new case is used to retrieve a case from the collection of previous cases (case base). The retrieved case is combined with the new case through reuse into a solved case, a proposed solution to the initial problem. In the revise process, this solution is tested for success, e.g. by being applied to the real world environment or evaluated by an expert, and repaired if failed. During the retain process, useful experience is stored as the case for future reuse, and the case base is updated by a new learned case.
A case is a contextualized piece of knowledge representing an experience. It contains the past experience that is the content of the case and the context in which the experience can be used. There are three major parts of a case, although for some particular cases, they may not be all filled in (Kolodner, 1993):

(1) **Problem/situation description**, the state of the world at the time the case was happening and, if appropriate, what problem needed solving at that time,

(2) **Solution**, the derived solution to the problem specified in the problem description, or the reaction to its situation, and
(3) **Outcome**, the resulting state of the world when the solution was carried out.

Cases, which comprise problems and their solutions, can be used to derive solutions to new problems. A solution comprising problems and outcomes can be used to evaluate new situations. If, in addition, such cases contain solutions, they can be used to evaluate the outcome of proposed solutions and prevent potential problems. Cases can be represented in a variety of forms using the full range of AI representational formalisms including frames, objects, predicates, semantic nets and rules.
Chapter 3
Discovering Network Access Patterns with Large Access Graph

In this chapter we employ the LAG algorithm to discover network access patterns of students residing in the dormitories of National Sun Yat-Sen University (NSYSU). LAG algorithm discovers association rules in the network transaction log. The network administrator examines the pattern generating from the LAG algorithm to capture network access relationship.

3.1 NSYSU Dormitory Network

In our study, we focus on the network of dormitory at the National Sun Yat-Sen University (NSYSU). The dorm network (Dorm-Net) is composed of 25 class C subnetworks, where the number of hosts of each subnetwork is up to 254. The core router is Cabletron SSR 2000. Switches segment each subnet. The bandwidth, 100Mbps, is sufficient for the regular traffic. Figure 3.1 shows the network topology of the NSYSU.

![Figure 3.1 The Network Topology of NSYSU](image)

Currently, the Dorm-Net is regulated by certain policies to ensure fair access.
The speed of data flowing from Dorm-Net to TANet is limited at 128 kilobits per second. The data flowing from TANet to the Dorm-Net is allowed up to 5 gigabytes per week. However, there is no restriction between flows from Dorm-Net to NSYSU’s campus network. It means that students in dormitories can freely access data from campus, but are limited from accessing TANet backbone due to the bandwidth congestion of TANet backbone. With this particular policy, the Dorm-Net is like a semi-open network. By observing the network, we cannot only identify the machine operation in terms of the transactions among network servers but also the student behavior in accessing network resources.

### 3.2 Data Collection

First, we use the cflowd to collect NetFlow data from the Dom-Net core router of NSYSU. The NetFlow infrastructure is shown in Figure 3.2. The cflowd collects data every 5 minutes. We collect network traffic data from December 1 to December 21 in 2001 to demonstrate the discovery of network access patterns.

The flow number of every five minutes in the log file is around 80,000 flows in the peak periods, 1,000 flows in the non-peak periods. The average is 40,000 flows per 5 minutes. There are 288 five-minute data, more than 10 million flows collected in one day. The compressed cflowd data is about 0.5 MBytes per five minutes, 4 to 5 GBytes per month.

The schema of flow record is defined as

\[
\text{Flow} \{ \text{Terminating Date}, \text{Terminating Time}, \text{Source IP}, \text{Source Port}, \text{Destination IP}, \text{Destination Port}, \text{Protocol}, \text{Packets}, \text{Bytes} \}
\]

Every five minutes, cflowd transfers the router log data into a compressed file.
3.3 Data Pre-processing

The data collected in the NetFlow is based on the flow-oriented. In order to transform the raw data from the flow-oriented into link-oriented, the flows are aggregated into a link with the same source IP and destination IP in each log file (five minutes). We focus on Layer-3 information rather than Layer-4 information. For example, Figure 3.3 shows the transformation of the raw data from the original flow-oriented into the link-oriented. There are six flows in the Figure 3.3 (a), and among them, there are two links with the same source IP and destination IP, from 140.117.11.1 to 140.117.190.135 and from 203.187.15.241 to 140.117.199.108. The six flows are aggregated into the two links where the bytes and the packets are...
accumulated from the original flow, as shown in Figure 3.3 (b). Each link is stored into the database with such scheme as \textit{Link\{Transaction ID, Source IP, Destination IP, Bytes\}}, where the transaction ID is the time that the cflowd dump the log data. For example, the transaction ID 12150015 means that the link is gathered at 00:15 on December 15, 2001.

