The First Results of Software Redesign of DKTS 30 Switching System to Support Extended Capacity

Milan Jovanović[1], Branko Kolašinović[1], Mirko Markov[1], Dimitar Komlenović[1]

Abstract – **The first results of work on the software redesign projects of the DKTS 30 switching system to support extended capacity were presented in this paper. The extended capacity switching system had been compared with the original DKTS 30 switching system, and some of the obtained results were given in this paper.**

Keywords – **switching system, software redesign, interprocessor communication.**

I. INTRODUCTION

The DKTS 30 public digital telephone exchange [1] is the newest product from the series of DKTS digital telephone switching systems. Although it has been successfully commercially exploited since 1999, it is still under development in order to:

- achieve better quality,
- provide new services and
- reduce the production price.

Like other modern telephone switching systems it is based on a large number of off-the-shelf microprocessors and microcontrollers, that frequently need to communicate with each other. That means that from this paper's point of view the DKTS 30 system represents distributed heterogeneous multicomputer that works in real time.

The simplified DKTS 30 system architecture was shown in Figure 1. The goal was to show only blocks taht are necesary to understand the presentation to follow. The system consists of central blocks and peripheral blocks. Central blocks are: administration (ADM), switching (KOM), synchronization (OSC), and UCP (*Universal Central Processor*). The UCP unit distributes the messages among the central and peripheral blocks. In order to increase the system reliability, the central blocks are duplicated. The central blocks are connected via a local Ethernet, which is doubled, too. All central blocks, except UCP units, are connected to both Ethernet networks.

The peripheral blocks are connected to UCP blocks via serial HDLC links. One pair of UCP units works in the load-sharing mode for a group of six peripheral units. Each of those six peripheral units is connected to to each of two UCP units by its own separate link. The peripheral blocks (PB) are subscriber blocks and interexchange trunks.

One of the main challenges facing DKTS 30 software engineering is the variety of microprocessors (Motorola 68360 and Intel family are present, processors from the PowerPC family are planned for future use), as well as the variety of operating systems that run on different parts of the DKTS 30 switching system (WinNT, Linux, pSOS, RTEMS).

[1] PUPIN TELECOM DKTS, Batajnički put 23, Belgrade, Serbia
E-mail: { milanj, brankok, mmarkov, dimitark}@ dkts.co.yu

The DKTS 30 software is based on object-oriented principles [2]. It was developed using UML notation [3] and standards for software projecting. The Rational Rose CASE tool and C++ language were used. The software is organized hierarchically with layers. Each layer provides a service to the higher layer, and simultaneously it is a client of the lower layer. Also, the software is organized as a collection of server objects that are distributed. Main software abstractions are modeled by these server objects. These servers are implemented as finite state machines (FSM). This is a common approach in design of embedded real-time systems [4]. Each FSM is designed according to the *Bridge* pattern [5], and consists of an interface and an implementation object. Interface and implementation objects may reside on different processors, and the only connection between them is their unique identifier of the object.

Software that runs on peripheral blocks is differently organized. In order to run faster, it is written on machine language and standard C language. It runs on an operating system originally developed by Iritel institute.

II. SOFTWARE REDESIGN PROJECT

The DKTS company has been slowly reorganized in order to become an organization based on projects. Using appropriate software tools: project plans are made, reports are written, sucessfulness of project realization is monitored, etc. A team is formed for each planned project. Members of one team can work on more differnet projects at the same time.

The maximal capacity of the original DKTS 30 switching system was 15872 subscribers. However, this number was shown to be insufficient with the uprising market demands. Therefore, the project of extending the capacity was undertaken. The project goal was to provide the maximum capacity of 174592 subscribers. The hardware changes were described in [6], while the planned software changes were described in details in [7]. This paper is dedicated to the first results of the software redesign project, so the planned and done hardware and software changes were described only to understand the presentation to follow.

This project includes: a redesign of the system image, a redesign of the interprocessor communication, more natural integration of the remote subscriber unit into the system, algorithms for distributed system supervision, appropriate changes in the system data base and the graphical user interface.

The project requirement was not only to trivially provide capabilities for a larger number of blocks in the system, but also to pay specific attention to the system performance. It was highly unlikely that the algorithms that were shown to be efficient with one traffic load would be efficient in the same way with much heavier traffic load.

