Interactive Exploration of Data Traffic with Hierarchical Network Maps

Florian Mansmann and Svetlana Vinnik

***Abstract*— Network communication has become indispensable in business, education, and government. With the pervasive role of the Internet as a means of sharing information across networks, its misuse for destructive purposes, such as spreading malicious code, compromising remote hosts or damaging data through unauthorized access, has grown immensely in the recent years. The classical way of monitoring the operation of large network systems is by analyzing the system logs for detecting anomalies. In this work, we introduce *Hierarchical Network Map*, an interactive visualization technique for gaining a deeper insight into network flow behavior by means of user-driven visual exploration. Our approach is meant as an enhancement to conventional analysis methods based on statistics or machine learning. We use multidimensional modeling combined with position and display awareness to view source and target data of the hosts in a hierarchical fashion with the ability to interactively change the level of aggregation or apply filtering. The interdisciplinary approach integrating data warehouse technology, information visualization, and decision support, brings about the benefit of efficiently collecting the input data and aggregating over very large data sets, visualizing the results, and providing interactivity to facilitate analytical reasoning.**

***Index Terms*— Data and knowledge visualization, Information Visualization, Visual Analytics, Network Security**

I. Introduction

SINCE the Internet has de-facto become the information medium of first resort, each host on the network is forced to face the danger of being continuously exposed to the hostile environment. Therefore, network security has turned into one of the central and most challenging issues in network communication for practitioners as well as for researchers. Security vulnerabilities in the system are exploited with the intention to infect computers with worms and viruses, to ”hack” company networks and steal confidential information, to run criminal activities through the compromised network infrastructure, or to paralyze online services through denial-of-service attacks. Frequency and intensity of the attacks disallow any laxity in monitoring the network behavior of the system.

F. Mansmann and S. Vinnik are with the University of Konstanz, Germany.

Various automatic methods, such as firewalls, virus scanners and intrusion detection systems, have emerged as a response to the need of protecting the systems from harmful network traffic. However, as there will always exist human and machine failures, no fully automated method can provide absolute protection. One may distinguish between unintentional defects due to human failures or other malfunctions, referred to as *flaws*, and the intentional misuses of the system, known as *intrusions*.

A non-trivial task of detecting different kinds of system vulnerabilities can be successfully solved by applying the visual analytics approach. Whenever machine learning algorithms become insufficient for recognizing malicious patterns, advanced visualization and interaction techniques encourage expert users to visually explore the relevant data and take advantage of human perception, intuition, and background knowledge. In the process of human involvement acquired knowledge can be further used for advancing automatic detection mechanisms.

Intrusion detection is the major preventive mechanism for timely recognition of malicious use of the system endangering its integrity and stability. Our research is concerned primarily with *anomaly-based* intrusion detection systems (IDS) which offer a higher potential for discovering novel attacks, compared to the *signature-based* IDS. Anomaly-based detection is carried out by defining the normal state and behavior of the system with alerts sent out whenever that state is violated. It is a rather complicated task to define the normal behavior precisely enough as to minimize false alerts and especially not to let any attacks evolve unnoticed. This is where our approach enters.

We offer a visual view on the system behavior with the input data read from the system logs of

network gateway routers or other sources logging network traffic. The user is free to explore the input along various dimensions and to retrieve the details of the network flows. IP addresses and other attributes are modeled as hierarchical dimensions in which the instances are grouped into upper-class categories at multiple levels, thus forming a tree structure. The visual interface is optimized for displaying hierarchies, preserving geographical position, proximity, and the size of network nodes, thus facilitating their comparability.

We call our technique *Hierarchical Network Map*, since it organizes the display area as a map with a zoomable rectangle area for each network, using the associated IP addresses for computing the node's coordinates. The underlying data model was inspired by the OLAP (online analytical processing) approach used in building data warehouses for efficiently managing huge data volumes and computing aggregates under high performance requirements.

The rest of the paper is organized as follows: in the next section we discuss work related to our research, followed by the introduction of our approach in section 3. The underlying multidimensional data model is presented in section 4. Implementation and visualization issues can be found in section 5. Section 6 presents the results and findings of applying the Hierarchical Network Map, followed by an evaluation of our work. Conclusions and future work are given in section 8.

II. Related Work

The goal of visualization for computer security is to display misbehavior and failures within a single computer or a whole network and to ultimately support the administrator in his task to gain insight into their causes. For network security, this goal can be achieved either through interpretation of events produced by an IDS or through measurement of traffic peculiarities. Our work focuses on the latter aspect due to the lack of availability of IDS data. However, this kind of data could be easily imported into our database and analyzed in our system.

An important subarea is the visualization of port activity which is an indication to the running network applications. Lau [1], for example, presented the *Spinning Cube of Potential Doom*. The visualization is based on a rotating cube used as 3D scatterplot. The variables such as local IP address space, port number, and global IP address space, are assigned to its axes. Each measured traffic packet is mapped to a point in the 3D scatterplot. The cube is able to show network scans very easily due to emerging patterns such as horizontal and vertical lines or areas. Another example is the *PortVis* tool described by McPearson et al. in [2]. It implements scatterplots (e.g., port/time or source/port), port activity displays and various means of interaction to visualize and detect port scans as well as suspicious behavior on certain ports. Muelder et al. [3] further extend the tool by producing wavelets from scatterplots of network scans. Those wavelets are then clustered to classify the scans according to their characteristics.

In some scenarios, the amounts of alerts produced by an IDS are too large to be scanned manually. Visualization modules such as *IDS Rainstorm* by Abdullah et al. [4] bridge the gap between large data sets and human perception. They provide a scatterplot-like visualization of local IP addresses versus time enhanced by zooming functionalities. The IDS events are then connected by lines to the global IP address space in the detail views.

A. IP Address Space Visualizations

Errors (e.g., routing instabilities) and intrusions in computer networks were the focus of a study conducted by Teoh et al. [5]. The authors introduced a recursive QuadTree that assigns positions according to the prefixes of the IP addresses. For each event, the point representing the IP address is connected with the two involved autonomous systems (placed outside of the QuadTree) by lines. Depending on the security events occurring in the network, characteristic patterns appear.

