IEEE International Conference on Communications, **ICC-2001**
IEEE Catalog No. 01CH37240, Library of Congress 81-649547
©IEEE, June 2001

Visualization of Traffic Structures

Oliver Niggemann and Benno Stein
Dept. of Mathematics and Computer Science
University of Paderborn, Germany
Email: {niggemann, stein}@uni-paderborn.de

Jens Tölle
Institute of Computer Science IV
University of Bonn, Germany

Abstract— **An in-depth analysis of traffic data provides valuable insights into network traffic structures. This paper presents both methods for such an in-depth analysis and their implementation within the system** STRUCTUREMINER**. Moreover, it is shown in which way the analysis performed by** STRUCTUREMINER **can be used to tackle several administrative network tasks.**

Keywords— **network analysis, traffic analysis, graph drawing, clustering**

I. INTRODUCTION

This paper is devoted to the traffic analysis in computer networks. Existing tools and approaches to network traffic analysis apply a desriptive statistical approach: Traffic data is measured at various places, and the amount and the distribution of the data is presented in the one or other type of bar charts.[1]

Compared to the information that could be extracted from the available data (e. g. packet traces using the RF 1761 snoop format), this analysis is rather superficial. In fact, applying an in-depth analysis, significant structures can be identified within the traffic data. Moreover, these structures can be visualized and used for several high-level analysis, such as

- the recognition of cooperating users,
- the animation of traffic flows,
- the analysis of network topologies, or even
- intrusion detection.

Such an in-depth analysis can be realized by the combination of clustering methods, concepts from case-based reasoning (CBR), and graph drawing algorithms. Fig. 1 shows the steps involved.

The physical network is abstracted towards its traffic graph G, defined below. In a second step clustering is used as a powerful preprocessing for recognition and visualization: G is analyzed in order to detect, isolate, or emphasize particular structures. In a third step these structures as well as the nodes within these structures are arranged. At this point may draw up knowledge-based methods that interpret the clustering and layout results, e. g. with respect to an automatic network reconfiguration or automatic intrusion detection.

The system STRUCTUREMINER has been developed to support structure identification in technical systems. Section II gives an introduction to the system and its underlying

[1]The tools CINEMA (Hirschmann), TRENDREPORTER (NetScout Systems), NETWORKAUDIT (Kaspia Systems), OPTIVITY (Bay Networks), OBSERVER (Network Instruments), or SURVEYER (Shomiti) are representatives of this approach. A non-commercial system has been developed by [8], [7].

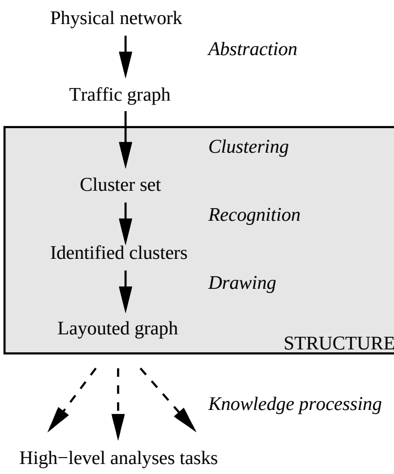


Fig. 1. Processing steps within a high-level analysis of traffic data. The shaded area indicates the steps automated by the STRUCTUREMINER system.

methods related to network and traffic analysis. Section III then presents three high-level analysis tasks and shows how they can be tackled by our approach. The remainder of this section provides related definitions.

Fig. 2 shows on the left a network from its physical setup (see also Fig. 12). The nodes in this network form a set V, the physical lines (cables) form a set of edges. The closure of this graph with respect to accessibleness yields the graph (V, E); it contains an edge between each two nodes that are physically connected (see Fig. 2, right-hand side).

Given such a completely connected graph (V, E) of a physical network, the related traffic matrix can be interpreted

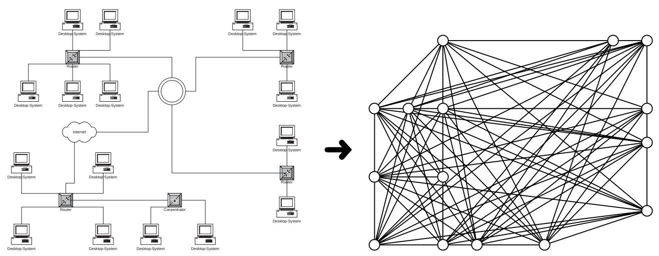


Fig. 2. A small network from its physical setup (left-hand side) and the related traffic matrix, shown as (weighted) graph (on the right).