<table>
<thead>
<tr>
<th>NO</th>
<th>Src_IP</th>
<th>Src_Port</th>
<th>Dst_IP</th>
<th>Dst_Port Packets</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>140.117.11.1</td>
<td>53</td>
<td>140.117.190.135</td>
<td>2086</td>
<td>512</td>
</tr>
<tr>
<td>2</td>
<td>140.117.11.1</td>
<td>53</td>
<td>140.117.190.135</td>
<td>2084</td>
<td>252</td>
</tr>
<tr>
<td>3</td>
<td>140.117.11.1</td>
<td>53</td>
<td>140.117.190.135</td>
<td>2083</td>
<td>218</td>
</tr>
<tr>
<td>4</td>
<td>203.187.15.241</td>
<td>1755</td>
<td>140.117.199.108</td>
<td>1097</td>
<td>204</td>
</tr>
<tr>
<td>5</td>
<td>203.187.15.241</td>
<td>2568</td>
<td>140.117.199.108</td>
<td>1084</td>
<td>974</td>
</tr>
<tr>
<td>6</td>
<td>203.187.15.241</td>
<td>1755</td>
<td>140.117.199.108</td>
<td>1097</td>
<td>204</td>
</tr>
</tbody>
</table>

(a). The Flow-oriented Raw data

<table>
<thead>
<tr>
<th>NO</th>
<th>Src_IP</th>
<th>Dst_IP</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>140.117.11.1</td>
<td>140.117.190.135</td>
<td>982</td>
</tr>
<tr>
<td>2</td>
<td>203.187.15.241</td>
<td>140.117.199.108</td>
<td>149525401</td>
</tr>
</tbody>
</table>

(b). The Link-oriented Raw data

Figure 3.3 Transforming Flow-oriented to Link-oriented Data Records

Because of the tremendous quantity of the network log data described in Section 3-2, and a great number of links not stronger enough to present the apparent interaction of two hosts, we filter out links with small traffic from the log data. We define the threshold measured in bytes for each link. The link is filtered out with the bytes smaller than the threshold. Table 3.1 shows the link number with different thresholds.
Table 3.1 The Number of Links with Different Flow Size Thresholds

<table>
<thead>
<tr>
<th>Duration Thresholds</th>
<th>12/1~12/7</th>
<th>12/8~12/14</th>
<th>12/15~12/21</th>
<th>Total Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 MBytes</td>
<td>473,243</td>
<td>399,826</td>
<td>694,853</td>
<td>1,567,922</td>
</tr>
<tr>
<td>0.5 MBytes</td>
<td>651,848</td>
<td>536,921</td>
<td>826,821</td>
<td>2,015,590</td>
</tr>
<tr>
<td>100 KBytes</td>
<td>1,467,022</td>
<td>1,103,689</td>
<td>1,682,148</td>
<td>4,252,859</td>
</tr>
</tbody>
</table>

3.4 LAG Mining

We adopt the LAG\textsubscript{2} algorithm described in Section 2.2 to discover the network access patterns in Dorm-Net. We choose one week as a cycle for the experiment to generate the access pattern and there are three weeks from December 1 to December 21. Table 3.1 lists the number of links with different flow size thresholds in these three weeks.

We define the minimum support rate as 1.2% for the test. The support rate means the ratio of the occurrence of connections to the number of the five-minute transactions. With the terms used in sequence mining, the connection or link between two hosts can be viewed as an item in the item set. There are 288 five-minute transactions in a day, 2016 transactions in a week. The support rate 1.2% equals 24 five-minute transactions in a week. The patterns, called the large graphs, are composed of the large item-sets. In the experiment, the large-4 graph, for example, means that there are four links appearing in the same transactions equal to or more than 24 times.
<table>
<thead>
<tr>
<th></th>
<th>1Mbytes</th>
<th>0.5M</th>
<th>100K</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>12/1 ~ 12/7</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LG1: 38</td>
<td>LG2: 16</td>
<td>LG3: 28</td>
<td>LG4: 15</td>
</tr>
<tr>
<td>LG3: 9</td>
<td>LG4: 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>12/8 ~ 12/14</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LG1: 35</td>
<td>LG2: 15</td>
<td>LG3: 17</td>
<td>LG4: 9</td>
</tr>
<tr>
<td>LG3: 10</td>
<td>LG4: 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LG5: 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>12/15 ~ 12/21</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LG3: 16</td>
<td>LG4: 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LG5: 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.2 shows the number of large graphs with the minimum support 0.12% for different thresholds. LG\textsubscript{n} donates the large-\textit{n} graph. For example, there are 38 large-one graphs, 16 large-two graphs, 9 large-three graphs, and 3 large-four graphs during December 1 to December 7 with the 1Mbytes threshold. Large graph is the pattern generated by LAG\textsubscript{2} algorithm from the NetFlow data. Large-one graph means one link (shared by two nodes), large-two graph means two links, large-three means three links, and so on. Figure 3.4 illustrates the large-two graphs.
3.5 Pattern Generation

The patterns generated by LAG2 are stored in the pattern base. The network administrator finds some interesting results from the pattern. The network access pattern can be viewed in machine operation or human action. Machine operation stands for the service that is automatically triggered by the program without the manpower. On the contrary, human action is the condition when people manually trigger the network transactions. With the pattern mined by LAG2, many patterns are discovered in machine operation because machine operation is more regular than human action. Besides, machine operation is prolonged and lasting in accessing the network resource, yet human action, however, is tired easily to keep doing the same thing in access the network resource more than two hours.
Table 3.3 The Special Case with Pattern Generation