Fink and North [6] introduce the *Root Polar Layout* of the IP address space. Their assumption is that the classification of the IP address space in trust levels can reveal useful information. IP addresses of the highest trust level are drawn in the inner circle, with IP addresses of lower trust levels arranged on concentric rings around this circle.

The major advantage of our *Hierarchical Network Map* versus the QuadTree approach and the Root Polar Layout is a more meaningful positioning of the IP addresses on the display. We combine network topology with geographic information for an improved understanding of and orientation in

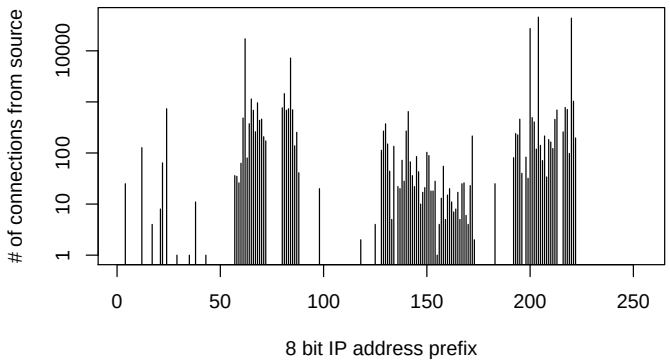


Fig. 1. Density histogram showing the distribution of sessions over the IP address space.

the analyzed data sets. However, due to the uncertainty involved in the geographic placement of autonomous systems, misplacements are to be taken into account.

B. Commercial IDS Visualization Modules

It is worth mentioning that visualization techniques like graphs and Parallel Coordinates have meanwhile found their way into commercial products, such as the *RNA Visualization Module* of SourceFire [7]. In this context, the techniques are used to display the connectivity of network nodes and the correlations in the multi-dimensional characteristics of associated network traffic.

III. Hierarchical Network Map Approach

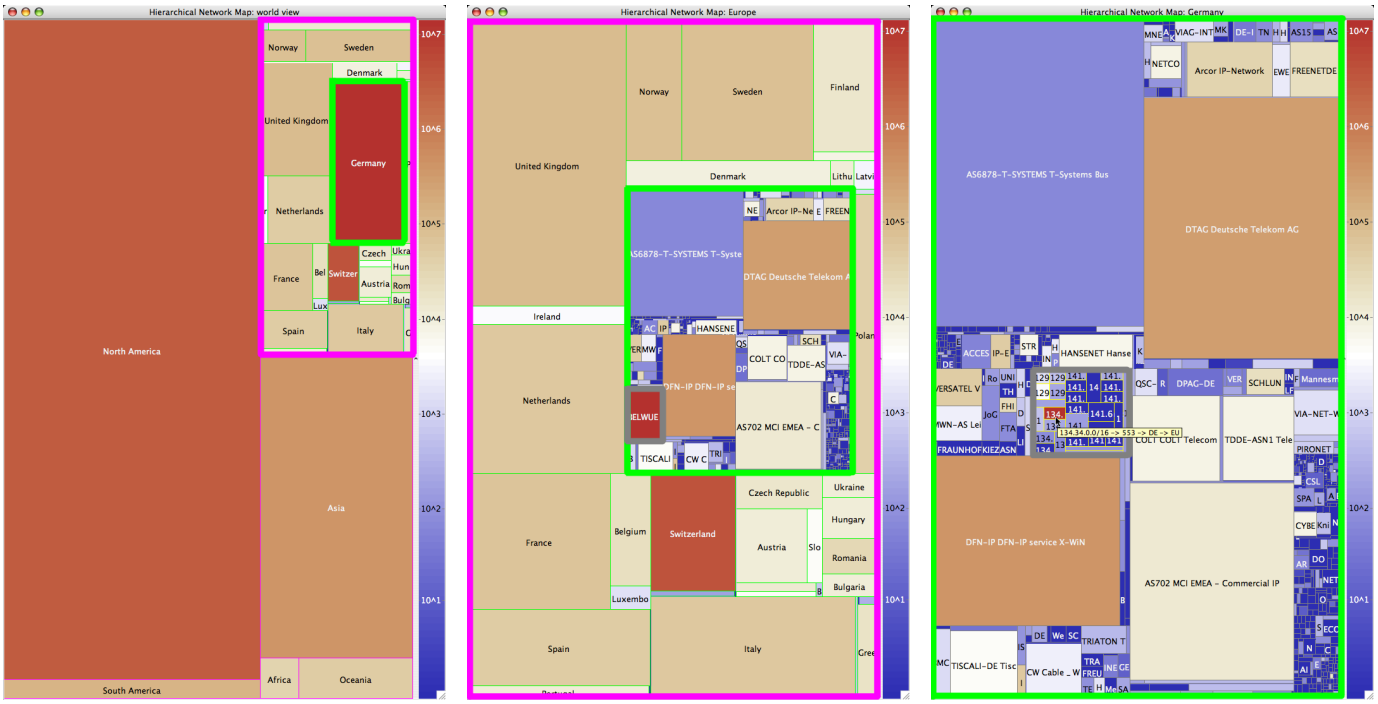
Hierarchical Network Map is a visual exploration approach aimed at displaying the distribution of source and target data traffic of network nodes (IP addresses) in a more expressive way than a simple density diagram like the one shown in Fig. 1. The intention to provide an overview of the entire Internet on the screen has forced us to treat every pixel as a valuable asset and to use a space-filling technique. Compared to node-link diagrams, our technique has three major advantages: 1) no display space is wasted, 2) larger data sets can be visualized mapping network size to the node's area, and 3) better support for multi-resolution exploration (drill-downs have only local effects on the layout).

The whole network structure can be viewed as a hierarchy with the IP addresses of single hosts as the bottom-level nodes. Each host belongs to a local IP network which, in turn, is members of an autonomous system. On top of the network levels mentioned above, the IP address hierarchy is extended by two geographic upper classes, namely, countries and continents. The display rationale is to recursively nest child node rectangles in their parent rectangles, from the continent level down to local networks (see Fig. 2), or even single hosts.

