NetTrouble: A TTS for Network Management

Luís Santos, Pedro Costa, Paulo Simões
Communications and Telematics Services Group
Department of Informatics Engineering
University of Coimbra - 3030 COIMBRA, Portugal
E-mail: lsantos@dei.uc.pt, p-jose-costa@telecom.pt, psimoes@dei.uc.pt

Abstract - Present information systems need tools that are able to register the variety of problems that occur in the system, co-ordinating the efforts of resolution and keeping the experience gained in those efforts. These applications are traditionally called Trouble Ticket Systems (TTS). Basically, a TTS system is expected to operate as "a hospital chart, co-ordinating the work of multiple people who may need to work on the problem" [1].

At the moment there is a great number of tools corresponding to this TTS definition. Since these are usually developed for specific environments, such as project development, hosts administration and helpdesk, they have very specialised information and administrative models, lacking flexibility when applied outside their native domains. Maybe that's why they still can't get satisfactory results in systems and network management (NM), though their good results in other environments. NM environment involves several issues not yet addressed by current systems, namely the heterogeneity of data networks, its geographic dissemination (of the managed system and of the management team), the multi-organisational attribute of wide area networks (with many carriers and communication service providers between network ends), and the decomposition (both functional and hierarchical) of the associated tasks and responsibilities.

This paper explains the most important results of a project developed at the University of Coimbra that involved the development of NetTrouble: a TTS designed for NM that brings out new and innovative concepts such as the use of administrative domains (useful to deal with multi-organisational systems), the autonomic definition of co-operation policies between different organisations, a flexible administrative model (going to ticket-level detail and able to involve both local and remote human resources), and the virtual presence of tickets belonging to remote domains. The notification model of NetTrouble, another interesting feature, will also be described.

1. INTRODUCTION

The TTS concept appeared in the early 80s with large databases designed to schedule, dispatch, and maintain job tickets. The main purpose of these systems was to co-ordinate workgroups in a simpler and more productive way. Today, the concept and aims of TTSs remain nearly untouched. The most visible changes were the deployment of the concept trough new application areas and the use of new technologies.

1.1. Present Systems

The analysis of present systems [2,3,4,5,6] reveals that it is possible to identify common points that somehow characterise a TTS, even if there is no normalisation work in this area. The typical TTS structure and information model will be presented in the next subsection.

1.1.1. Structure

The usual TTS structure has three main modules: the database (DB) where trouble tickets are kept, a middle layer mediating the access of final users to the DB, and the front-end providing the user interface (fig. 1).

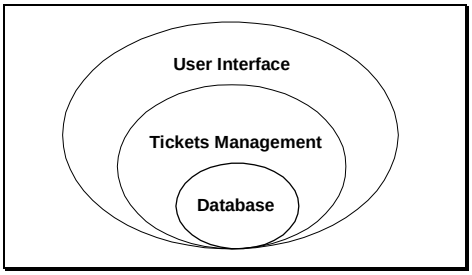

Figure 1: Typical Structure of a TTS

It is the intermediate module that lodges the most critical units of the system: the administrative model and the functional model. Together, they rule the interaction between the main entity — the ticket — and the system users: the first defines the possible intervention ways of each team member, while the second assures a coherent path in the ticket resolution process.

1.1.2. Information and dataflow

The majority of existing TTSs are "closed systems" with centralised databases, even if some of the packages have the user interface untied to the DB, therefore allowing remote access across the network [2,4,6] or by e-mail [3]. The "trouble ticket", the core

entity of stored information, is usually composed of two main fields: the header and the body. The first has common attributes, while the second allows flexible storage way of specific information related to the problem it represents.

It is also possible to point out some common aspects for the information model of current systems: after trouble tickets registration, human resources are linked to them. The system then co-ordinates work at progress, hopefully converging to the resolution of the problem. When the problem is considered as solved, trouble ticket life ends and all its information is cleaned from the database (most frequent case) or stored as acquired experience (solution provided by sophisticated systems).

1.2. Network management specific needs

In RFC1297 [1], Johnson summarises a set of requisites that a NM oriented TTS should include. In a general way, he points out the importance of integration with other NM tools and selects some desirable characteristics of the associated administration, operational and the information models.

At a first glance, only at low level we can find out some inadequacies between present solutions and the requisites referred by Johnson. In other words, it would suffice to make small changes to existing systems, not affecting inherent concepts, to achieve good results. However, a deeper look will prove that such approach is clearly insufficient, since some of the NM specific issues are not properly addressed by current TTSs.

