INED

An Application Independent Network Editor

J. Schönwälder H. Langendörfer

Institute for Operating Systems and Computer Networks
TU Braunschweig
Bültenweg 74/75
D-3300 Braunschweig, Germany
E-mail: schoenw@ibr.cs.tu-bs.de
FAX: +49 531 391 5936

Abstract

This paper presents an application independent network editor called INED which can be used for network documentation tasks and as a graphical user interface for network management applications. We will discuss the requirements that motivated the design of INED and we will introduce the interface which allows applications to use INED as a graphical user interface.

1 Introduction

The decentralization of computing resources has resulted in very complex networks which have become an increasingly important factor for the success of many enterprises. This is due to the fact that the likelihood and costs associated with faults or performance inefficiencies increase with the scale and complexity of a network.

Planning and maintaining large scale complex networked systems requires a set of tools that assist network designers, operators and users. A central document is the network map, which describes the network structure and how nodes and subnetworks are connected. Such maps form the basis for many different tasks during the network life cycle:

- Network planning and design
- Network simulation
- Network operation
- Error discovery and diagnosis
- Network evaluation
- Network documentation
- Training of network operators and administrators

Currently, most of these tasks are supported by different tools often from different vendors. The idea behind the INED editor is to provide an application independent network editor that could be extended by applications to support all tasks listed above.

In the following section we will describe the basic properties of the INED editor. Section 3 will show how applications can make use of it and section 4 discusses the current implementation status and presents some extensions we are currently working on. We conclude in section 5.

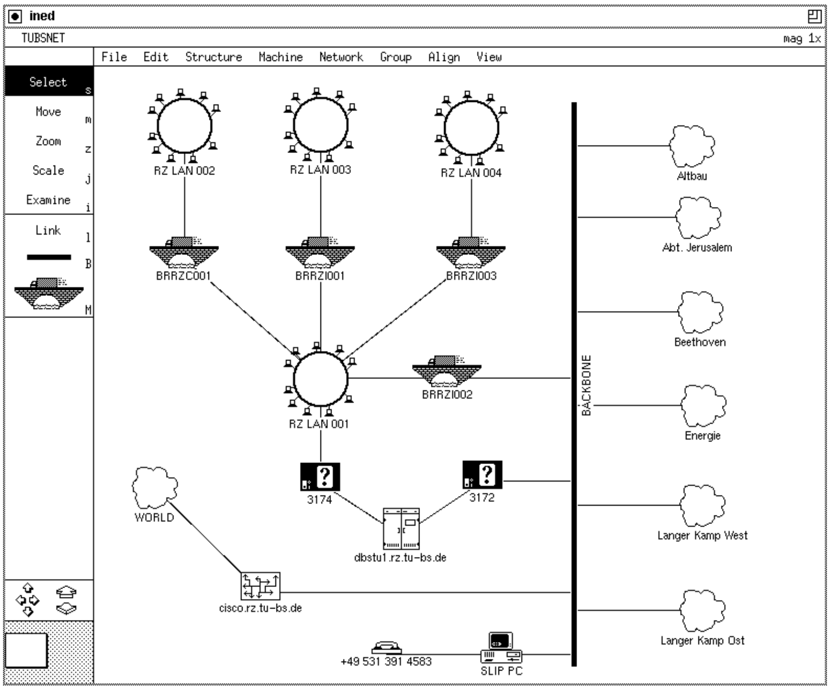

Figure 1: INED showing parts of our university network.

2 The Editor

Graphical network editors have been widely used in network management applications. Most of them are tied to specific tasks and users have to take the trouble to switch between different environments. This motivated us to develop a network editor that can be used by as graphical frontend for a wide range of applications.

In order to achieve this goal, we have compiled a set of requirements. Some of them are based on evaluations of network management systems [2, 3] while others are needed to guarantee flexibility:

1. The editor must provide basic editing features for network maps (selecting, moving, copying, cutting and pasting objects, undo and redo operations).
2. The editor must be able to maintain connections between objects.
3. The editor must be independent of any network type or network layer.
4. The appearance of network nodes must be user definable.
5. Hierarchical maps should allow the decomposition of a large maps with many devices.
6. Different views on the same network should be supported. Most common views are logical network maps and maps showing the physical location of devices. Other views representing a network grouped by network functionalities should be possible.
7. There must be an interface to include other graphics as background images (e.g floor plans) or legends.
8. There must be a simple interface that can be used by applications to act on the network map shown by the editor.
9. It should be possible to integrate applications without the need to build a new binary.

The development of INED was guided by these requirements. Figure 1 shows a screen dump of the editor showing parts of our university network. The editor is composed of a status line, a menu bar, a tool panel and a viewing area. The pull down menus contain commands to load or save maps, to cut, copy, paste and align objects, to select icons and to control the views. The tool box contains commands to insert networks, network nodes, and to link, move and select objects. The bottom of the tool box contains a panner to manipulate the visible part of the map shown in the viewer.