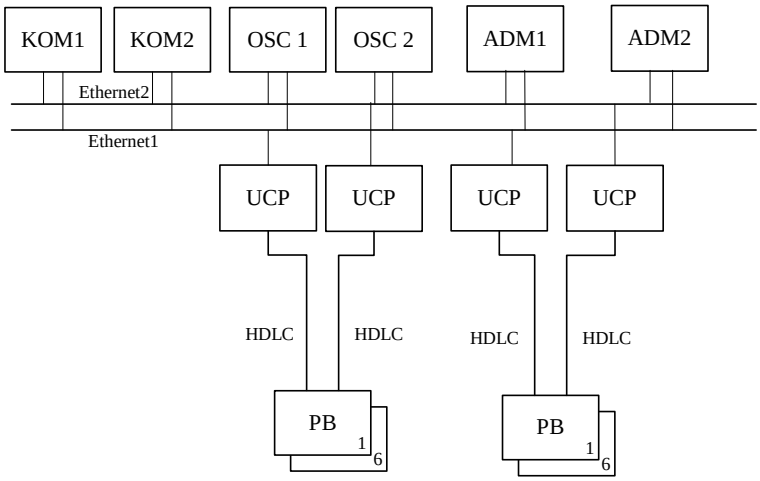

Figure 1. A simplified DKTS 30 system architecture

Concerning the software compatibility, it was decided that the new software has to be compatible with the original hardware, but not with the software of the original solution. That means that the new software can be executed on all DKTS 30 platforms, but that it can not be mixed with the old software.

The goal was to move beyond some obstacles that were present in previous software solutions due to required compatibility with the DKTS 20 system, which was put aside in the meanwhile. Special attention was payed to the intermediate period in order to make it possible to add new funcionalities to the software of the old system as well as to the software of the system with extended capacity.

Because of the decision to abandon compatibility with DKTS 20 peripheral blocks, it was convenient to undertake a redesign of the system software on peripheral blocks, which was shown to be a complex task contributing to the delay of the entire project.

Because of the large number of different processor types, different hardware platforms, various operating systems, situations in which it is necessary to perform potential error corrections in the field and in laboratory conditions, concurrently with adding new funcionality to the fisrt DKTS 30 system, a great attention was payed on developing methodologies for software testing.

As a result, a set of tests was designed. The set comprised different testing techniques: from the partial to the integral testing, with and without the use of call generator, etc. In the testing phase, the use of error information is increased. However, the great part of this code will be excluded from the final version in order to assure executable programs with the minimal time of execution. In addition to this, finite state machines whose only purpose is testing are designed. The FSM whose only task is to generate desired traffic is instantiated on every processor. This is an example of how interprocessor communication is tested.

III. APPLIED SOLUTIONS

From the software soultions realized in the project of extending capacity that concern this paper's topic, the most important ones are those that concern interprocessor communication. The base request of the redesign of the interprocessor communication was to decrease the number of messages needed in the operation of the switching system, as well as to additinaly incerease reliablity of the interprocessor communication. In order to fullfil these requirements, it was necessary to make some changes in a message header. The message header remained of the same length as it was. It contains the same fields as in the previous implementation: message type, source and destination internet address, message identification, destination object's identification. However, some changes were inevitable. First of all, the format of internet address had to be changed. Next, the way message type is marked was also changed. As a result of these changes, incompatibility with the previous software implementation became unavoidable. Among other things, with the new message caracterization, it became possible for clients in upper software layers to supress acknowledgement messages on the protocol layer when functional messages are used to acknowledge an application layer acknoledgement request.

In the aim of acchieving faster interprocessor communication, the SP protocol was abandoned. The SP protocol was used for communication between peripheral and UCP blocks via HDLC links [8]. The reason for the presence of this protocol in the previous implementation lay in requested compatibility with DKTS 20 peripheral blocks. Since the concept has been abandoned in the meanwhile, the NLC layer of the interprocessor communication protocol stack was eliminated. In adition to this, another historical role of the UCP board, which also existed due to requested compatibility with DKTS 20 peripheral blocks, became unnecessary in the new software environment. That was the NLB layer whose task was to perform message format conversion between DKTS 20 and DKTS 30, and vice verse.

The design of the first DKTS 30 switching system did not support communication between UCP boards belonging to one Ethernet with UCP boards that belong to another Ethernet. However, during the exploitation phase, the need for communication between peer boards was brought to attention. In addition to this, there was the need to put interprocessor communication under software supervision. Accordingly, it was decided to correct this defect by allowing software

routing from one Ethernet to another. Central blocks that have available network interfaces belonging to both Ethernets became potential routers. It was more than likely that this task would be committed to the OSC board, since the OSC board is the processor that is not overloaded with interprocessor communication responsibilities. The routing is now supported in system image. In addition to everything else, this routing may be used in the case of a failure of a network interface on boards with two network interfaces. For example, in previous software versions, the KOM board with an inactive network interface that belongs to the Ethernet A could not send a message to a UCP board connected to the Ethernet A.