1.2.1. Geographic scattering

Apparently, existing TTSs where the user interface is separated of the DB [2,6] could satisfactorily cope with the geographic scattering of medium and large organisations. However, their centralised DB would have to be located at a single point. This results in higher access times, latency and overloaded lines when accessing the DB from remote locations, even if the accessed ticket regards a “local” problem (i.e., concerning only the remote location).

1.2.2. Multi-organisational nature

The multi-organisational nature of certain metropolitan and wide area networks, where several service providers operate between the network end points, demands new concepts in the TTS area. Management is now handled by several organisations participating in the process with different roles. Current systems lack support for communication and co-ordination between TTSs of different organisations.

1.2.3. Hierarchical and functional decomposition

In order to maximise group efficiency, functional division of the NM activity usually assumes an hierarchical structure of responsibilities. The tools provided by TTSs to cope with this decomposition are either not existent or too inflexible.

1.3. Need for a new approach

Since current TTSs do not perform well with these and other specific issues of NM environment, *NetTrouble* development tried new approaches to the problem:

- to solve the problems introduced by geographic scattering, *NetTrouble* proposes a distributed database concept where information is spanned over all the branches of an organisation;
- the concept of administrative domain was introduced to deal with the multi-organisational nature of network management environment;
- the *NetTrouble* administrative model also gives specific support to hierarchical decomposition.

Despite of the introduced innovations, traditional concepts were not forgotten: requirements proposed by [1] were addressed and even further explored (e.g. the category concept and its administrative usage).

2. NETTROUBLE MAIN CHARACTERISTICS

This section will begin with the discussion of the most relevant keystones of *NetTrouble*. Specific aspects, considered innovative and interesting (even if not so important for the overall system), will later be presented.

2.1. General aspects

2.1.1. Administrative Domain

The **administrative domain** concept was introduced as a way to deal with the multi-organisational nature of NM environments. An administrative domain is a self-sufficient cell supported by one *NetTrouble* server. This server keeps local “trouble-tickets” and manages local human resources. It may also communicate with external domains (i.e., with external *NetTrouble* servers) in order to share resources. Co-operation policies with the exterior are defined by specifying the set of co-operative domains and, for each of them, selecting the resources to be shared in order to solve common problems.

Although an administrative domain is usually the representation of an organisation, it can also represent

internal cells (within a single organisation) separated geographically or administratively. Furthermore, it can be used to reflect hierarchical levels of competence in the management of large metropolitan networks.

2.1.2. Client-Server Architecture and the Communication Protocol

The *NetTrouble* framework entails two distinct contexts of communication: the communication between an user and its domain server, and the communication between domains. A common solution was developed for these two contexts, using the client-server model.

The server keeps the DB and ensures proper access to it. On the other hand, the client will be any entity that needs to work on the trouble tickets kept by the server. The client might be a local user (using specific *browsing* software) or a peer domain (i.e. another server). A communication protocol was specified for *NetTrouble*. This protocol (with just two primitives: *data_request* and *data_set*) should allow, in the future, communication with other systems (e.g. different TTSs or the alert systems suggested by [1]).

2.1.3. Distributed Information

The NetTrouble servers create a web where each node has complete autonomy to specify which trouble tickets and which human resources it wants to share with each organisation. Each ticket remains in a single administrative domain but, if it involves more than one organisation (a common fact in network management) it will be virtually present in the other involved domains. This way, each organisation has access to all the "interesting" information, even if it is not locally kept. Thus, it is possible to assemble a task force to solve a given problem including managers from distinct domains. This **information and resource decentralisation** was achieved by the **forward mechanism** (fig. 2).

After contact, the server determines the type of client issuing the request. If the client is a server (another domain), it answers only to requests about local data. On the other hand, if the client is a local user, it analyses the scope of the request. When external information is requested, the server forwards the request to the domains that keep the information (D_2,..., D_n). Then, it concatenates replies in a single answer returned to the original client (answer= $answer_1$+ $answer_2$+...+$answer_n$).

It must be noted that the access to the forwarding mechanism comprehends only the domains for which the user manager in question has been exported by the local domain.

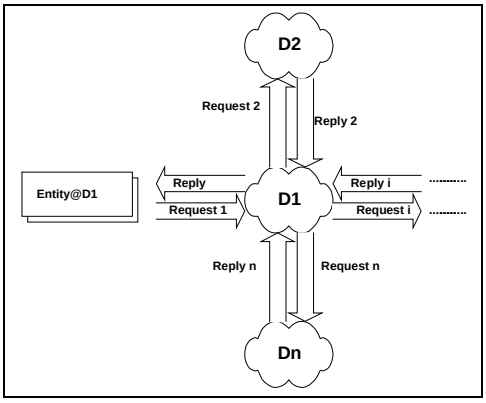

Figure 2: Forward Mechanism

2.2. Specific aspects

Following, the main aspects of the internal structure of the *NetTrouble* are presented. The different entities supported by *NetTrouble* will be presented prior to the description of the most relevant concepts associated to the information core: the "trouble ticket".