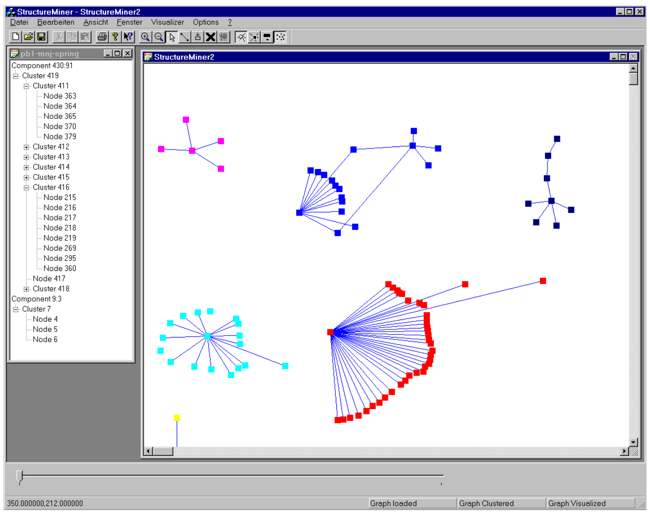


Fig. 3. Using STRUCTUREMINER for Visualizing Network Traffic.

as a set of edge weights. In particular, the traffic matrix defines a mapping $\omega : E \to \mathbf{R}^+$, it assigns each edge $e \in E$ a number that specifies the amount of traffic between the nodes that are adjacent to e. This weighted graph, $G = (V, E, \omega)$ is called traffic graph and forms the base for subsequent clustering and drawing steps.

Definition 1 (Clustering) *A clustering $\mathcal{C} = \{C \mid C \subseteq V\}$ of a graph $G = (V, E, \omega)$ is a division of V into sets for which the following conditions hold: $\bigcup_{C_i \in \mathcal{C}} C_i = V$, and $\forall C_i, C_j \in \mathcal{C} : C_i \cap C_{j \neq i} = \emptyset$. The induced subgraphs $G(C_i)$ are called clusters. $E_{\mathcal{C}} \subseteq E$ consists of the edges between the clusters.*

Our clustering and graph drawing methods are of a generic type and can be applied to all graphs of the form $G = (V, E, \omega)$. The clustering process assigns each node $v \in V$ a cluster number; the recognition process assigns two clusters a similarity value; the drawing process assigns each node $v \in V$ a point in the Euclidean plane.

Two design goals are pursued during graph drawing:
1. The error of a misclassification is to be minimized. Let $\epsilon(i, j) \geq 0$ denote the misclassification penalty for a node v that has mistakenly been assigned to cluster C_j instead to cluster C_i, whereas $\epsilon(i, i) = 0$. Then $\sum_{C_j \in \mathcal{C}} \sum_{v \in C_j} \epsilon(i, j)$ shall be minimum.
2. The nodes in the clusters as well as the clusters itself are to be drawn clearly.

II. THE SYSTEM STRUCTUREMINER

The software system STRUCTUREMINER employs the visualization strategy described in Section I. Traffic graphs or traffic traces (snoop format, RFC 1761) can be imported, the graph is clustered, the clusters are classified and the graph is finally layouted.

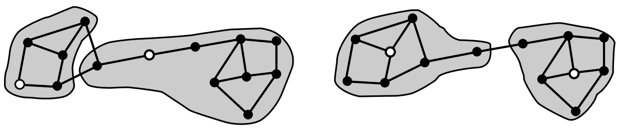


Fig. 4. Kohonen Clustering: Step 1 (left-hand side) and step 2 (right-hand side).

Fig. 3 shows the surface of STRUCTUREMINER: The structure of the graph is presented in the tree view on the left-hand side of the screenshot, while the graph itself is rendered in the graph view on the right-hand side.