<table>
<thead>
<tr>
<th>Machine operations</th>
<th>Human actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multicast services</td>
<td>BBS service</td>
</tr>
<tr>
<td>LFAP data collection</td>
<td>On-line game</td>
</tr>
<tr>
<td>News exchange</td>
<td>FTP service</td>
</tr>
<tr>
<td>Network management agent</td>
<td>Web service (Proxy)</td>
</tr>
<tr>
<td>MUD service</td>
<td></td>
</tr>
<tr>
<td>DNS queries</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.3 illustrates the interesting patterns discovered by the LAG$_2$. It lists six machine operations and four human actions. Many large-$n$ patterns may represent the same access pattern because not all of large-$n$ pattern with the same access pattern can be combined into large-$n$+1 pattern. For example, Figure 3.5 shows that two large-2 graphs represent the FTP service but they cannot be combined into a large-3 graph because the two large-2 graphs in the different transactions.

We elaborate instances of each access pattern listed in Table 3.3. These instances from various patterns are inspected by the FQDN query, ports speculation from the original log data. The large graph depends on the variation of network access conditions. One access graph may be contributed by several network services. For example, the transaction quantity in a large access graph may accumulate flow quantity of FTP and BBS services. The proxy service is seen as a human action because it is indirectly triggered by people to access the network resource.
Machine operations are listed as follows.

(1) Multicast. Figure 3.6 shows a large 5 graph to illustrate the multicast access condition. Although the IP, 140.117.18.10, does not belong to the Dorm-Net, the router forwards the packet for the multicast. The link is uni-direction, out through the 140.117.18.10 to 224.*.*.* (reserved for multicast).

(2) LFAP data collection. Figure 3.7 shows a large 8 graph to present the log data collection of the network. LFAP is a mechanism of the log data collection like NetFlow. The data keeps dumping to the host, 140.117.205.12, constantly and the flow is large enough over the threshold.

(3) News exchange. News server mutually exchanges the news frequently. The relationship shows in Figure 3.8. There are two news servers in the regional network, 140.117.205.10 and 140.117.201.52. They exchange a great number of the news data with others routinely.

(4) Network management agent. Two large graphs show in Figure 3.9 with the same host, 140.117.19.150. Although they are two different large-6 graphs, they

---

Figure 3.5 An Example: Two Large-2 Graph Cannot Combine into a Large-3 Graph

![Diagram](image_url)
represent the same access condition. They are the network management system, IBM NetView. Each link is bi-direction for the data exchange frequently.

(5) MUD service. The host, 140.117.201.101, works in the regional network with MUD as a regular service. MUD exchanges the data simultaneously like the news service, but the links are unidirectional which are different from the news service. Figure 3.10 shows the example of the MUD service.

(6) DNS query. Even though the DNS query happens all the time in the network, it is hard to be identified by LAG$_2$ for the bytes count of the flow smaller than the threshold. A special case shows in Figure 3.11. The host 140.117.11.6 is a famous BBS in NSYSU. It should look up the user’s host name all the time. Its DNS sets to 140.117.205.5 whose parent is 140.117.11.1. So they construct the large 4 graph.
Figure 3.6 Large-5 Graph – Multicast Service

Figure 3.7 Large-8 Graph – LFAP Data Collection

Figure 3.8 Large-8 Graph – News Exchange

Figure 3.9 Two Large-6 Graphs – IBM NetView

Figure 3.10 Large-5 Graph – MUD Service

Figure 3.11 Large-4 Graph – DNS Query
Human actions are listed as follows.

(1) BBS service. 140.117.11.6 is a well-known BBS service in NSYSU. The school announces news in the site all the time. There are more than 2000 thousand Dorm-Net users who may access BBS service, but the access graphs is small (i.e., small degree of connectivity) because there are few users who login and interact (over the threshold) at the same time frame. Figure 3.12 shows an instance of BBS service.

![Figure 3.12 Large-6 Graph – BBS service](image)

(2) On-line game. On-Line game can be categorized into a client-server mode and a peer-to-peer mode. Figure 3.13 (a) shows a large graph of the client-server mode such as the role-play game. There are three users in the subnet 181 logon the game server at the same time. Peer-to-peer on-line game such as real time strategic game does not exist a regular server. As Figure 3.13 (b) shows, only a host connects to the three hosts outside the Dorm-Net. The link in the large graph of the on-line game is usually bi-directional.
(3) FTP service. The large graphs of the FTP services, shown in Figure 3.14, are usually small because the users randomly use the FTP service instead of regularly continuous downloading. The link is usually bi-directional because the bytes count of the connection is huge and the TCP acknowledgement is relatively huge over the threshold.

(4) Web service (Proxy). Large graph of the web service usually is not big because the web service is not connection oriented and the users browse the different sites. A special case of the web service is proxy, shown in Figure 3.15, which is like the FTP service. Cyber-University is an on-line learning site. It provides teaching
materials for students to read on line. Besides the web transmission, it transfers the multimedia image as well. Figure 3.16 shows the large graph of the cyber-university connection.