In many space-filling hierarchical visualizations such as TreeMaps [8], the area is partitioned according to a variable statistical measure so that the sizes of the resulting rectangles correspond to the respective value. In our scenario, choosing the per-node traffic as a measure would lead to constantly changing sizes and positions of node rectangles on the display thus aggravating user's orientation. Since the success of our technique is measured by its applicability for continuous network monitoring and analysis, recognition and familiarity turn into critical requirements. Therefore, we map the screen area and its partitioning to a rather non-volatile (at least in the short-run) measure, namely, the total size of the network and its components. Given a constant ratio of the display space, this design results in an almost static map layout, as network architectures hardly experience any changes in the short term.

While the containment relationships within rectangles show the IP address hierarchy and class membership, the size of each hierarchical region, from a single IP address all the way up to a whole continent, is proportional to the number of IP addresses it contains. Furthermore, the geographical nodes, i.e. those at country and continent level, additionally preserve their relative geographical position (similar to a cartogram). It is this geographic awareness of the node that allowed us to classify our technique as a *map*. At the level of autonomous systems and local networks, the contained rectangles appear sorted by IP address in a left-to-right and top-down fashion, thus accelerating the visual lookup of any sub-node.

In terms of analytical reasoning, the statistical values $\mathcal{X} = (x_i)_{i=1,\ldots,n}$ with $x_i \geq 0$ and $x_i \in \mathbb{R}$ in our scenario are the per-node traffic loads measured as a total number of bytes, packets, or sessions within a specified time frame. These statistical values are represented by the filling color of the node's rectangle areas, from dark blue for the lower bound to dark red for the upper one. We favor the logarithmic color scale as shown in Fig. 2 due to its improved discrimination for low values and smoothing effect on outliers.



(a) World view (b) Continent view (c) Country view

Fig. 2. Multi-resolution approach using the *Hierarchical Network Map* to investigate the outgoing traffic of a chosen local network within the University of Konstanz in terms of number of packets sent from 27th September to 4th of August 2005. a) In the world view, with the European continent drilled down, one may observe high network traffic targeting Germany. b) A continent view of Europe, with a drill-down on Germany, reveals that the volume at a single autonomous system (AS 553, Belwue) in Germany exceeds the aggregated traffic to Switzerland. c) In the view of Germany, a drill-down on BelWue (the research network of the federal state Baden-Württemberg, Germany) confirms the expectation that our university network (134.34.0.0/16) receives most data packets.

Further characteristics of network flows are implemented as filters, which are either compulsory or optional.

Compulsory filters:

1) *Type of load*: Since each IP address functions both as source (sending packets) and as target (receiving packets), the network load to display can refer either to the packets sent, or the packets received, or their total.
2) *Time window*: Each visualization instance displays the traffic within a specified time interval.

Optional filters:

1) *Port or Port Cluster*: Single ports or their groupings can be used as filter to display only the portion of traffic occurring at those ports (for instance, showing the outgoing email load by selecting the source port 25 (SMTP)).
2) *Protocol*: It might be useful to filter the traffic of a particular protocol, for instance to separate UDP traffic from that of TCP.

It turned out to be infeasible to process the entire network data the way it is protocolled by the system logs. The generated amount of ”raw” operations is simply too large to store and analyze even on high-performance workstations. For example, to apply our approach for exploring traffic at a gateway of a middle-sized university would result in storing several gigabytes of log data per hour. To avoid input data overflow, we store the aggregated operations, in which the packets are grouped into sessions and the sessions of related flows, i.e. those referring to the same source and target within close timestamps, are summed up. Therefore, the load recorded by such aggregated entry is described by the number of sessions, the number of packets transferred, and the transferred bytes.

Our proposed visualization can be applied both in *online* mode, for actual network traffic monitoring, and in *offline* mode, for exploring the network’s behavior in a selected time and space window. Let us consider each operation mode to reveal their constraints and implications in terms of required input data and performance.

A. Network Flow Exploration (Offline Mode)

In offline mode, past traffic data is loaded from the database. The user specifies the type of load and the time window to set as compulsory filters. Optional filters can be specified as well or added in the process of interaction.

Suppose that a network administrator is concerned about the degrading performance of UDP traffic in the network as more and more packets get dropped. He starts off by choosing the target load filter to trace where the traffic was sent to, and proceeds by selecting a time period starting at the moment he first discovered the problem. Finally, the protocol filter is activated to show only UDP traffic. Running the *Hierarchical Network Map* helps him to formulate a hypothesis about the target distribution of the analyzed data flows. Interactively, he discovers that most of the traffic is sent to Germany and selects another date to verify whether such large amounts of traffic are typical for that target node. To verify whether the spotted traffic pattern is the result of a continuous growth or whether it appeared all of a sudden, the network administrator runs an animated view of the daily traffic from the previous month. Observation of a continuous upward trend make it obvious that there is a need for better connectivity to the target network.

B. Network Flow Monitoring (Online Mode)

Monitoring network flows in *online mode* is a challenging task due to high data arrival rate and run-time requirements. Monitoring can be performed by refreshing the display either continuously or periodically. Continuous re-rendering imposes extreme costs since each incoming event affects the visualization. Therefore, it is rather imperative to define a reasonably short interval in which the visualization gets updated with newly arrived data.

In many networks, daily operations are monitored by measuring the performance of the gateway routers. We believe that by extending the mere observation of traffic statistics to also take into account the source and target distribution in terms of IP addresses involved, may result in improved adjustment of the routing policies of large networks. In the long term, considerable cost savings can be achieved if the same performance of the network is achieved with less hardware resources due to more intelligent routing policies.

IV. Network traffic as OLAP cubes

We have found the *OLAP* (On-Line Analytical Processing) architecture [9] to be a suitable option for processing huge amounts of network data in an anomaly detection scenario. OLAP based solutions, initially adopted by traditional business applications, have proven to be beneficial and extendable to serve a much wider range of application domains, such as health care, biology, government, education, and network security, to name a few. An example for an advanced visual OLAP tool is Polaris [10]. The underlying *multidimensional data model* [11] allows mapping of detailed transactional data, denoted *facts*, as being arranged in a multidimensional space, or *hypercube*. The numerical values under analysis, termed *measures*, are characterized by descriptive values, drawn from a number of *dimensions*. The values of any single dimension can be further organized in a containment-type *hierarchy*, thus analytical queries can efficiently compute the measure aggregates for various levels of detail.