A. Clustering

Clustering is one of the oldest areas in Machine Learning, early work dates back from 1894 (see [18]). Three typical approaches have been implemented in STRUCTUREMINER and can be applied to the traffic graph.
1. *MinCut Clustering.* This method subdivides a graph $G = (V, E, \omega)$ recursively at its minimum cut.[2] Since this method would always result in $|V|$ clusters, each containing exactly one node, it also has to be defined when a cluster should not be subdivided anymore: STRUCTUREMINER only subdivides a cluster when the division would result in an improvement of the so-called λ-value of the clustering (see [20], [6]).
2. *K-Means or Kohonen Clustering.* This method defines a clustering of a graph $G = (V, E, \omega)$ implicitly by centroid nodes: Each node belongs to its closest centroid node. Initially $m, m < |V|$ centroid nodes are chosen randomly from $|V|$ (see Fig. 4). Iteratively each centroid node moves into the center of its cluster (Fig. 4, left-hand side). The algorithm stops, when the positions of all centroid nodes have stabilized. Details can be found in [1], [12], [16].
3. *MajorClust.* This method has been especially developed for STRUCTUREMINER by Stein and Niggemann (see [20], [17]). Initially, the algorithm assigns each node of a graph its own cluster. Within the following agglomerative re-clustering steps, a node adopts the same cluster as the majority of its neighbors belong to. If there exist several such clusters, one of them is chosen randomly. If re-clustering comes to an end, the algorithm terminates.

[2]The minimum cut of a graph is the smallest set of edges, whose removal would divide the graph into two not connected components (see also [9]), details can be found in [21], [15].

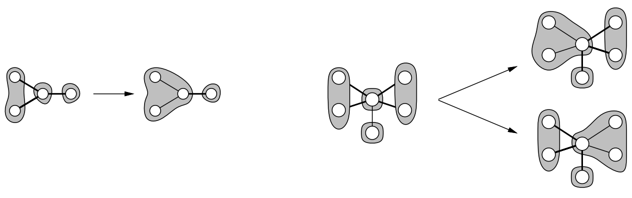
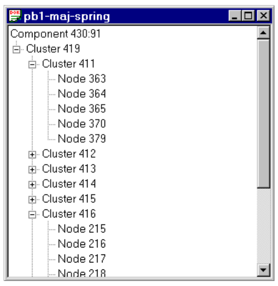


Fig. 5. A definite majority clustering situation (left) and an undecided majority clustering situation (right).


pb1-maj-spring

Component 430:91
- Cluster 419
  - Cluster 411
    - Node 363
    - Node 364
    - Node 365
    - Node 370
    - Node 379
  - Cluster 412
  - Cluster 413
  - Cluster 414
  - Cluster 415
  - Cluster 416
    - Node 215
    - Node 216
    - Node 217
    - Node 218

Fig. 6. The Tree View of STRUCTUREMINER.

The left-hand side of Fig. 5 shows the definite case; most of the neighbors of the central node belong to the left cluster, and the central node becomes a member of that cluster. In the situation depicted on the right-hand side, the central node has the choice between the left and the right cluster.

After the clustering, the detected graph structure is shown in the tree view (see Fig. 6). On the top level the connected components of the graph can be seen, below each connected component the individual clusters are arranged. If subclusters exist[3], they can be found below their parent cluster. The nodes form the leafs of the structure tree.

B. Recognition

The administrator should be able to label clusters, typical labels are e.g. "Project X" or "AI Lab". When a new cluster is identified, it should automatically be given the same label as the most similar cluster encountered before, i.e. the clusters are classified using a case-based approach (see [13], [14]).

In order to find a similar cluster, a so-called similarity measure is used. Here the similarity measure is a function $sim : \mathcal{P}(V) \times \mathcal{P}(V) \to \mathbf{R}$, which measures the similarity between two clusters. When a new cluster is found, it is compared to all previously labeled clusters using the similarity function. The label of the previous cluster with the highest similarity (i.e. the highest value of the similarity function) is used for the new cluster.

For network administration purposes the following similarity measure makes sense:

Definition 2 (Cluster Similarity) *Let C_1 and C_2 be two clusters of the graph $G = (V, E, \omega)$, i.e. $C_1, C_2 \subseteq V$ (C_1 and C_2 need not be disjunctive). Then the similarity function sim between two clusters is defined as:*

$$sim(C_1, C_2) = |C_1 \cap C_2|$$

C. Graph Drawing

STRUCTUREMINER uses graph drawing methods to place clusters and to position nodes within clusters. For this, two common graph layout methods have been implemented and can be applied by the user to the clustered traffic graph.

[3]All three clustering methods are able to detect further subclusters within cluster.