![Figure 3.15 Large 3 Graph and Large 4 Graph– Proxy](image)

Figure 3.15 Large 3 Graph and Large 4 Graph– Proxy

![Figure 3.16 Large-5 Graph – Cyber University](image)

Figure 3.16 Large-5 Graph – Cyber University

LAG algorithm is a good mechanism to overview the network access conditions. With the large graph, we can identify the change of the network inter-connection in the different time periods. By the link-oriented view, frequent connections in the network can be discovered easily. The network administrator can utilize the large graph to comprehend the long-term access trend in the network.

The machine operation is more easily to be identified than the human action because the machine operation is regular, while the human action is random. People
are easily getting tired to sit in front of the computer accessing the network resource for hours. Therefore, the larger the support rate is, the less the access condition of the human action is discovered.

Because we define the threshold to filter out the links with small traffic, some state cannot be identified with LAG algorithm. For example, the virus attack is notorious in the network management domain. The virus generating a great number of the tiny flow leads the waste of the bandwidth and the heavily load to the system. The flow is so small that it is viewed as a noise and filtered out by the threshold. Some short-term states like the virus attack are not discovered easily using the LAG algorithm. Therefore, we propose a solution to detect the abnormal transactions with a short-term view in next chapter.
Chapter 4

Diagnosing Instant Networks Events with Case-Based Reasoning

Large access graph algorithm provides an effective mechanism to overview the regional network access patterns. It needs a long duration of the data and strong connection to construct the network interaction. It is necessary to develop a more efficient method to respond to short-term events instead of viewing long-term trends. In this chapter, we propose a case-based reasoning (CBR) method to facilitate the network management. CBR cycle introduced in the Subsection 2-3 presents a formal process dealing with the problem. We will follow the CBR cycle to diagnose achieve the instant network events.

4.1 Problem Statement

Large access graph algorithm discussed in Chapter 3 presents a whole picture of the network with the history log data. We view CBR as a means to quickly react to network traffic events, such as high volume of packet size, large quantity of flow connections, etc. Network usage steadily increases from day to day. The increase of applications on the network demands more and more bandwidth. The bandwidth cannot be expanded swiftly on demand since bandwidth is always not enough as people feel. Therefore, it is necessary to monitor and control the network traffic for the efficient utilization.

Many tools help the administrator to identify the network traffic with packets and bytes, which is based on the Layer-3 protocol. It is necessary to look up to the Layer-4 protocol to recognize the application. In general, to identify the application is via the port number. The well-known port has been integrated by many organizations such as Seifried (http://seifried.org/security/ports/index.html). Some
special applications using the particular port are also collected; for instance, the site http://www.u.arizona.edu/~trw/games/ports.htm collects the port information of the popular game.

Unfortunately, all the port numbers are not standard. Although we can observe port number easily from the log data, we are not definitely sure that the actual application. For example, FTP server usually uses the port 21 to communicate with clients. However, some unauthorized FTP services change the port number to others except port 21 to avoid to be specified easily. Notwithstanding, we can identify the application by certain features though the vary port number is different from the general one. For example, FTP server usually uses port 21 to transfer command and the uses port 20 to transfer data, where port 20 has larger flow size than port 20. Although port number changes from 21 to $n$, the port $n-1$ will accompany with the port $n$, and port $n-1$ has the larger flow.

Besides to monitor and to identify the application of the network utilization, the detection of abnormal behaviors can be also dealt. Some viruses, such as Trojan horse, use the security hole to intrude the system, and attack the host in the network, which waste the network bandwidth or crash the network equipments due to the heavy traffic. Port scanning is also an abnormal behavior in the network. It may lead to the security problem. The virus attack and the port scanning have the similar feature, such as a large number of flows (defined in the Sub-section 2.1). The feature can be extracted from the problem. By integrating these features, we can construct the case stored in the case-base for further comparison with similar conditions.

4.2 Data Processing

The log data collected from the NetFlow is used in this study to implement the
CBR system. We transform flows into IP sets \( R \rightarrow (IP_1, IP_2, \cdots, IP_n) \) where \( R \) denotes the raw log data and the \( IP_i \) denotes the specified IP, which construct the problem (will describe in the next section). Each problematic IP is composed of connections that are different from the links based on the Layer-3 protocol described in the Subsection 3.3. The connection is based on the Layer-4 including port numbers. Because this study focuses on the IP set, each flow generates two connections, one is from the source IP and the other is from the destination IP. That is, the connection is un-directional here. The connection is defined as \( \text{Connection} \{ \text{Ego IP, Ego Port, Actor IP, Actor Port, Flows, Bytes} \} \), where the \( \text{Ego IP} \) and \( \text{Ego Port} \) represent the IP and port from the problematic IP, while the \( \text{Actor IP} \) and \( \text{Actor Port} \) represent the IP and port interacting with the problematic IP. The Flows and Bytes store the quantity of the flows and the bytes.