Transactional facts are periodically retrieved from network traffic logs and stored in the fact table *Log*. Initial log entry describes a single network packet transfer by storing source and target IP addresses and ports, the timestamp, and the packet size. To reduce the volume of facts to be stored, all packets and session referring to the same source, destination and timestamp are grouped into a single fact, with the number of sessions, the number of packets and their total size in bytes as its three measures.

Each of *Log*'s dimensions has one or more hierarchies defined upon it, as follows:

- *SourceIP* and *TargetIP* are grouped by *network* → *autonomous system* → *country* → *continent*. Hosting country of an autonomous system is determined by looking up the geographical positions of the IP addresses of all the networks it contains and choosing the prevailing country. For this, we rely on the GeoIP database of Maxmind, Ltd. [12] (approx. 99% accuracy).
- *Timestamps* are aggregated as *millisecond* → *second* → *minute* → *hour* → *day* → *month* → *year*.
- *SourcePort* and *TargetPort* may be grouped into *categories* (such as well-known, registered and dynamic ports) on the one hand, and into *domains* (e.g., web, email, database, etc.), on the other hand.

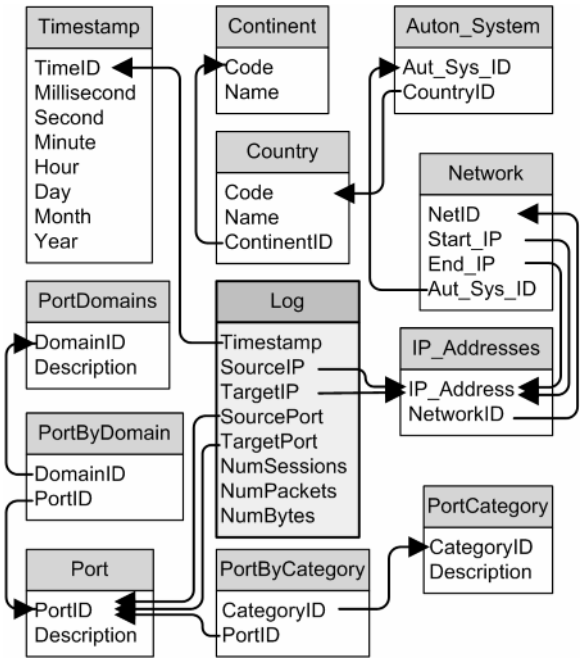


Fig. 3. Modelling network traffic as an OLAP cube.

Overview of the above logical database design is depicted in Figure 3. Decomposing dimensional tables into subtables for each granularity level results in a logical design called *snowflake schema*. For performance considerations the dimension *Time* was left denormalized.

A. Measures

Notice that the original *Log* table allows to trace either the outgoing or the incoming traffic load at each IP address, but not their total. This total load is obtained by deriving a new fact table ***Load*** (*Timestamp, IP, Port, NumSessions, NumPackets, NumBytes*), whereby the outgoing (*SourceIP*, *SourcePort*) and the incoming (*TargetIP*, *TargetPort*) traffic for each IP address is merged by joining the original *Log* table or its aggregate with itself.

B. OLAP Operations and Queries

The fact table contains a huge volume of ”raw” or poorly aggregated data. OLAP operations are used for computing a measure’s aggregates (subtotals, averages, bounds) for a chosen combination of dimensions and their granularity levels:

- *Roll-up* (aggregating) and *drill-down* (disaggregating) allow to change the granularity level.
- *Slice-and-dice* defines a subcube of interest or even reduces the dimensionality by ”flattening” single dimensions to one selected value.
- *Ranking* can be applied as a dynamic filter showing the specified number of marginal (top or bottom) measure aggregates.

Since our visualization approach restricts itself to a rather limited set of OLAP operations, we were able to relax some of the classical OLAP restrictions in part of hierarchy properties, such as balancedness, covering and strictness. For instance, one port may belong to multiple domains or not belong to any.

C. Performance via Pre-aggregation

As elaborated in the previous section, it may be unfeasible or simply impossible to manage the entire log data ”as it is” in the database. Besides, the older the data entries get the less interesting they are for the analysis, since anomaly detection is basically concerned with recent and especially current traffic flows. Therefore, we suggest to split the Log table into multiple subtables with varying level-of-detail:

- *ShortTermLog* stores the most recent transactions (e.g., for the last one hour) “as they are”, i.e. without any aggregation. This table will be used for the *run-time* network load visualization.
- *MiddleTermLog* stores the network data of the current day with the granularity reduced from millisecond to minute. This table offers still rather detailed information for offline exploration of the current day’s network behavior.
- *LongTermLog* aggregates the transactions even further (e.g., by hour or day) to store them as historical data for less detailed exploration.

Other granularity levels along *Time* dimension can be defined depending on the application needs.

V. Implementation and Visualization Issues

The starting point of our visualization technique is a *squarified treemap* [13] in which the aspect ratio of the resulting rectangles is optimized to produce nearly square partitions, thus improving the visibility of the mapped hierarchical structure and the comparability of rectangle areas. We extend the above approach by setting rectangle’s *positioning* with respect to its neighbors to be another relevant

TABLE I
CONFLICTING OPTIMIZATION CRITERIA

	space utilization	area	position	aspect ratio
space utilization			X	X
area			X	X
position	X	X		X
aspect ratio	X	X	X	

optimization criteria. Additionally, we make use of the *color* attribute.

Our work has been inspired by geographical distortion methods such as PixelMap [14] and HistoScale [15] that reposition points and polygons. RecMap [16] as a screen partitioning algorithm is capable of displaying leaves of a hierarchical tree, but does not visualize the hierarchy itself. *Hierarchical Network Map* is a further development of the spatial distortion methods introduced in HistoMap [17]. The main advancement over HistoMap is an optimization towards the use of hierarchical (network hierarchy) and one-dimensional data (IP addresses). Furthermore, our algorithm tries to avoid long and thin rectangles to enhance their visibility and comparability.