A reduction of the number of messages passing through the system is acheived by the use of multicast techniques [10]. Instead of sending a large number of single messages to different destinations, it is now possible to send only one multicast message that is received by all procesors that are members of the specified multicast group. The great problem was the presence of several operatin system in the DKTS 30 switching system, each of which providing similar (but different) programming interface towards the multicast facilities. In addition to this problem, it was important to provide that all processors in the group receive a multicast message, but also to avoid duplicated messages, which may be the consequence of the use of alternative routes. Periodical updating of local image, sending information on block or network interface failure or activation, periodical checking of block states are cases in which it is evident that significant improvements in speed and efficiency are achieved.

IV. CONDITIONS OF THE ANALYSIS

There is a model of DKTS 30 switching system dedicated to extending capacity project in the research and development laboratory of DKTS. Although this model is rather modest comparing to the maximal capacity switching system, it has been shown to be adequate for phases of develpment and testing accomplished until now. All experiments to be described have been conducted on it. This model consists of 2 ADM units, 2 KOM boards, 2 OSC boards, 4 UCP boards, 4 and 4 clasical subscriber blocks, 2 interexchange trunks, and one extended subscriber block. There is a substantially better equipped model of DKTS 30 system also dedicated to extended capacity project in the DKTS testing department.

A call simulator *Anritsu* EF111A was used during the testing phase. The calls are established via 4 subscriber blocks, 3 of witch are connected to one pair of UCP boards, and the last one is connected to other pair of UCP boards. Due to limited number of available cables, only 24 dual telephone connections are formed, divided into 3 groups with 8 connections per each. To cover all cases of interest for testing, the first group consists of connections between subscribers that are connected to the same subscriber board, the second group consists of connections between subscribers that are connected to two subscriber boards that are on the same pair of UCP boards, and the third group consists of connections between subscribers that are connected to two subscriber boards that are connected to different pairs of UCP boards.

An asinchronous mode of establishing connections was chosen. The generated telephone traffic was controlled by call simulator parameters: duration of a call (path hold time) and the time between the end of one call and the beginning of the next call (release time)

V. RESULTS

Some results of accomplished comparison of the extended capacity *s*witching system (ECSS) and the original DKTS 30 switching system (OSS) were presented in this section.

The first experiment is related to software download to peripheral blocks: a clasical subscriber block (cSB), an extended subscriber block (eSB), a clasical interexchange trunk (cIT), and an interexchange trunk with SS7 signalization (sIT). The sizes of files to be downloaded are practically the same for both variants.

TABLE 1
SOFTWARE DONWLOAD TO PERIPHERAL BLOCKS

Peripheral unit	Number of units [512B]	OSS [s]	ECSS [s]
CSB	257	11	6
ESB	436	18	9
CIT	513	21	11
SIT	692	29	14

Table 1 shows that significant improvements were fulfilled. The software download to periperal blocks in the ECSS vatiant is even two times faster. The main reason for this improvement is abandonment of SP protocol that was used in communication between peripheral blocks and UCP blocks. The number of messages needed to download the sotware to peripheral blocks was significantly decreased in this way.

Besides, by a change of the download alghoritam, although it is still centralized, with the master administration in charge, additionaly was decreased the number of messages needed to download the software to peripheral blocks. The obtained improvement of software download to peripheral blocks enables significant decrease of startup time of the whole switching system during a cold start.

A new realization of switching system supervision also brings a significant improvement. A notification of all blocks in the system about a change in the system was given as an example (Table 2). One has to notice that functional answers to this notification messages are used to check a state of the blocks. Three variants are compared: original (centralized) solution (OS), original (centralized) solution with multicast to central blocks and unicast to peripheral blocks (MC), and distributed solution with multicast to central blocks where UCP boards have a role to notify and monitor peripheral blocks that are connected to them (DS). An average response time was given according to observing of the Ethernets and it is calculated from an appearance of the first message of one notification phase until the reception of the last response message of that phase. The number of messages on Ethernet is given for the maximal capacity original switching system, what means the system with 128 peripheral blocks. The

switching system model in the testing department was used for this experiment.