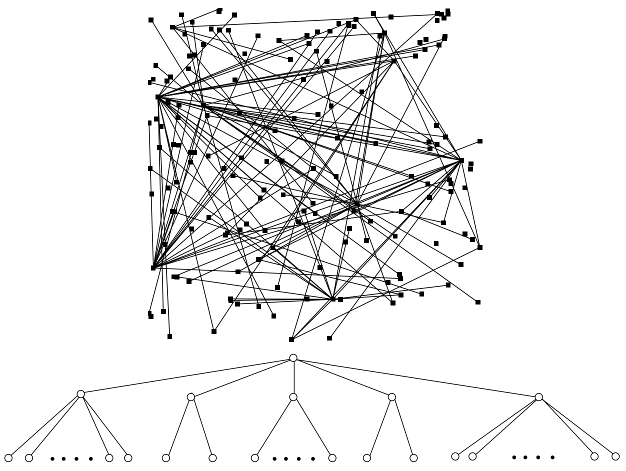


Fig. 7. A randomly drawn real-world traffic graph (top) and its clustering (bottom).

1. *Spring-Embedder.* The spring embedder method (see [4], [11], [5]) relys on information about the optimal distances between connected vertices. As long as the distance between two connected vertices exceeds this length, the vertices attract each other. Otherwise the force becomes repelling. By iteratively applying these forces, an equilibrium, i. e. a layout, is reached.
2. *Hierarchical-Graph-Drawer.* An hierarchical graph drawer (see [10]) subdivides the vertices into vertical layers. Edge crossings are minimized in a second step by ordering the vertices within the layers.

STRUCTUREMINER does not only draw the graphs, but it also combines the clustering information with the drawing methods. This leads to a graph layout, which emphasises the graph's inherent structure and which supports the human understanding of complex traffic graphs.

This method is now illustrated using an example: The graph at the top of Fig. 7 depicts a random layout of a small

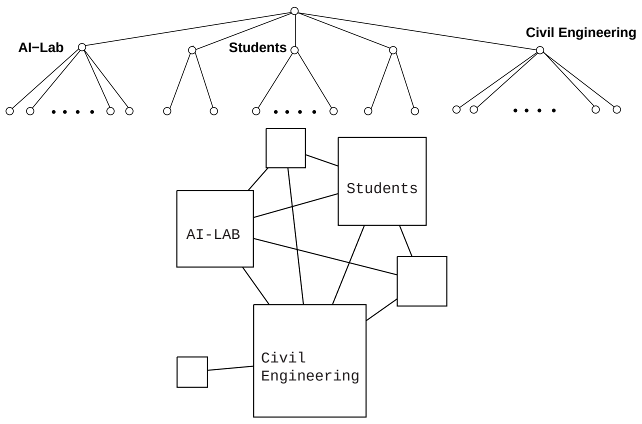


Fig. 8. Example at step 3 (top) and example at step 4 (bottom).

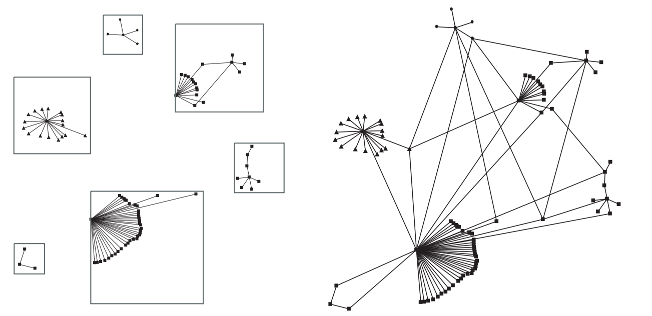


Fig. 9. Example at step 5 (left-hand side) and example at step 6 (right-hand side).

traffic graph. The user decides to cluster this graph using MAJORCLUST. The result can be seen at the bottom of Fig. 7. Three clusters could be classified using a list of already known clusters, Fig. 8 (top) shows the result. In the next step, STRUCTUREMINER layouts the clusters (see Fig. 8, bottom), each cluster is represented by a bounding box. In this example a spring embedder algorithm has been used to position the bounding-boxes. This abstract view onto the graph is called the structure mode of STRUCTUREMINER. Cluster labels are only shown in this visualization mode. STRUCTUREMINER now draws the nodes within each cluster (Fig. 9, left-hand side). The right-hand side of Fig. 9 shows the final result.