We figured out that the following four criteria are to be taken into consideration: 1) Full *space utilization*: an optimal solution for space-filling is achieved by recursively splitting the display space into two rectangles. 2) Preserving *proportions* across rectangles: this criteria is fulfilled by setting the proportions of the two resulting rectangles according to the number of IP addresses they represent. 3) *Geographic awareness* (position): The optimal positions are taken to initialize the layout, but are not considered in the process of further optimization. 4) *Aspect ratio* of rectangle areas is optimized by evaluating several alternative layouts. Some of these criteria are conflicting (see Table I), for instance, aspect ratio conflicts with full space utilization, since the former cannot avoid rendering rather "stretched" screen partitions.

A. The Partitioning Algorithm

As an initial setting, we have one spatial point p_i for each continent, country, autonomous system, or network; $\mathcal{P} = \{p_1, \ldots, p_n\}$ where $(p_i)_{i=1,\ldots,n} \in \mathbb{R}^2$. A weight $w_i \in \mathbb{N}$ is attached to each rectangle and is equal to the number of IP addresses represented by p_i. Function $area$ determines the area of each rectangle on the screen as follows (d_x is the width and d_y the height of the display):

$$area(r_i) \;=\; \frac{w_i}{\sum_{i=1}^{n} w_i} \times d_x \times d_y \qquad (1)$$

The algorithm searches for a partitioning of the display space into $|\mathcal{P}| = n$ rectangles $\mathcal{R} = \{r_1, ..., r_n\}$. Each time a point set P is split into P_1 and P_2, the representative rectangle r is split into two rectangles r_1 and r_2. Split direction depends on the squareness of r_1 and r_2; position of the split line between the rectangles is determined by the sum of weights associated with each point set.

```
procedure partition (P)
begin
    if |P| > 1 then
        (P1, P2) ← splitRect (P);
        partition (P1);
        partition (P2);
    else
        drawRect (P);
end
procedure splitRect (P)
begin
    (P1,P2) = splitHorizontal (P);
    (P3,P4) = splitVertical (P);
    if quality (P1,P2) > quality (P3,P4) then
        return (P1,P2)
    else
        return (P3,P4)
end
```

By allowing the algorithm to do one *look-ahead*, we consider $n \times 2^2$ solutions; instead of checking just two solution options (like above), the function splitRect has to check four alternatives. We define a quality function for the aspect ratios of the resulting rectangles. The optimum is set to squares for visibility and comparability:

$$quality(\overline{\mathcal{P}}) \;=\; \frac{1}{n}\sum_{i=1}^{n} w_i \times max\left(\frac{dr_i^x}{dr_i^y}, \frac{dr_i^y}{dr_i^x}\right) \qquad (2)$$

B. Data Preprocessing

The algorithm needs a set of spatial points $\mathcal{P}$ as input. For *continent* and *country* levels of the hierarchy, the assignment is a chosen central point within each continent or country. However, *autonomous system* and *network* levels are only defined in one-dimensional space. To acquire the missing second dimension, we set $p_i^x = p_i^y = ip_{mid}$, where $ip_{mid} =$

$\frac{1}{m}\sum_{j=1}^{m} ip_j$ (m is the number of IP addresses contained in the node). Arising conflicts between two or more autonomous systems or networks having the same spatial point assigned are resolved during the recursion by moving the first point to the left.

From the fact that $p_i^x = p_i^y$, we conclude that vertical and horizontal partitioning split the initial point set into the same resulting point sets. Exploiting this fact, we can calculate the splits and weights a priori and run the optimization only on the widths and heights of the resulting rectangles. This reduces the runtime complexity from $O(n^2 \times 2^{\ell+1})$ to $O(2^{\ell+1} + n \times log(n))$, where ℓ is the number of *look-aheads* at each level. Arranging one-dimensional items into a binary tree (each item needs to be a leaf) can be seen as a sorting problem solvable in $O(n \times log(n))$. Note that our algorithm computes a good solution to the problem rather than the optimal one.

C. Descending to the Pixel Level

The technical limit of a visualization is to use every pixel for displaying distinct values, where the latter are mapped to the pixels' color. Further attributes of the data points can be mapped to the pixels' coordinates. Pixel-based visualization builds up the finest granularity view of the *Hierarchical Network Map*, namely, the bahavior of single hosts. For instance, it can be employed for determining whether the network traffic just comes from a limited number of IP addresses or is scattered over the considered network (see Fig. 4). Pixels are arranged according to the *recursive pattern* [18] starting at the top left corner going down row by row with alternating forward-backward direction for better cluster preservation. Thereby, the displayed IP addresses appear sorted in descending order. Using a large display wall with 8.9 Megapixels, we can show up to 8.9 million IP addresses mapping each IP address to one pixel.

D. Interaction

Interactivity is the key to many visual analytics applications, and our *Hierarchical Network Map* is by no means an exception. The user chooses which region of the network traffic data set at hand should be investigated in more detail, with the choice of the next step depending on the patterns revealed in the preceeding exploration steps.

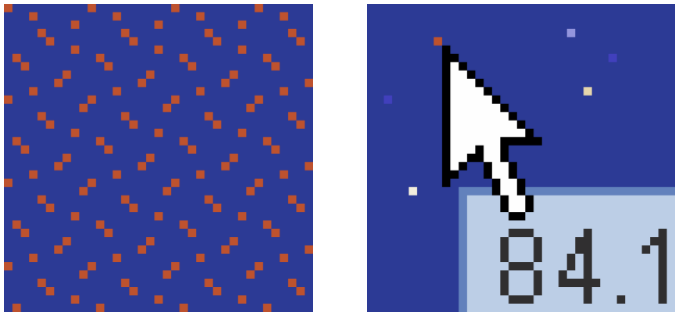


Fig. 4. Recursive pattern pixel visualization can show the active hosts within the network and reveal their distribution behavior. Left: pattern of a network scan, affecting every 10^{th} IP address. Right: a pattern with only a few target hosts. Moving the cursor over the pixel (or its surrounding region) triggers label appearance.

