The Norwegian State Railway System GTL (1976)

Tor Olav Steine

The Norwegian State Railway System GTL (1976)

Tor Olav Steine, with the help of former colleagues

tos@alfatroll.com

Abstract. In 1976 the Norwegian State Railway System (NSB) planned a new system to keep track of all its freight cars. Among the duties were these: arranging trains at the shifting station in Alnabru outside Oslo, following (tracking) the trains as they moved along the tracks inside Norway, optimization of car maintenance and statistics. The system could save millions of Norwegian Crowns by better utilization of the car pool. It was named GTL for Gods Transport Ledelse (there is no Divine link; "Gods" simply means cargo in Norwegian). Norsk Data won the contract in competition with US-based mainframe vendors and, against all odds, delivered a system on time that "never failed" for the many years it was in operation.

Keywords: computer, minicomputer, Norsk Data, administrative system, transaction processing, data network, database.

1 The Background

We are back in time to the early childhood of computing as we know it these days. There were no personal computers - they were either shared or used by one user at the time. Users might either punch cards in separate card punching rooms and thereafter read these into the computer for processing, or share one computer through a multi-user facility called time-sharing. In the first case, the output was ready either later in the day or the day after, while time sharing allowed several users access to the computer simultaneously. On-line computing was expensive and not common. Yet, in the early 1970s minicomputers had arrived with time-sharing systems (TS systems for short) that allowed users to interact directly with software for on-line editing of code, with subsequent compiling and testing of the same. It was, however, not common for users to have access to screens with graphic displays - it was all pure text.

For non-computer people one could now make systems where the end users had direct access to central databases, and even update and modify these. But most large mainframe computers had operating systems that were optimized for batch processing (large quantities of data in, large quantities of data and printouts out).

Hence, one was forced to develop "operating systems within the operating system" – or teleprocessing monitors (TP monitors for short) to allow a large number of users to have online or simultaneous access to a large, shared database.

TP monitors differed from time-sharing systems in that they were running specially developed software systems, tailored to what the users were allowed to do, while TS systems might allow skilled users to do whatever the available software let them have access to, including developing their own solutions.

TP monitors also had another important property: Systems sharing a common database might fail and cause damage to the central database, thereby corrupting the data. Therefore, one must have a solution allowing the user to reconstruct the database into a non-corrupted state. This solution was called “Rollback/Recovery”, and it could either work for all users combined or work for only a single user and his/her session with the online computer system.

Norsk Data regarded itself mainly as a vendor to technical and scientific organizations. In 1974 the company had 140 employees, and a revenue of 38 million Norwegian Kroner (ca. 5 million USD), growing to 211 employees and 81 million Crowns in 1976.

In 1976 the Norwegian State Railway System (NSB) distributed its request for bids for a new system to keep track of all its freight cars. Norsk Data possessed a few basic technologies for use in the desired solution, but far from all. Norsk Data had never delivered any similar system. The competitors, however, were skilled in this game and could provide references in large numbers. This was the situation when the request for bids arrived to deliver the GTL system.

2 The Bidding Process

The request for bids called for an on-line system with many functions. Among the duties were: arranging trains at the shifting station in Alnabru outside Oslo, following (tracking) the trains as they moved along the tracks inside Norway, and optimization of car maintenance. The system might save millions of Norwegian Crowns by providing better utilization of the car pool and was named GTL for Gods Transport Ledelse (there is no Divine link, “Gods” simply means cargo in Norwegian).

The system required a Transaction Processing (TP) monitor capable of handling 150 terminals and heavy online traffic with a 24/7 operational capability. If not delivered on time, daily penalty fees were due. ND’s competitors were all US-based mainframe vendors, with systems like CICS in the bag already (IBM). Yet Honeywell Bull was the fiercest competitor, with its TDS TP Monitor (later TP8), IDS database, and a firm vendor relationship to Norwegian public institutions.

ND had already delivered a network (ARPANET style, named Nordnet) where all terminals were connected, and the only link between the GTL system and the terminals would be via a single pair of wires and a network protocol. Dave Walden made the basics for the network while he worked for ND during 1970-71. Dave Walden was one of the original developers of the ARPANET, and his implementation on ND machines proved to be highly reliable and efficient.

The Oslo-based research institute, Sentralinstituttet (SI - later to become SINTEF) already in 1971 made the world’s first minicomputer with full virtual memory, and in 1974 they made one of the first CODASYL database management systems for minicomputers, SIBAS.

Almost in parallel ND hired Bo Lewendal, a young Swedish developer, and he made one of the first time-sharing systems for minicomputers, Nord TSS. Starting in 1974, ND combined most of these components into its next generation of computers and operating system: Nord-10 with the Sintran III operating system. This system was capable of doing real-time, time-sharing, batch and virtual memory – all at the same time.