The reader may note that node positions within the clusters must also take the general cluster layout into consideration: A node in cluster A, that is also connected to another node in cluster B, should be placed as close as possible to cluster B. STRUCTUREMINER extends existing layout methods to allow for this.

Further features of STRUCTUREMINER are:
- For each cluster it is possible to choose the drawing method individually, i.e. the cluster “students” could be visualized using a spring-embedder and the cluster “AI-Lab” could use an hierarchical-layout method. The new drawing method can be applied recursively to the subclusters, too.
- By using a slider control, edges with weights below a given threshold can be blinded out. This allows for the concentration on important edges.
- In a separate file, the user can associate name templates with colors, i.e. node names matching the regular expression “128.234.*.*” (e.g. 128.234.28.34) may be associated with the color red. Nodes are visualized using the appropriate color.
- Zooming of the graph.
- All results are cached for later use.
- Several graphs can be visualized at the same time.
- Graphs can be printed and exported (e.g. into the wmf-format).
- Nodes and edges can be added, deleted, and moved, i.e. STRUCTUREMINER works also as a network editor.

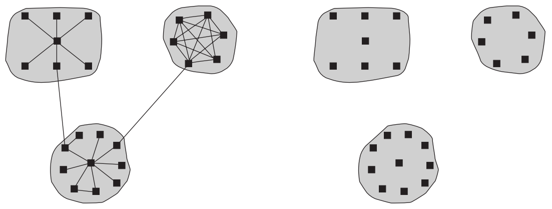


Fig. 10. Traffic Animation: Initial clustering with edges (left-hand side) and without edges (right-hand side).

III. SPECIAL NETWORK APPLICATIONS

Besides the visualization of network traffic, STRUCTUREMINER has been extended and applied to three closely related problems: The visualization of temporal change of network traffic (Section III-A), the analysis of network topologies (Section III-B), and the detection of intrusions (Section III-C). The traffic used has been recorded at the universities of Paderborn and Bonn.

A. *Traffic Animation*

The administrator is highly interested in understanding the change of traffic during a day, a week or a year. Since the amount of network traffic makes a quantitative presentation impossible, a qualitative visualization of traffic change has to be found. For this the visualization technique described above can be extended as follows:

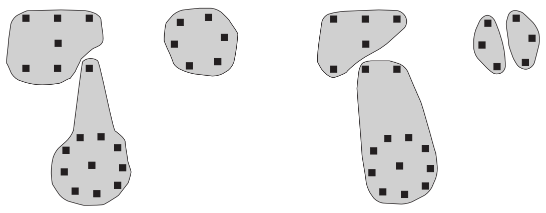


Fig. 11. Traffic Animation: Step 2 (left-hand side) and step 3 (right-hand side).

The network traffic is now defined by a set of traffic graphs, which share the same nodes: $\{G_1, \ldots, G_p\}$ with $G_i = (V, E_i, \omega_i), 1 \leq i \leq p$, where $\omega_i : E_i \rightarrow \mathbf{R}$, is the function defining the edge weights. Each graph represents the traffic of a period of time (e.g. the graph G_1 may represent the traffic from 8am thru 9am, G_2 is the traffic from 9am thru 10am etc.).

The graph G_1 is clustered and visualized using STRUCTUREMINER. The edges are normally not shown. Colors are used to highlight clusters. The main idea is now to leave this first layout unchanged during the presentation of the next graphs $G_i, i > 1$, but to modify the cluster membership of the nodes, i.e. their color, only. This enables the user to recognize changes between the initial clustering (represented by the node layout) and the actual clustering (represented by the node colors).

An example can be seen in Fig. 10 (left-hand side), the

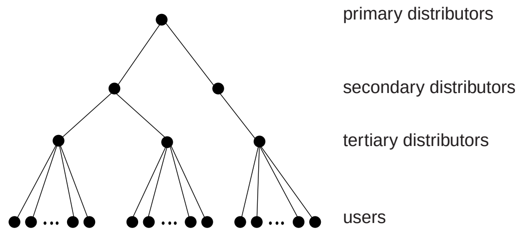


Fig. 12. Structured Cabling.