TABLE 2
NOTIFICATION AND POLLING OF BLOCKS

Verion	Number of messages on Ethernet	Average respnse time
OS	708	600 ms
MC	612	500 ms
DS	90	350 ms

The advantage of distributed solution compared to centralized solutions is clear. The benefit of using multicast has not to be neglected, although it is not as significant as it was expected, because of higher number of peripheral blocks compared to number of centralized blocks.

Encouraging results were also attained by testing the switching system by the call simulator (Table 3). During the testing of the original switching system each subscriber block was loaded with 2500 calls per hour (above that some problems are generated). The number of unseccesful calls of 500000 generated calls is about 1000, that means 0.2%. During the testing of extended capacity switching system each subscriber block was loaded with 6000 calls per hour. The number of unseccesful calls of 500000 generated calls is about 25, that means 0.005%.

TABLE 3
TESTING WITH CALL SIMULATOR

Version	Number of calls per subscriber block	Percent of unsecesful calls
OSS	2500	0.2%
ECSS	6000	0.005%

It was shown that discarding the compatibility with DKST 20 perihperal blocks, what means abandonment of SP protocol that was used in communication between peripheral and UCP blocks, as well as abandonment of conversion of messages on UCP boards, significantly increases throughput power of peripheral blocks, along with improved stability. According to paper [11], this means that it was reached almost the maximal traffic on peripheral blocks that KOM boards with current swithcing system architecture and organization can persist without saturation.

VI. CONCLUSION

A flexible base for the DKTS 30 switching system was provided by the software redesign project, although the main goal was to increase the capacity of the system. The first results of the work are encouraging, as was shown in this paper.

Significant improvements were fulfilled concerning software donwload on peripheral blocks. Even two times faster is download of software in the extended variant comparing with the original variant. The obtained improvement enables significant decrease of startup time of the switching system during a cold start.

A lot has been done to decrease the numer of messages that propagate through the system (SP protocol was abandoned, using of multicast, unnecessary response messages both on protocol and functional level were discarded). This, in the last instance, makes the response time to certain events significantly shorter. So, considerable improvement was fulfilled in the process of notification and polling of blocks in the swithcing system. The advantage of distributed solutions compared to centralized solution is clearly shown even for the original switching system with maximal capacity.

By abandonment of compatibility with DKTS 20 systems, the interprocessor communication with peripheral blocks was refined. Testing of the switching system by call simulator shows increased throughput power of peripheral blocks with new software, along with substantial improvement of work stability.

Because of not small complexity of the software redesign project in order to extend capcaity of digital public switching system DKTS 30, parallel work of the project members on different development projects which priority have been changed during the time, situations in which it is necessary to perform potential error corrections in the field and in laboratory conditions, concurrently with adding new funcionality to the first DKTS 30 system, there was no enough time to compare newly realized solutions with the original ones. As the project approaches its end, there will be more time to analysis and comparing of original DKTS 30 switching system with the switching system with extended capacity.

REFERENCES

[1] Jovanović M., Šuh T., *System DKTS 30 Main Characteristics, Telfor 1997*, Beograd, 1997.

[2] Booch G., *Object-Oriented Analysis and Design*, Second Edition, Benjamin-Cummings, 1994.

[3] *UML Semantics*, Rational Software Corporation, 1997.

[4] Selic B., Gullekson B., T.Ward P., *Real-Time Object-Oriented Modeiling*, Willey Professional Computing, 1994.

[5] Gamma E., Helm R., Johnson R., Vilsides J., *Design Patterns – Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.

[6] Laketa S., Vidić P., Nikolić N., "Povećanje kapaciteta sistema DKTS", *Telfor 2003*, Beograd, 2003.

[7] Kolašinović B., Komlenović D., Jovanović M., "The Redesign of the Software of the DKTS 30 Switching System to Support Extended Capacity", *ICEST 2004*, Bitola, Makedonia, 2004.

[8] Jovanović M., Hiršl V., "Međuprocesorska komunikacija u telefonskoj centrali DKTS 30," *YU INFO 1999*, Kopaonik, 1999.

[9] Vujadinović D., Jovović Ž., "Funkcionalna konverzija poruka u sistemu DKTS20/30," *IT'98*, Žabljak, pp. 64-66, 1998.

[10] Deering S., "Host Extensions for IP Multicasting," *RFC 1112*, 1989.

[11] Markov M., Kolašinović B., Jovanović M., "Merenje opterećenja nekih resursa telefonske centrale DKTS 30 softverskom simulacijom poziva," *YUINFO 2002*, Kopaonik, 2002.