The port number selecting is the key issue in this step. In general, a network server with the specified port connects to clients. The service port is identified with the scaling method. We give scale one to a port appearing once, additional ten scores to a port number with bytes greater than one Mbytes, and ten more scores to a port number with bytes greater than ten Mbytes (i.e., the port with bytes greater than ten Mbytes gets twenty-one scores). In general, port numbers below 1024 are used for the well-known service, and they usually obtain thirty scores. We ignore a port below thirty scores and replace it with an asterisk (*). Figure 4.1 shows an example of the transformation from the raw data to the problem with connections. Figure 4.1(a) is a fragment of the raw data from the NetFlow, and Figure 4.1(b) is the result of the transformation from Figure 4.1(a).
### (a). The Flow-oriented Raw data

<table>
<thead>
<tr>
<th>NO</th>
<th>Src_IP</th>
<th>Src_Port</th>
<th>Dst_IP</th>
<th>Dst_Port</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>203.187.15.241</td>
<td>1632</td>
<td>140.117.199.108</td>
<td>21</td>
<td>587</td>
</tr>
<tr>
<td>2</td>
<td>140.117.199.108</td>
<td>21</td>
<td>1203.187.15.241</td>
<td>1632</td>
<td>776</td>
</tr>
<tr>
<td>3</td>
<td>140.117.199.108</td>
<td>1433</td>
<td>140.117.11.1</td>
<td>53</td>
<td>107</td>
</tr>
<tr>
<td>4</td>
<td>140.117.11.1</td>
<td>53</td>
<td>140.117.199.108</td>
<td>1433</td>
<td>156</td>
</tr>
<tr>
<td>5</td>
<td>203.187.15.241</td>
<td>2568</td>
<td>140.117.199.108</td>
<td>20</td>
<td>1,635,249</td>
</tr>
<tr>
<td>6</td>
<td>140.117.199.108</td>
<td>20</td>
<td>1203.187.15.241</td>
<td>2568</td>
<td>53,218</td>
</tr>
<tr>
<td>7</td>
<td>203.187.15.241</td>
<td>2569</td>
<td>140.117.199.108</td>
<td>20</td>
<td>15,698,214</td>
</tr>
<tr>
<td>8</td>
<td>140.117.199.108</td>
<td>20</td>
<td>1203.187.15.241</td>
<td>2568</td>
<td>669,884</td>
</tr>
</tbody>
</table>

### (b). The Connection-oriented Representation

<table>
<thead>
<tr>
<th>Problem IP</th>
<th>Ego IP</th>
<th>Ego Port</th>
<th>Actor IP</th>
<th>Actor Port</th>
<th>Flows</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>203.187.15.241</td>
<td>203.187.15.241</td>
<td>*</td>
<td>140.117.199.108</td>
<td>21</td>
<td>2</td>
<td>1,363</td>
</tr>
<tr>
<td></td>
<td>203.187.15.241</td>
<td>*</td>
<td>140.117.199.108</td>
<td>20</td>
<td>4</td>
<td>17,333,463</td>
</tr>
<tr>
<td>140.117.199.108</td>
<td>140.117.199.108</td>
<td>21</td>
<td>203.187.15.241</td>
<td>*</td>
<td>2</td>
<td>1,363</td>
</tr>
<tr>
<td></td>
<td>140.117.199.108</td>
<td>20</td>
<td>203.187.15.241</td>
<td>*</td>
<td>4</td>
<td>17,333,463</td>
</tr>
<tr>
<td></td>
<td>140.117.199.108</td>
<td>*</td>
<td>140.117.11.1</td>
<td>53</td>
<td>2</td>
<td>263</td>
</tr>
<tr>
<td>140.117.11.1</td>
<td>140.117.11.1</td>
<td>53</td>
<td>140.117.199.108</td>
<td>*</td>
<td>2</td>
<td>263</td>
</tr>
</tbody>
</table>

Figure 4.1 Transformation from Raw Data to Connection-oriented Records

### 4.3 Case Representation

One essential part of a CBR system is a library of stored cases. Each case in the library can be identified by a set of features (attributes). These features constitute the problem domain of the cases. In addition, each case is associated with a label, called state description in this study. More formally, if \((F_1, F_2, \cdots, F_n)\) denotes a set
of features for the cases and $S$ the variable corresponding to the state description, a particular case can be represented in a form $n+1$-dimensional vector $e_i = (x_{i1}, x_{i2}, \cdots, x_{im}; y_i)$ where $x_{ij}$ corresponds to the value of $F_j$, and $y_i$ is the state description $S$ for this particular case.

In this study, we give a new problem $e$ and request the system to determine an appropriate state description for the new problem based on the library of historical cases. The new problem is represented by $e = (f_1, f_2, \cdots, f_n)$ in which $f_j$ corresponds to the value of feature $F_j$. The system will use the vector $(f_1, f_2, \cdots, f_n)$ to search the case library for the similar cases. The state descriptions of these similar cases are then used to determine a label for the new problem. When a new problem associates with a label, then it becomes a case.