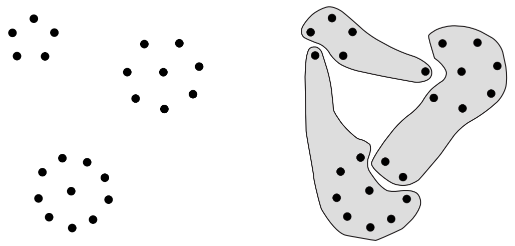


Fig. 13. The Groups of the physical Topology (left-hand side) and a superimposed second Clustering (right-hand side).

colors used in STRUCTUREMINER are here replaced by borderlines. The figure on the right-hand side shows the same graph layout, but without edges. Now the next graph G_2 is clustered. This new clustering is applied to the old layout, i.e. all nodes keep their positions, but they may belong to different clusters. A new cluster membership results in a different node color. This process continues for the graphs $G_i, i > 2$. Fig. 11 depicts two typical cluster changes.

B. Analysis of Network Topologies

Choosing a network topology is a main problem for a network administrator, he/she must find a solution which takes the cabling and the traffic into consideration. Usually every computer is connected to a so-called tertiary distributor by a single cable[4]. The tertiary distributors are connected to secondary distributors, which are connected to a single primary distributor. Therefore the cabling forms a tree (see Fig. 12). This view onto the network is called the physical topology of the network.

The administrator is now faced with the problem of subdividing the set of computers connected to a tertiary distributor into several groups. Computers in the same group are connected directly using a hub or a switch. Because of technical restrictions the maximum number of computers connected to a single switch or hub is restricted. Since communication within a group is faster than the communication between groups, the administrator wants computers that communicate with each other quite frequently to be in the same group. Once a good grouping has been found, changes of the user behaviour or new technologies can influence its quality.

STRUCTUREMINER can now help the administrator to verify the quality of an existing grouping. For this, the graph representing the physical topology of the network is visualized using STRUCTUREMINER. The clusters are provided by the user, they correspond to the existing groups.

Now the traffic graph of the same network, i.e. the graph, whose edges represent the network traffic, not physical connections, is clustered. This new clustering is added to the visualization by using node colors, i.e. the previous layout of the network is not changed. One clustering is now represented by the layout of the nodes (the physical topology) and another clustering (the traffic structure) is superimposed using node colors. By comparing those two clusterings, the administrator can see whether the existing grouping still makes sense.

[4]This type of cabling has been standardized in EN 50173 and ISO/IEC DIS 11801.

Fig. 13 (left-hand side) shows an example. The groups can be easily recognized by the distributions of the nodes, edges are not shown. Fig. 13 (right-hand side) shows the superimposed clustering of the traffic graph (the colors are replaced by borderlines), obviously the clusterings differ only slightly.

C. Intrusion Detection

The security of computers and computer networks is increasingly important. E-commerce, online-banking, and sensitive documents in enterprise networks need adequate protection. A major threat arises from attack tools which can be downloaded easily from public web servers (see for example [3]) and which can even be used by untrained attackers ("script-kiddies"). In-depth-knowledge of the attacked systems is no longer necessary. These attacks, dangerous as they are, have a certain lack of sophistication in common, i.e. their way of attacking is predictable. This allows for the use of software tools, which can detect typical attack patterns.

Today, besides traditional concepts of network security like firewalls and cryptographic protocols, intrusion detection systems (IDS) are used in networks. The goal of IDS usage is the early detection of security violations in order to prevent or (at least) to reduce damage. A graph-based system for the detection of intrusions in computer networks has been described in [19].

D. Denning presents in [2] a model for intrusion detection and states "... exploitation of a system's vulnerabilities involves abnormal use of the system; therefore, security violations could be detected from abnormal patterns of system usage.". The visualization technique presented before and specially the clustering method of MAJORCLUST is the base for the new anomaly detection system described here.

Clustering leads to a simplified and stable structure graph, representing the general traffic structure in the surveyed network. The left-hand side of Fig. 14 shows an example of a typical communication structure found in a surveyed network. The star-like topology of the clusters is caused by the structure of client/server applications and the fact that a lot of data is exchanged inside working groups but much less data

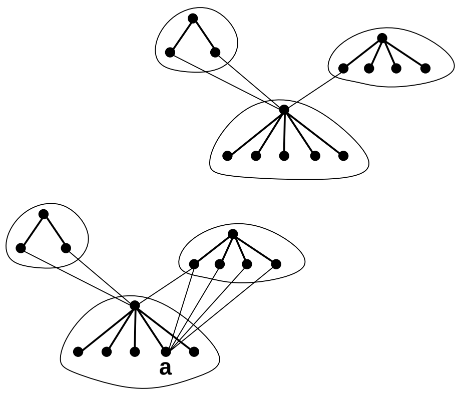


Fig. 14. A typical Communication Structure(left-hand side) and a scanning attack by node "a" (right-hand side).

is exchanged between different groups.