2.2.1. NetTrouble entities

NetTrouble supports the existence of several managers for the domain. This is accomplished using three different kinds of administrative entities (fig. 3).

The **manager** entity represents any technician of the organisation. All interactions with the system will be triggered by an entity of this kind, even in interactions across domains, where this entity behaves as a domain entity (due to the forward mechanism).

The **manager group** provides a grouping mechanism used to simplify the management of local domain human resources. It makes possible to group individual managers (or even other groups) under a single identification, thus allowing to delegate competencies and responsibilities at group level.

Finally, the **domain** entity represents the concept of administrative domain. This entity is very important for the forward mechanism, since forwarded requests will be made in the name of the local domain (and not the local user). This avoids the need to register local users in external domains.

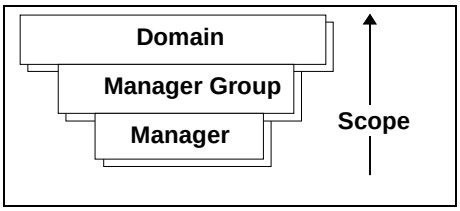

Figure 3: NetTrouble Administrative Entities

2.2.2. Problem stages

Every problem tracking system has a continuously growing database keeping information on problem proposals, current problems and, sometimes, information on problems already solved (therefore acting as a knowledge base for future solving processes).

From the functional point of view, the evolution of a problem presented to *NetTrouble* is summarised in Figure 4. Different stages are justified by the different demands posed by the problem solving process (at the information level and in terms of involved human resources). This stages will be described after the discussion of an associated concept, the **categories**.

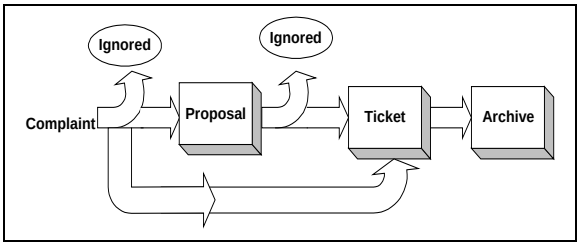

Figure 4: NetTrouble Problem Evolution

Categories

Few of the analysed TTSs support the problem categorisation feature [2] ensured by NetTrouble, where each problem is included in one of the existing categories at creation time. This is a way of organising incoming information. The category scheme of NetTrouble has an hierarchical structure (a set of inverted trees) reflecting the usual problem classification by type and specific aspects (fig. 5).

This concept was further extended: management of a category is delivered to a local manager, called, in this context, the **category responsible**. Obviously, a manager may be responsible for more than one category. The responsible for a given category may divide it in several subcategories and delegate some of them to another manager.

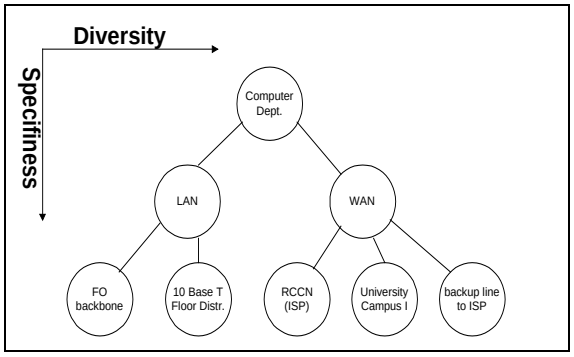

Figure 5: Hierarchical Structure of Problems Categories

These mechanisms allow the definition of an hierarchical structure close to the real structure used in many organisations.

The proposal

A problem is generally detected by a failure or a lack of functionality of some management object (a network branch, a router, a printer, a server, an application,...) [1]. Therefore, the problem gets to the TTS as a complaint. *NetTrouble* designates this complaint as **proposal** since it will, potentially, become a "trouble ticket", though not yet treated as one. The distinction between proposal and trouble ticket allows the temporary storage of several symptoms that will, probably, produce a single ticket [1] (usually after exhaustive diagnostic).

The Trouble Ticket

The "trouble ticket" (or simply the "ticket") is the evolution stage where the problem resolution will take place. Therefore, this is the stage where greater demand will be posed to the system.