In practice, we construct the problem from the NetFlow raw data. The problem is formulated as three part, problem IP, connections, and features. The problem IP records the IP of the problem to be resolved. The connections store the connections introduced in the Subsection 1.2. The features record the extractions from the connections, where its scheme is Feature $\{\text{Port, Flows, Bytes}\}$. Figure 4.2 shows an example of the structure of the problem with the IP, 140.117.92.207. Moreover, the connections are the use in the case repair step to enlarge the case library, and the features are used to case similarity in the case retrieval step.
### Problem IP

**IP:** 140.117.92.207

#### Connections

*Connection:* 140.117.92.207: * - 140.117.197.59: 20

*Flows:* 80  
*Bytes:* 40679652

*Connection:* 140.117.92.207: * - 140.117.197.59: 21

*Flows:* 4  
*Bytes:* 25697

#### Features

*Ego_Port:* *  
*Flows:* 84  
*Bytes:* 40705349

*Actor_Port:* 20  
*Flows:* 80  
*Bytes:* 40679652

*Actor_Port:* 21  
*Flows:* 4  
*Bytes:* 25697

---

**Figure 4.2 Structure of a Problem**

---

### Case Name

**FTP**

#### Features

*Actor_Port:* 20  
*Flows:* Ignored  
*Bytes:* Large

*Actor_Port:* 21  
*Flows:* Ignored  
*Bytes:* Ignored

#### State Description

A FTP Client Connecting with Port Number 21

---

**Figure 4.3 The FTP Client Case**

---

A case stored in the case-base has three general parts, case name, features, and state description, where Figure 4.3 illustrates an instance of the FTP case. The case name records the name of the case and the features is similar with the feature of the problem mentioned above. A problem is compared with the cases in the case-base and the similar case is retrieved to label the state description on the problem, then the problem is solved. Figure 4.4 shows a solved problem, which is the problem in Figure 4.2 similar with the case in Figure 4.3.
<table>
<thead>
<tr>
<th>Case Name</th>
<th>FTP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Features</strong></td>
<td></td>
</tr>
<tr>
<td>Ego Port: *</td>
<td>Flows: 84</td>
</tr>
<tr>
<td>Actor Port: 20</td>
<td>Flows: 80</td>
</tr>
<tr>
<td>Actor Port: 21</td>
<td>Flows: 4</td>
</tr>
<tr>
<td><strong>State Description</strong></td>
<td>A FTP Client Connecting with Port Number 21</td>
</tr>
</tbody>
</table>

Figure 4.4  A Solved Problem with State Description

4.4 Case Retrieval

The process of the case retrieval is to select the similar cases in the case-base with the problem. The similarity function is defined as

\[
\text{Similarity}(P_m, C_n) = \frac{\sum W_j \times (1 - \text{dis}(f_{i_{\text{problem}}}, f_{i_{\text{history}}}))}{\sum W_j},
\]

where \( P_m \) denotes the problem to resolve, \( C_n \) denotes the case stored in the case-base, \( W_j \) denotes the weight of the feature in the case, and the function \( \text{dis}() \) denotes the measure of the distance between the problem and the case. In the distance function, the degree of the large quantity can be defined with the fuzzy logic. In our evaluation discussed in Chapter 5, we define it by dividing the quantity of the specified port by the total quantity of the IP set.

After the degree of the similarity between the problem and the case is determined, we define a threshold to allocate the case to retrieve. If the similarity degree is above the threshold, the state description of the case is labeled on the problem. Then,
the problem is resolved. In this study, a problem may be labeled with more than one state description. For example, a FTP server providing a FTP service may be infected with the virus to attack others as well.

4.5 Case Repair and Case Retain

After the problem is resolved, the administrator can identify the access condition of the specified host. However, not all the host can be identified by the CBR system. The administrator should observe the problem, label it, and determine the weight of the feature. Besides the problem is not identified, the problem may get a wrong label from the CBR system. The administrator should also revise the case to strengthen the case base. For example, Figure 4.5 illustrates a new case, which the administrator observes and the weights of features are determined.

<table>
<thead>
<tr>
<th>Case Name</th>
<th>DNS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Features</strong></td>
<td><strong>State Description</strong></td>
</tr>
<tr>
<td><em>Ego Port: 53</em></td>
<td><em>A DNS server with port number 53</em></td>
</tr>
<tr>
<td>Weight: 0.5</td>
<td><em>Flows: Large</em></td>
</tr>
<tr>
<td>Weight: 0.4</td>
<td>Weight: 0.1</td>
</tr>
<tr>
<td><em>Bytes: Large</em></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.5 The DNS Service Case with Weights of Features
Chapter 5 Evaluation and Results

We implemented the CBR method by building case base from the NetFlow data collected from the core router at the NSYSU Dorm-Net. From the core router, we identify user connection history and build the case library following the CBR cycle. The data of network traffic is collected from November 1 to 21, 2001. We take two weeks data from November 1 to November 14 for training in order to construct the case library, and then data from November 15 to 21 is used for evaluation. These cases in the library are listed in Appendix A. The cflowd collects the NetFlow data each five minutes. We aggregate six five-minute data as a transaction.