Most kind of network attacks cause modifications of this traffic structure. Fig. 14 (right-hand side) presents a visualization of an attack. The origin of the attack is node "a", which is systematically scanning other computers in the surveyed network. The amount of additional traffic is not large enough to modify the clustering, but statistical values have changed. The attack causes a larger inter-cluster traffic.

In Fig. 15, we present a different scenario: Node "a" is now being attacked by a flooding or denial-of-service attack. The large amount of packets sent leads to a different clustering result. In addition, a lot of new nodes can be seen, representing the different spoofed source addresses of the attacking packets.

A visualization of these structures helps the network administrator to detect the sources of anomalies. It enables him to detect changes of the traffic structure on a relatively abstract level, while normally intrusions are covered by the enormous amount of network packets.

IV. Summary

In this work a general strategy for a structure-emphazising visualization of network traffic is introduced. The automatic abstraction, recognition, and drawing of traffic graphs allows the administrator to understand even complex traffic situations and to recognize typical traffic patterns. In addition, an implementation of this strategy is described. Finally, three application are presented, which use the visualized traffic graphs to solve non-trivial problems of network administration.

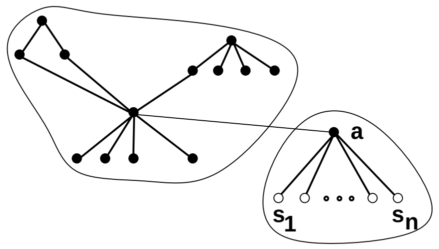


Fig. 15. Victim "a" being attacked by packets with spoofed source addresses $s_1 \ldots s_n$.

References

[1] R. Beale and T. Jackson. Neural computing. Institute of Physics Publishing, Bristol, Philadephia, 1990.

[2] D. E. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, February 1987.

[3] D. Dittrich. web-page with papers on different attack tools. http://www.washington.edu/People/dad.

[4] P. Eades. A heuristic for graph-drawing. *Congressus Numerantium*, 42:149–160, 1984.

[5] T. Fruchterman and E. Reingold. Graph-drawing by force-directed placement. *Software-Practice and Experience*, 21(11):1129–1164, 1991.

[6] Sven Meyer zu Eißen. Natürliche Graphpartitionierung am Beispiel von Aufgabenmodellen in Unternehmensnetzwerken. Master's thesis, University of Paderborn, 2000.

[7] B. Huffaker, J. Jung, D. Wessels, and K. Claffy. Visualization of the growth and topology of the nlanr caching hierarchy. http://www.caida.org/Tools/Plankton/Paper.

[8] B. Huffaker, E. Nemeth, and K. Claffy. Otter: A general-purpose network visualization tool. http://www.caida.org/Tools/Otter/Paper.

[9] D. Jungnickel. *Graphen, Netzwerke und Algorithmen*. BI Wissenschaftsverlag, Wien, 1990.

[10] S. T. K. Sugiyama and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernectics*, 11(2), 1981.

[11] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31:7–15, 1989.

[12] T. Kohonen. *Self Organization and Assoziative Memory*. Springer, 1990.

[13] J. Kolodner. *Case-based reasoning*. Morgan Kaufmann, San Mateo, CA, 1993.

[14] D. B. Leake. Case-Based Reasoning: Issues, Methods, and Technology, 1995.

[15] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. B. G. Teubner, Stuttgart, 1990.

[16] B. Mirkin. *Mathematical Classification and Clustering*. Kluwer Academics Publishers, 1996.

[17] O. Nigggemann and B. Stein. A meta heuristic for graph drawing. *Proceedings of the Working Conference on Advanced Visual Interfaces*, 2000.

[18] K. Pearson. Contribution to the mathematical theory of evolution. *Phil. Trans.*, 1894.

[19] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle. Grids: A graph- based intrusion detection system for large networks. In *Proceedings of the 19th National Information Systems Security Conference*, 1996.

[20] B. Stein and O. Niggemann. *25. Workshop on Graph Theory*, chapter On the Nature of Structure and its Identification. Lecture Notes on Computer Science, LNCS. Springer, Ascona, Italy, July 1999.

[21] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, November 1993.