Explorative tasks are enabled through typical OLAP database operations such as drill-down (disaggregation), roll-up (aggregation) and slice & dice (filtering). All these basic interactions are mapped to the mouse wheel and left mouse button to foster easy interaction. A popup menu (right mouse button) offers additional support for untrained users as well as some expert features such as multiple selection or drill-down / roll-up operations on the nodes of the same level. Furthermore, interactive time, host and port activity diagrams can be accessed through this menu.

E. Visual Scalability

The term *visual scalability* [19] describes the capability of visualization tools to display large data sets in terms of the number or the dimensionality of data elements. We designed the *Hierarchical Network Map* to be highly scalable to support a large tree structure (7 continents, 237 countries, 19 731 autonomous systems and 188 248 networks) yet with special effort to facilitate the overview. To which extent this explorative potential can actually be exploited depends on the available display size. Technological advancements such as large display walls enhance scalability while maintaining the overview (see Fig. 5).

F. Visual Hints

In our technique, the visual variable color is used for mapping the specified traffic load measure. Employing a bipolar colormap (blue to red over white as the switching point) supports the observation that not only the existence of high load values (dark

Fig. 5. *Hierarchical Network Map* displaying all 19 731 autonomous systems (one can still zoom in twice for details) on a large display wall (5.20m × 2.15m, 8.9 Megapixels, powered by 8 projectors). The query interface on the top left shows the traffic distribution over time and specifies the selected data, in this case the traffic entering the gateway of the University of Konstanz on *well-known ports (0-1023)* on Nov. 29, 2005 using "transferred bytes" as measure with logarithmic color mapping. One recognizes a heavy traffic load from AS 3320 (red) of "Deutsche Telekom" as well as to neighboring autonomous systems in Germany. A port histogram reveals high activity on the web ports 80 and 443. For security and privacy reasons, the data was aggregated and sanitized.

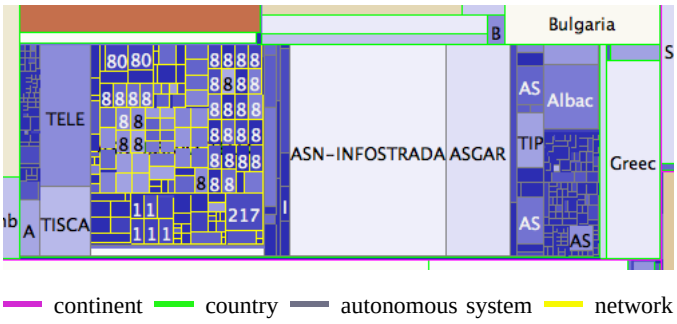


Fig. 6. Borders at each level have distinct colors to preserve the visibility of various hierarchy levels. Borders are only drawn if a rectangle is higher and wider than 2 pixels.

red) can imply interesting findings, but also the non-existence of any traffic (dark blue). Use of square root and logarithmic color scales helps to make the visualization more resistant to outliers, taking into account that comparing nominal values on the color scale becomes infeasible. For this purpose, linear mapping is also available.

Notice that adjacent rectangles are visually separated from each other solely by one-pixel thin borders, irrespective of the rectangles' resolution. No additional borders are placed around parent rectangles in order to maximize the effective display area. To improve visual perception of the hierarchical structure represented by the borders and of the depth of rectangle partitions, the borders are highlighted with a distinct color for each granularity level (see Fig. 6). For better visibility of the labels, these are displayed in black or white depending on the better contrast to the rectangle in the background.

VI. Results and Findings

To demonstrate the intrusion detection potential of the Hierarchical Network Map, let us consider some use scenarios. In case of *trojan horse* activity, malicious code in form of so-called root kits installs itself on the infected computer and connects to internet relay chat (IRC) servers (TCP ports 194, 529, 994, 6665-6669) from where it receives further commands. The trojan horse *Backdoor.Ryknos.B*, for example, establishes connections to the public IRC servers at the IP addresses 68.101.14.76, 24.210.44.45, 67.171.67.190, 152.7.24.186 or 35.10.203.93. The network administrator can proceed by filtering out the traffic on the affected ports and checking for possible dependencies between chatting activities within the supervised network and known compromised chat servers.

Another example is the *slammer worm* that sends itself to the UDP port 1434, on which the Microsoft SQL Server Resolution Service listens. Due to a security vulnerability, the worm infected many MS SQL servers and opened countless connections at UDP port 1434 to randomly chosen IP addresses leading to network traffic congestion. We tested our visualization by choosing the measure target connections on such a scenario. The simulated traffic to random destinations made those networks more prominent (white to red) in the visualization.

Running two parallel instances of the Hierarchical Network Map for source and target IP addresses, respectively, of the same data set is a powerful way of filtering distinct data flows and exploring the data from several perspectives. A specific destination network can be chosen (such as the network with the highest load) and matched with the locally administrated IP address space. This is especially useful when supervising a large network.

Investigation of the historical trends can be improved by pre-normalizing the color map to the highest occurring value within the whole time span. In the online monitoring mode, however, the maximum is unknown and, therefore, has to be either estimated or adapted dynamically.

With respect to UDP versus TCP traffic behavior, at the time of our measurements we made the following observation: TCP traffic prevails at western destinations, such as the United States, whereas UDP loads dominate towards Asia. Europe naturally plays the central role considering the traffic of both protocols, apparently due to the geographical location of our gateway and the orientation of its users towards German web contents and services.

Another usage scenario is shown in Figure 7. In this case, the Hierarchical Network Map serves as a grouping function for host activity histograms over the number of connections and transferred bytes. In general, few connections and high transfer volume indicate a download link whereas many connections and low transfer suggest “chatty” network services.

VII. Evaluation

We presented our visualization to the security expert of our university. From her perspective, it was very important to show detailed host activities within a network, autonomous system or country. Usually, she compares the number of connections with the transferred bytes of active hosts to find suspicious patterns. After getting this valuable feedback, we integrated host activity diagrams (see Fig. 7) as well as port activity diagrams into our application. While using our system, we realized that some very small networks with high traffic are only rendered as a few pixels on the screen and are thus hardly perceivable. Even though a large display wall would compensate for this shortcoming, we plan to find a solution for improving their visibility.