INED distinguishes between four object types. The first object type is a network shown as a line. The line thickness may be used to distinguish different network types and a label can be used to assign a name to a network. The second object type is a network node shown as an icon. A node can represent anything from a mainframe to a connector box, depending on the icons defined by the INED user. Every node may have a name (shown as a label) and a network address.

Connections between nodes and networks are represented by link objects. INED automatically keeps track of link objects when a node or a network is moved. The map shown in figure 1 contains a network labeled BACKBONE. On the bottom of the viewer are two nodes representing a PC and a modem. The PC is connected to the modem and to the network with two link objects.

The fourth object type supported by INED are group objects that can contain other objects (including group objects). A group can be collapsed to a single icon or expanded to show the objects contained therein. The contents of a group object (collapsed or not) can be edited in a separate view using the examine command. Figure 1 contains groups representing token ring networks and other subnetworks shown as clouds.

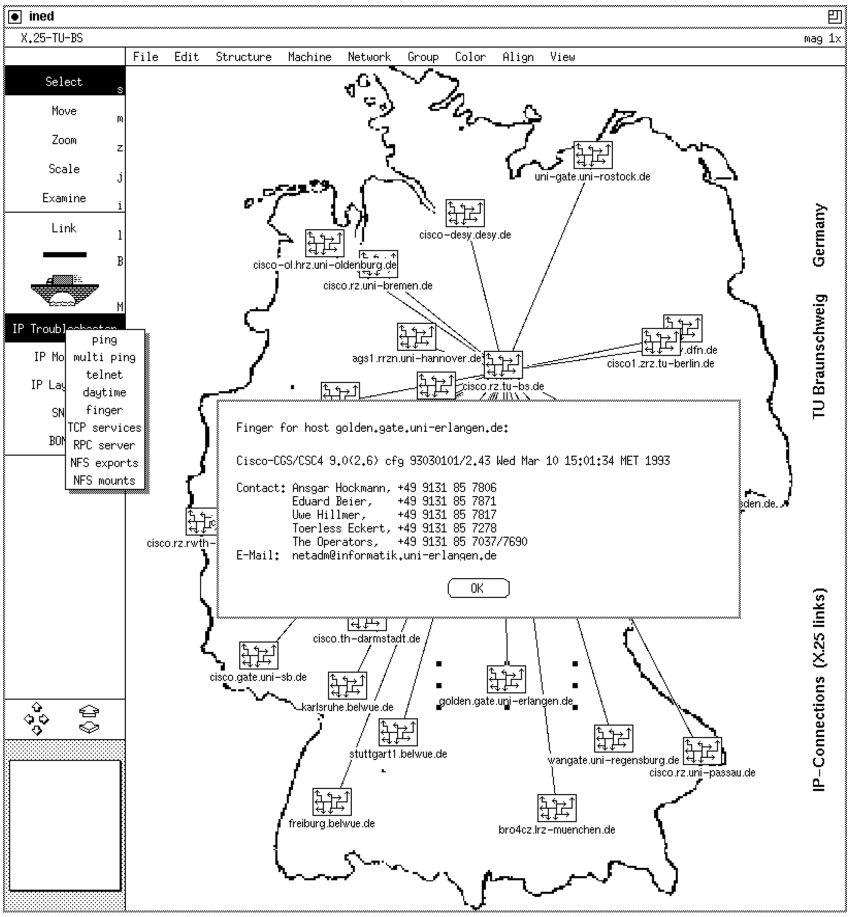

Figure 2: INED managing X.25 links of our university.

Figure 2 shows a background image loaded into our editor. A background image can be moved, scaled, copied and removed. All other operations on a background image will be ignored.

```
ined create NODE
ined create NETWORK
ined create LINK <id1> <id2>
ined create GROUP [<id1> <id2> ...]
ined delete <id1> [<id2> ...]

ined create TOOL <name> <command1> [<command2> ...]
ined delete TOOL <name>

ined retrieve <id1> [<id2> ...]
ined move <id> [<x> <y>]
ined icon <id> [<string>]

ined name <id> [<string>]
ined address <id> [<string>]
ined oid <id> [<string>]

ined acknowledge <title> <subtitle> <string>
ined confirm <title> <subtitle> <string>
ined request <title> <subtitle> <string>
ined browse <title> <subtitle> <string1> [<string2> ...]
```

Figure 3: The tcl interface.

3 The Interface

While INED is useful for documentation tasks, there is an interface that allows INED to integrate tools that can act on the network shown by the editor. This interface is based on the tool command language (tcl) [5]. Tcl is a small command interpreter which can be easily embedded into applications.

Applications that use INED as a graphical user interface run as separate processes containing an embedded tcl interpreter that is called whenever a user selects a command provided by the application. Communication between the application and INED is handled by a simple communication protocol that is completely written in tcl and could be loaded in any tcl interpreter. The communication protocol extends the tcl interpreter with commands performed by the INED editor (see figure 3 for more details).