Information supported by the ticket — most TTSs organise the trouble ticket information in two major fields: the header and the body. The header contains a fixed set of fields with the information pieces common to all the tickets. On the other hand, the body (also named description) has a flexible format where it is possible to describe in a flexible way all the resolution process.

The developed system also adopted this structure, keeping in the header fields like *identifier*, *title*, *category*, *state* and *priority*. In the description field, *NetTrouble* takes each contribution (with freeform) as a distinct piece of the global problem description. Each of these pieces includes an author signature (thus enforcing process credibility) and a time stamp. It should be noted that NetTrouble can generate contributions by itself when relevant header fields are changed (further increasing the process credibility).

Associated classes — the access control to the ticket information is based on two mechanisms: classes and masks (extending the effect of the classes).

The **responsible** class represents the manager in charge of that ticket (implicitly, the manager responsible for the category of ticket). The responsible has several tasks: management of the ticket information; assignment of human resources to solve the problem; and the definition of the access rules for the other classes. The **involved** class represents the team selected by the responsible to solve the problem. This class may include managers

from local and remote domains. Entities inside this class have full reading access to all the ticket information and also permission to add descriptions to the ticket body. Full access to header fields (and even management of lower classes) can be granted by the responsible by the use of masks.

The **audience** class allows to give some entities permission to read the ticket information. At the bottom of the hierarchy, the **public** class represents extern people somehow affected by the problem. This last class can include people not registered in the TTS, since it relies on e-mail to contact its members.

Each ticket has one instance of each of this classes, and a given entity will have access to a given ticket if it is included in one of its classes (fig. 6). The specific access permissions for that entity depend on the class it belongs too and, eventually, on its access mask.

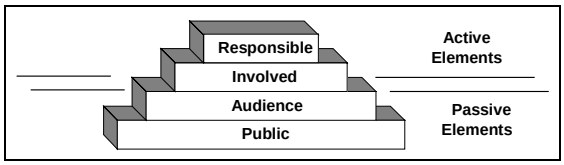

Figure 6: Class Hierarchy

Functional model of the Ticket — the resolution of the problem is a free process. Nevertheless, *NetTrouble* links to the ticket an attribute — the *state* — to describe the current situation (possible values: open, suspended, blocked, solved or archived). But the role of this field isn't only descriptive. This state is also used to manage the ticket evolution.

Associated priorities and time-out mechanism — unlike usual systems, where the *priority* field is merely informative, *NetTrouble* assigns a timer to each ticket (the period of time depends on the category and priority of the ticket), thus implementing the **time-out** concept suggested by [1]. If by the end of the defined period of time the ticket remains unchanged, the responsible will receive an e-mail notification.

Link mechanism — a failure can easily result in several complaints apparently not related until a more advanced stage of its resolution. Therefore, several tickets might be directly tied to a single problem. As soon as this is detected by the manager, it is useful to unify the resolution of these tickets. To make this possible, *NetTrouble* allows the establishment of links between tickets. There are three types of links, allowing a ticket to:

- accept to be consulted by entities from another ticket;
- express to other ticket the intention of being able to consult its information;
- temporarily block the resolution process of another ticket.

This last feature allows managers to define **precedences** in problems resolution, thus avoiding unnecessary or even conflicting efforts.

Integrated notification mechanism — some of the analysed TTSs have a notification scheme based upon e-mail. Nevertheless, this scheme is always presented as a feature associated with the *browser*, not with the TTS. *NetTrouble* supplies an integrated notification scheme that makes possible to send a message to a specific class of the ticket. This is the **mail to a ticket** concept (fig. 7).

NetTrouble joins, as the subject prefix, the universal problem identification (LocalIdentifier@Domain) and, as a message prefix, the complete signature of the sender. It is also important to point out that it is through this notification mechanism that becomes possible to contact the public class of a ticket.

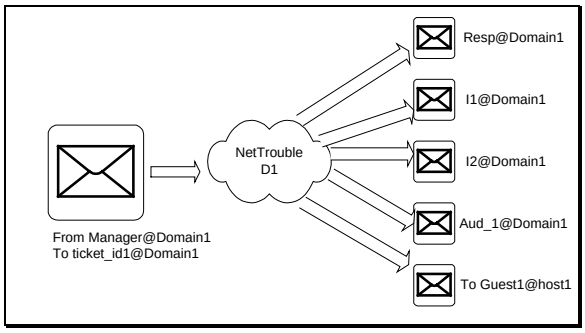

Figure 7: *mail* to a Ticket

The archive

The ticket ends out its active existence when the responsible changes its state to *archived*. As soon as this happens, all the administrative information is lost. On the other hand, the information of the *description* field remains available for statistical treatment and future consult, acting as a "knowledge base" [1].