TABLE II
Performance measures

	1 h data set	24 h data set
237 countries	2 sec.	20 sec.
19 731 autonomous systems	9 sec.	36 sec.
188 248 networks	23 sec.	60 sec.

With respect to the use of geographic information, we realized two things: a) in some cases neighborhood preservation for countries fails and b) assigning autonomous systems to countries involves uncertainty; thus erroneous information can be communicated. However, until now we have found only one misplaced autonomous system. The system’s strength lies in the possibility to show network traffic at different granularity, so that detailed information can be retrieved at continent, country, autonomous system, or network level.

In our performance experiments, we loaded the world view in three different scenarios, showing all 237 countries, all 19 731 autonomous systems and all 188 248 networks (see Table II). Concerning the performance of our application, updating all countries in the world view with pre-aggregated data spanning 1 hour takes about 2 seconds; a data set spanning 24 hours (hourly aggregated) can be loaded within 20 seconds. The system runs on an Apple Power Mac G5 machine with Dual 2.3 GHz clock speed and 2 GB RAM. The input data is managed in a PostGreSQL 8.0 database. The database performance could be significantly improved by employing high-end server hardware and by using a commercial database product. Note that typical starting point of the exploratory analysis process is a highly aggregated representation in which the user performs a series of successive drill down steps. Thus, the subset of data to be loaded at each step, can normally be done within few seconds.

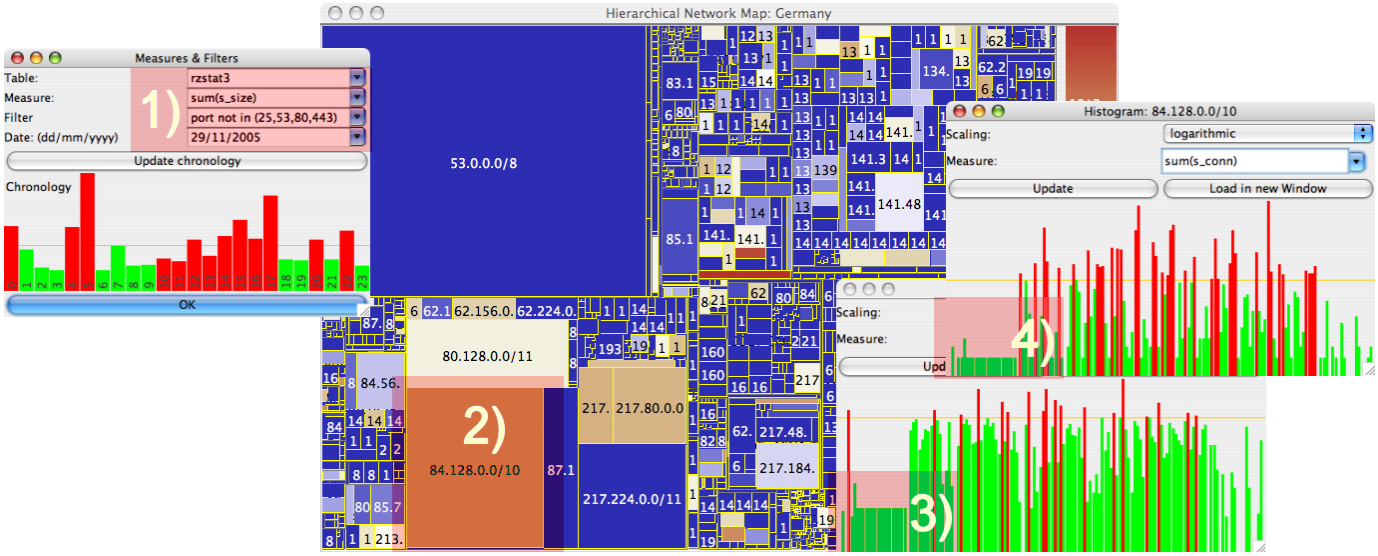


Fig. 7. *Interactive Data Exploration*. 1) We start by specifying the data set, the measure (sum of all incoming connections), the filter (discarding heavy mail, DNS, and web traffic) as well as the date. 2) Having launched the Hierarchical Network Map, we select all autonomous systems within Germany and drill down to the networks. For further analysis, we open a large network (84.128.0.0/10) as we expect more connections to different hosts there than in small networks. 3) The host activity diagram is displayed using "transferred bytes" as measure. By applying a logarithmic scale (only outliers are visible on a linear scale), we identify a set of neighboring hosts that sent the same amount of bytes. 4) The second diagram with the measure "number of connections" reveals the same pattern. After an examination of the used ports, we suspected the trojan Netcontroller on several hosts to actively communicate with a computer in our network on port 123. However, this hypothesis was discarded as only UDP traffic was measured, indicating the network time protocol (NTP).

VIII. Conclusions

Our work can be seen as a novel application in the field of computer security visualization. To the best of our knowledge, such large amounts of IP-related network traffic data have never been shown in a geographically aware hierarchical and space-filling context.

In this work, we presented the *Hierarchical Network Map* as a hierarchical space-filling mapping method for visualizing traffic of IP hosts. Our algorithm is an application of the space-filling variant of the RecMap [16] algorithm which was modified to fit our needs with respect to one-dimensional (IP addresses) and hierarchical data (network hierarchies). The approach aims at finding a compromise between four optimization criteria: using display area to represent network size, full utilization of the screen space, position awareness, and aspect ratio. The performance challenges are mastered by employing OLAP cube operations commonly used in data warehouses to efficiently aggregate over large volumes of detailed data.

We have experimented with traffic data from our university's gateway and from our local computers to demonstrate that visual exploration of the actual network behavior as well as simulation of anomaly scenarios reveal valuable insights in network security. A nested recursive pattern pixel visualization further extends the exploration paradigm by using single pixels as the lowest physical level of the visualization. To adapt the visualization for the large hierarchy at hand, we use one-pixel-thin colored borders between the nodes of each level. Furthermore, intuitive mouse interactions are implemented to facilitate the explorative task.

In the future, we want to further optimize the split sequence of our algorithm with respect to the best measured screen layout and to conduct a formal evaluation of this aspect as well as of the performance aspects of our database and visualization. Additionally, we plan to introduce some novel visualization concepts that foster the exploration of time and port dimensions. Innovative devices to facilitate the interaction with the large display wall will be considered as a traditional mouse does not efficiently support the moving analyst.