A tcl interpreter can register commands by creating a new tool. A tool definition includes a tool name and the set of commands provided by this tool. The names of all registered tools are shown in the tool box. The commands of a tool will pop up whenever the tool is selected. Figure 2 shows the commands of a tool called IP Troubleshooter.

Execution of one of theses commands on a selection of the displayed network results in a call of a corresponding tcl procedure. The argument of this procedure is the list of the current selection. The four object types are represented as structured lists as shown in figure 4.

```
{ NODE <id> <name> <address> <oid> <list of links> }
{ NETWORK <id> <name> <address> <oid> <list of links> }
{ LINK <id> <id1> <id2> }
{ GROUP <id> <name> <oid> <list of contained objects> }
```

Figure 4: External representation of INED objects.

To demonstrate the usefulness of our approach, we have written a tcl interpreter called SCOTTY that forms the basis of some troubleshooting applications for TCP/IP networks. SCOTTY uses tcl extensions to access the ICMP and TCP protocols, to test various RPC services and to access the domain name

```
#!/usr/local/bin/scotty -nf
proc finger {list} {
    foreach comp $list {
        if {[lindex $comp 0]=="NODE"} {
            set host [lindex $comp 2]
            set ip [lindex $comp 3]
            set result ""
            if {[catch {connect $ip finger} f]} {
                ined acknowledge "Can not connect to $host."
                continue
            }
            puts $f ""
            while {! [eof $f]} {
                lappend result [string trimright [gets $f] "\r\n"]
            }
            shutdown $f all
            close $f
            eval "ined browse {Finger for host $host:} {} $result"
        }
    }
}
ined create TOOL "DEMO" finger
ined_main_loop
```

Figure 5: Sample script to finger all selected network nodes.

system. The output shown in figure 2 was generated by SCOTTY using the script shown in figure 5.

4 Implementation Status

INED is implemented in C++ based on the Unidraw framework [9, 8] of the InterViews toolkit [4] which provides a class hierarchy for the construction of domain specific editors. The current version has been developed on SUN SparcStations running SunOS 4.1.1 using gcc 2.3.3 and InterViews 3.1. It has also been tested on HP/UX Workstations running HP-UX 8.0 and the free LINUX operating system.

Work has just started to develop troubleshooting applications for TCP/IP networks. Therefore, we are implementing a tcl interpreter that provides access to the Simple Network Management Protocol (SNMP) [6]. We are also working on a discovering tool that will help to create and maintain network maps using information available from the network. We also work on the integration of a unix network administration package called BONES [7] and a network simulator called SIMULAN.

Experiences with the development of some applications have shown us that the current INED version has some restrictions that will be addressed by a future version:

- Networks are currently represented as either horizontal or vertical lines. We will support polylines in the future to better assist by the generation of wire maps.
- The tcl interface will be extended to allow access to additional graphical representations such as strip-chart icons.
- We consider to make arbitrary InterViews objects available to the applications using the tcl interface.
- We need a mechanism to control the interpreters from within the editor.

5 Conclusions

We have presented the basic properties of the Ined editor and how it interacts with underlying network management or system administration applications. The separation of the editor from the tools that act on the displayed network allows us to use Ined as a graphical frontend for different applications.

References

[1] A. Dupuy amd S. Sengupta, O. Wolfson, and Y. Yemini. NETMATE: A Network Management Environment. *IEEE Network*, pages 35–43, March 1991.

[2] H.E. Carter and J.P. Dia. Evaluating Network Management Systems: Criteria and Observations. In *Proc. International Symposium on Integrated Network Management*, pages 213–223, April 1991.

[3] J.L. Crawford. Graphics for Network Management: An Interactive Approach. In *Proc. International Symposium on Integrated Network Management*, pages 197–208, January 1989.

[4] M.A. Linton, P.R. Calder, and J.M. Vlissides. InterViews: A C++ Graphical Interface Toolkit. Technical Report CSL-TR-88-358, Stanford University, July 1988.

[5] J.K. Ousterhout. TCL: An Embeddable Command Language. In *Proc. Winter USENIX Conference*, pages 133–146, 1990.

[6] M.T. Rose. *The Simple Book – An Introduction to Management of TCP/IP based Internets*. Prentice Hall, 1991.

[7] J. Schönwälder and H. Langendörfer. Administration of large distributed UNIX LAN's with BONES. In *Proc. World Conference On Tools and Techniques for System Administration, Networking, and Security*, Arlington (Virginia), April 1993.

[8] J.M. Vlissides. Generalized Graphical Object Editing. Technical Report CSL-TR-90-427, Stanford University, June 1990.

[9] J.M. Vlissides and M.A. Linton. Unidraw: A Framework for Building Domain-Specific Graphical Editors. In *Proc. ACM SIGGRAPH Symposium on User Interface Software and Technology*, October 1989.