The schema of the flow record is Source IP, Destination IP, Source Port, Destination Port, and Bytes. We transform flows into unique connections with port number. The bytes and the flows are counted with the identical port for the separated IPs. The IP set is discarded if the total bytes of an IP host are less than 2048 bytes. Finally, the IP set is selected if the rank of total flows is in front of five percent or the rank of total bytes is in front of five percent. The degree of the large quantity is obtained by dividing the quantity of the specified port by the total quantity of the IP set. In the peak time, there are more than 20,000 IPs in a transaction and less than 4,000 IPs of the total bytes greater than 2048 bytes caused by virus attack spraying in the network. In the non-peak time, there are around 2,000 IPs in a transaction and the total bytes are greater than 2048 bytes. After selecting hosts based on the ranking data, there is only left 100 to 200 IPs.

CBR method is evaluated by comparing the result of the CBR system with experts’ judgment. Three criteria are used to evaluate, precision, recall, and accuracy. They are defined as precision = \( \frac{A \cap A'}{A'} \), recall = \( \frac{A \cap A'}{A} \), and accuracy = \( \frac{A \cap A'}{R} \).
where $A$ denotes the number of the IPs which the network administrator can label their problems, $A'$ denotes the number of IPs which CBR system labels their state description, and $R$ denotes the number of the total IPs in the transactions. Figure 5.1 shows the relationship between $R$, $A$, and $A'$.

![Figure 5.1 The Relationship between $R$, $A$, and $A'$](image)

Table 5.2 The Evaluation Results in Precision, Recall, and Accuracy

<table>
<thead>
<tr>
<th>Date</th>
<th>11/15</th>
<th>11/16</th>
<th>11/17</th>
<th>11/18</th>
<th>11/19</th>
<th>11/20</th>
<th>11/21</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>11,982</td>
<td>10,368</td>
<td>10,783</td>
<td>12,698</td>
<td>13,697</td>
<td>10,481</td>
<td>12,368</td>
<td>11,768</td>
</tr>
<tr>
<td>$A'$</td>
<td>9,426</td>
<td>9,248</td>
<td>9,045</td>
<td>9,942</td>
<td>10,390</td>
<td>9,039</td>
<td>9,531</td>
<td>9,517</td>
</tr>
<tr>
<td>$A \cap A'$</td>
<td>9,237</td>
<td>9,031</td>
<td>8,753</td>
<td>9,632</td>
<td>9,736</td>
<td>8,723</td>
<td>9,232</td>
<td>9,192</td>
</tr>
<tr>
<td>$R$</td>
<td>13,589</td>
<td>12,823</td>
<td>13,364</td>
<td>14,270</td>
<td>15,274</td>
<td>12,672</td>
<td>14,892</td>
<td>13,841</td>
</tr>
<tr>
<td>Precision</td>
<td>0.98</td>
<td>0.98</td>
<td>0.97</td>
<td>0.97</td>
<td>0.94</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
</tr>
<tr>
<td>Recall</td>
<td>0.77</td>
<td>0.87</td>
<td>0.81</td>
<td>0.76</td>
<td>0.71</td>
<td>0.83</td>
<td>0.75</td>
<td>0.79</td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.68</td>
<td>0.70</td>
<td>0.65</td>
<td>0.67</td>
<td>0.64</td>
<td>0.69</td>
<td>0.62</td>
<td>0.66</td>
</tr>
</tbody>
</table>

We accumulate 48 transactions in a day, and compute precision, recall, and
accuracy. The results are shown in Table 5.1

The precision is high in our study. The state description proposed by CBR system may be an incompatible situation. For example, one may be identified as retrieving the web service heavily and the virus attack host, but it is actually the virus attack only. Figure 5.2 shows the evaluation results measured by precision.

![Figure 5.2 The Evaluation Results Measured by Precision](image)

Figure 5.2 The Evaluation Results Measured by Precision

![Figure 5.3 The Evaluation Results Measured by Recall](image)

Figure 5.3 The Evaluation Results Measured by Recall
Figure 5.3 shows evaluation results measured by recall. The recall value is not as high as the precision because there are still some situations that cannot be formulated as the case representation easily, some situation cannot be figured out in the CBR system. For example, if the application uses the complex port to transfer data over port 1024, it may only be represented as the asterisk “*”.

![Figure 5.3](image)

Figure 5.3 The Evaluation Results Measured by Recall

Figure 5.4 shows the evaluation results measured by accuracy. The accuracy means the possibility of the CBR system proposing the result without the administrator. The accuracy is 66% in average. In reality, some special cases cannot be distinguished even by an administrator. The CBR system proposes the effective means to help the administrator to identify the network access conditions.

From the evaluation results, we found that

(1) The precision rate is high, which implies the results proposed by the CBR system are trustworthy. The administrator can use this system to identify hosts which contribute abnormal traffic to the network.
(2) In some situations, cases cannot be constructed easily because ports of the services are not apparent to identify, especially when the port identifier is above 1024. That is, port identifiers which are above 1024 may be marked with asterisk “*”. The real situation of the connection IP₁ with port * to IP₂ with port * is hard to be identified. That is the reason that the recall rate is not as high as the precision rate.

(3) The administrator domain knowledge is unable to be replaced. CBR can solve the problem with the past experience, but, new application not found yet in the case library cannot be labeled without domain expertise held by network administrators.
Chapter 6 Conclusions and Feature Research

In this study, we have applied the Large Access Graph (LAG) algorithm to discover the Dorm-Net access patterns of National Sun Yat-Sen University, and employed the case-Based reasoning (CBR) method to diagnose network dynamics. The router log data, NetFlow, comprises the network traffic data for us to generalize relationship among network transactions, and to detect instant network events.