To master the network traffic flow in real-time, we plan to apply a streaming database for monitoring purposes in online mode. The challenge to handle is the update strategy for maintaining the pre-aggregation architecture.

As a further application of the map, we plan

to load IP-related data from Intrusion Detection Systems to find non-trivial correlations in the IP dimension within cyber attacks. Moreover, we want to employ our methods to find hacked computers within the university network. As long as those computers are not used for network scans or denial of service attacks, their network traffic does not reveal any statistically significant peculiarities. Yet, they are a ticking time-bomb.

ACKNOWLEDGMENT

The authors thank Daniel A. Keim, Mike Sips, Christian Panse, Benjamin Bustos and Alexander Hinneburg for many fruitful discussions and the anonymous reviewers for their valuable input. Furthermore, we would like to give special thanks to Duncan Sparrell and his group from AT&T, as well as to Barbara Löhle and Marcel Waldvogel from the computing center of the University of Konstanz for providing valuable data for designing and evaluating our approach. This work was partially funded by the German Research Foundation (DFG) under grant GK-1042, Explorative Analysis and Visualization of Large Information Spaces, University of Konstanz.

REFERENCES

[1] S. Lau, "The spinning cube of potential doom," *Communications of the ACM*, vol. 47, no. 6, 2004.

[2] J. McPherson, K.-L. Ma, P. Krystosk, T. Bartoletti, and M. Christensen, "Portvis: a tool for port-based detection of security events," in *Proc. ACM workshop on visualization and data mining for computer security*, 2004.

[3] C. Muelder, K.-L. Ma, and T. Bartoletti, "A visualization methodology for characterization of network scans," in *Proc. IEEE Workshop on Visualization for Computer Security (VizSEC)*, Minneapolis, U.S.A., October 2005.

[4] K. Abdullah, C. Lee, G. Conti, J. A. Copeland, and J. Stasko, "Ids rainstorm: Visualizing ids alerts," in *Proc. IEEE Workshop on Visualization for Computer Security (VizSEC)*, Minneapolis, U.S.A., October 2005.

[5] S. T. Teoh, T. Jankun-Kelly, K.-L. Ma, and S. F. Wu, "Visual data analysis for detecting flaws and intruders in computer network systems," *IEEE Transactions on Computer Graphics and Applications*, September/Oktober 2004.

[6] G. A. Fink and C. North, "Root polar layout of internet address data for security administration," in *Proc. IEEE Workshop on Visualization for Computer Security (VizSEC)*, Minneapolis, U.S.A., October 2005.

[7] Sourcefire. (retrieved on 11/11/2005) Real-time network awareness. [Online]. Available: http://www.sourcefire.com/products/rna.html

[8] B. Johnson and B. Shneiderman, "Tree maps: A space-filling approach to the visualization of hierarchical information structures." in *IEEE Visualization*, 1991, pp. 284–291.

[9] E. F. Codd, S. B. Codd, and C. T. Salley, "Providing OLAP (online analytical processing) to user-analysts: An IT mandate," *Technical report, E.F.Codd & Associates*, 1993.

[10] C. Stolte, D. Tang, and P. Hanrahan, "Multiscale visualization using data cubes." *IEEE Trans. Vis. Comput. Graph.*, vol. 9, no. 2, pp. 176–187, 2003.

[11] T. B. Pedersen and C. S. Jensen, "Multidimensional database technology," *IEEE Computer*, vol. 34, no. 12, pp. 40–46, 2001.

[12] Maxmind, Ltd. (2005) Geoip database. [Online]. Available: http://www.maxmind.com

[13] D. Bruls, C. Huizing, and J. van Wijk, "Squarified treemaps," in *Proceedings of the joint Eurographics and IEEE TCVG Symposium on Visualization*, 2000.

[14] D. A. Keim, S. C. North, C. Panse, and M. Sips, "Pixelmaps: A new visual data mining approach for analyzing large spatial data sets," in *ICDM 2003, The Third IEEE International Conference on Data Mining, 19-22 November 2003, Melbourne, Florida, USA*. IEEE Computer Society, November 2003.

[15] D. A. Keim, S. C. North, C. Panse, M. Schäfer, and M. Sips, "HistoScale: An efficient approach for computing pseudo-cartograms," in *IEEE Visualization 2003, Seattle, Washington, USA*, October 2003, pp. 28–29.

[16] R. Heilmann, D. A. Keim, C. Panse, and M. Sips, "RecMap: Rectangular Map Approximations," in *InfoVis 2004, IEEE Symposium on Information Visualization, Austin, Texas*, October 2004, pp. 33–40.

[17] D. A. Keim, F. Mansmann, C. Panse, J. Schneidewind, and M. Sips, "Mail explorer - spatial and temporal exploration of electronic mail," in *Proc. Eurographics/IEEE-VGTC Symposium on Visualization (EuroVis 2005), Leeds, United Kingdom June 1st-3rd*, 2005.

[18] M. Ankerst, D. A. Keim, and H.-P. Kriegel, "Recursive pattern: A technique for visualizing very large amounts of data," in *Proc. Visualization '95, Atlanta, GA*, 1995, pp. 279–286.

[19] S. G. Eick, "Visual scalability," *Journal of Computational & Graphical Statistics*, March 2002.

Florian Mansmann finished his Bachelor degree in Information Engineering at the University of Konstanz (Germany) in 2003. In 2004, he graduated from the Solvay Management School at the Vrije Universiteit Brussel (Belgium) as a Master of Business Information Management. Since his graduation he is a scholar of the Graduate College Explorative Analysis and Exploration of Large Information Spaces at the University of Konstanz. His research fields are Information Visualization, Network Monitoring and Data Mining.

Svetlana Vinnik received the Bachelor's degree in International Economic Relations from the Belarusian State University, Belarus, in 1999, and the Master's degree in Information Engineering from the University of Konstanz, Germany, 2003. She is currently working as a research assistant in the Databases and Information Systems Group at the University of Konstanz, Germany. Her research fields are data warehousing and business intelligence, decision support, multidimensional data modeling, and e-government.