3. APLICATION FIELDS

The client-server paradigm and the used administrative model allow *NetTrouble* to adapt to the most common NM environment. In order to show possible application fields for *NetTrouble*, two typical scenarios will be presented bellow.

3.1. Horizontal structure

When several service providers operate between two organisation branches (fig. 8), network management is ensured by different and independent entities that

need to work together to solve common problems. A *NetTrouble* server (a domain) can be placed in each organisation. Each intervening domain will have the opportunity to make available human resources to the exterior (possibly to a specific set of domains). This way, it is possible for two or more organisations to co-ordinate their work on common problems with base on trouble tickets.

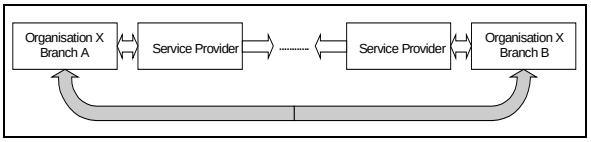

Figure 8: Multi-organisation Horizontal Structure

3.2. Hierarchical structure

Also very frequent is the case where network management assumes a hierarchical nature (fig. 9). This situation can also be fit by the developed system, placing a NetTrouble server in each of the nodes. Relationships between a section and the exterior are processed through a single channel with its upper node, and through multiples channels with subordinated sections.

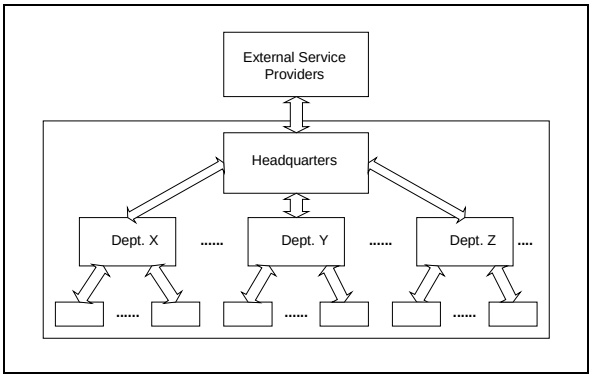

Figure 9: Hierarchical Structure

4. NETTROUBLE IMPLEMENTATION

In order to test the concepts and orientation lines presented in the previous section, NetTrouble was fully implemented and made available as public domain software [8].

The present implementation consists of a client, a server and an application programmers interface to simplify the development of clients for other architectures. The server runs in OSF/1 (using Oracle for database purposes) and the client was developed for SunOS (X-Windows/OpenLook environment). Network communication uses ONC-RPC [7], making very easy the development of clients for other platforms.

5. CONCLUSIONS AND FUTURE WORK

This project addressed the development of a TTS well suited to NM environments. Several new concepts were introduced, like the administrative domain concept; the autonomous way of defining co-operation rules between domains; a flexible administrative model; and the decentralised access to information and resources.

It is also important to remember that *NetTrouble* capabilities are not confined to problem management: it could be used as a job ticket system (in project management) or even as a news ticket system, due to its capabilities of organising thematic discussions.

At the conceptual level, there were some unsolved problems (like the management of links between tickets belonging to different domains) that could probably be handled using distributed database techniques. It would also be interesting to improve the specification of an open protocol for interdomain communication.

From a more technical point of view, web-based communication between the TTS and the users would enhance portability and accessibility.

6. THANKS

This work is part of the FADA project, partially sponsored by Junta Nacional de Investigação Científica e Tecnológica.

7. REFERENCES

[1] D. Johnson, “NOC Internal Integrated Trouble Ticket System Functional Specification Wishlist (NOC TT Requirements)”, Request For Comments 1297, 1992.
[2] “Problem Tracking System 1.05”, Zombie Software, 1994.
[3] “GNU GNATS (GNU Problem Report Management System) 3.2”, GNU.
[4] “Razor 4.0a”, Tower Concepts, Inc.
[5] “Under Control”, KJT Software, Inc.
[6] “PR-Tacker 1.2”, Softwise 1994.
[7] Sun Open Network Computing (SVR4).
[8] NetTrouble Package, http://cleo.uc.pt or http://www.dei.uc.pt/

Publication Information:
L. Santos, P. Costa, P. Simões, "NetTrouble: a TTS for Network Management", Proceedings of ITS'98 (SBT/IEEE International Telecommunications Symposium), São Paulo, Brazil, August 1998.