The NetFlow data is transformed from flow-oriented into link-oriented records based on Layer-3 of the TCP/IP model. We employed LAG algorithm to find various link numbers of large access graphs. The access patterns discovered by LAG algorithm focus on how often users interact above a certain quantity of traffic size instead of only accumulating traffic size. In elaborating network access patterns discovered from NSYSU Dorm-Net, the network access patterns are categorized into human action and machine operation. The access patterns of from machine operation are discovered more easily than human action. The large graphs construct the supply and demand relationship of the popular services. The large graphs are also discussed for the popular services.

Unlike LAG algorithm, CBR method we implemented uses the connection-oriented records based on Layer-4 protocol of the TCP/IP model. The port number is taken into consideration to identify the network applications. We follow CBR cycle for diagnosing instant network events. First, we transform the NetFlow log data into the connection-oriented records, and then the problem is constructed. Second, the problem matches with the cases stored in the case base, and the matched case is retrieved to solve the problem. Finally, the resolved problem is verified and revised as a new case to insert into the case base.
From the evaluation results in recall, precision, and accuracy using CBR method, we found that CBR method could precisely (97% precision rate) retrieve cases with about 79% recall rate to predict about 66% network events in testing network events occurring in 7 days. These results effectively support the use of the CBR approach for network event diagnosis.

There is still advance development of our study. In future works, the following works can be further developed.

(1) To include the port number into the LAG analysis to enrich the discovered access patterns. In this study, we only consider the source IP, destination IP, and the bytes count, but we do not consider the port number. For the administrator, it would be worthy to analyze the port number information in identifying network access patterns.

(2) The similarity of the case retrieval can be implemented with fuzzy logic in the CBR method. Some linguistic terms, such as large bytes count, can be calculated by fuzzy set. According to fuzzy membership functions, the problem becomes that of finding the degree of matching two fuzzy sets. With the fuzzy reference, it may obtain better accuracy in retrieving similar cases.

(3) Accounting and billing is also a primary domain in the network management. The special accounting or billing mechanisms can be designed as a case representation to do the further selection, especially in the complex and the fast changing situation.
References


CAIDA, cflowd: Traffic Flow Analysis Tool,


Chwastek, R., Network Management in Wired and Wireless Networks: Book Review


Dunkel, B. and Soparkar, N., Data Organization and Access for Efficient Data Mining,
    ICDE, Sydney, Australia, 1999.

Fayyad, U., G. Piatesky-Shapiro, and P. Smyth, From Data Mining to Knowledge
    Discovery: an Overview, in Advances in Knowledge Discovery and Data Mining
    Chapter 1, AAAI/ The MIT Press, 1996.

Kolodner, J., Case-Based Reasoning, Morgan Kaufmann Publishers, pp.145-160,
    1993.

Liao T. W., Zhang Z., Mount, C. R., Similarity Measures for Retrieval in Case-based

McRobb, D. W., cflowd design,

Park, J. S., Chen, M. S., and Yu, P. S., An Effective Hash Based Algorithm for Mining

Piatesky-Shapiro, G. and W. J. Frawley, Knowledge Discovery in Databases, The

Seifried, K., Ports Report for Applications,


Williams, T., Port Used by Computer Game,

Appendix A

1. The scheme of the case base.

<table>
<thead>
<tr>
<th>ID</th>
<th>Feature</th>
<th>Port</th>
<th>Weight₁</th>
<th>Flows</th>
<th>Weight₂</th>
<th>Bytes</th>
<th>Weight₃</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>State Description</td>
<td>/* Description for cases */</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. The case library constructed with the log data from December 1, 2001 to December 14, 2001.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Port</th>
<th>Weight₁</th>
<th>Flows</th>
<th>Weight₂</th>
<th>Bytes</th>
<th>Weight₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>State Description</td>
<td>MS-Windows Network Neighborhood File Sharing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State Description</td>
<td>FTP-PASV Mode with port 21 as command port</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State Description</td>
<td>Squid PROXY HTTP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State Description</td>
<td>Domain Name Server</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State Description</td>
<td>World Wide Web HTTP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State Description</td>
<td>Simple Mail Transfer Protocol</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State Description</td>
<td>Post Office Protocol - Version 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State Description</td>
<td>Telnet (BBS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State Description</td>
<td>FTP-PORT Mode with port 21 as command port</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State Description</td>
<td>Napster Peer to Peer File Sharing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State Description</td>
<td>Game-Battle Net</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State Description</td>
<td>MS-Windows Network Neighborhood File Sharing (Win2000)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Feature</td>
<td>ICMP 0.5</td>
<td>Peer to Peer File sharing</td>
<td>Viruses Attack (Nimda)</td>
<td>Media Service</td>
<td>FTP Service</td>
</tr>
<tr>
<td>---</td>
<td>---------</td>
<td>----------</td>
<td>-------------------------</td>
<td>------------------------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>13</td>
<td>Feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>* 0.333</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>