3 The Challenge

ND knew that a single minicomputer would be unable to deliver the required capacity. The TP system would have to split the job between four Nord-10 machines, each handling specific tasks. The machines would have to serve as individual backup for each other, and the Nordnet was used for inter-CPU communication even here. The database management component of the freight car system was the SIBAS system (just developed by SINTEF in Oslo) - a traditional CODASYL DBMS, first implemented on a Nord-1 and performing incredibly slowly in the beginning. The project was one of the largest software projects in Norway to date, with an estimated 35 man-years for the software alone. 150 TTY (Teletype) terminals in an ARPANET-style network were to have 24/7 operational access to the common database through a number of application programs - with "no errors", and a considerable transaction volume.

The main competitors were IBM and Honeywell Bull. Both had the skills, the hardware equipment, and the software solutions to deliver the solution. They were both huge organizations compared to Norsk Data, and with their mainframe computers they operated in a different league altogether.

The only asset ND had was its successful delivery of the networking system, and the fact that the other competitors also needed to interface to it. That required some detailed expertise.

The competitors had well-proven TP monitors: CICS (IBM) and TDS (Honeywell Bull) and databases: IMS (IBM) and IDS (HB).

ND had Nord-10 minicomputers with the Sintran III operating system, capable of simultaneous real-time and time-sharing with a maximum of 32 terminals per CPU. ND did not have a TP monitor and only posessed a newly developed DMBS system, SIBAS. In short: ND had no previous experience with on-line administrative projects of this order of magnitude.

4 The Solutions

When bidding for such an awesome contract, one really had to review one's inventory. The following is a reconstruction of the components that were vital for putting together a viable solution.

Bo Lewendal, as previously stated, was working alone during the summer of 1971, developing Norsk Data’s first TS system, based upon a Nord-1. Bo had fresh background from a similar project from California. It subsequently lead to Norsk Data winning a large contract with deliveries of a large number of Nord-10 machines to CERN both for administrative use and for the controlling the SPS ring (SPS: Super Proton Synchrotron), which was CERN’s largest particle accelerator at the time.

SI had also experimented with virtual memory on a Nord-1 computer already in 1971, and this became the first minicomputer with virtual memory worldwide!

In 1974 Norsk Data launched its first really smooth time-sharing system, based upon the new Nord-10 computer and the Sintran III virtual memory operating system. Sintran III allowed a moderate number of time-sharing users to have simultaneous

online access to the shared resources, such as editors, compilers and other online applications. In addition a number of real-time processes might operate in the background, invisible to the time-sharing users. These processes could, for example, control communication lines or perform process control in a production plant.

Since all users competed for the same common resources, such as CPU, memory, disk, and communication lines, inevitable delays occurred from time to time. Sintran III was, however, very efficient for its time, due to its virtual memory system and fast context switching ability.

In 1975 Norsk Data took over marketing and sales of the SIBAS Codasyl-type Database Management System (SIBAS DBMS) from SI. SIBAS allowed time-sharing users to have access to a common database, but without full rollback/recovery functions for other than the database itself. The first versions of the system were reportedly very slow.

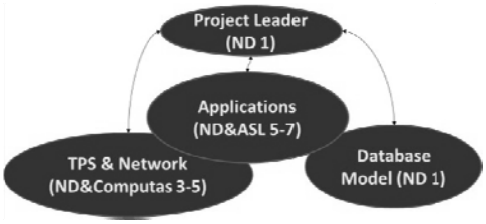
In 1970 Dave Walden, previously with Bolt, Beranek & Newman (and developer of large parts of the ARPANET), had made a communication system that allowed Nord computers to act as front ends for Oslo University's CDC mainframe computers. Norsk Data's developers converted this into an ARPANET-like network system. When delivered to NSB, it tied together approximately 150 teletype-like terminals in a large packet switched network, in operation from 1974.

Most of the necessary solutions were in place: a smart and highly efficient operating system with time-sharing and virtual memory capabilities, a database system with rollback/recovery, and an interface to the NSB ARPANET-like network.

What was missing, however, was the capability to handle a hundred or more simultaneous users. Norsk Data needed a TP monitor - Sintran III style. This TP monitor should be able to handle software and users spread across a number of CPUs, a number that could be altered if a fault occurred and the system needed to be restarted with a failed CPU taken out of the loop. This is how Nord TPS was born.

5 Organizing of the Project

The technical challenge is one side of the problem. There are also a number of people that need to be organized and put into effective work for obtaining the final solution. These were the players in the project game:



5.1 Norges Statsbaner, NSB

NSB was the customer. They wanted an efficient system with error-free 24/7 operation, readily accessible by all 150 terminals simultaneously. If the system operated successfully as planned, NSB could save millions of Norwegian Crowns.

5.2 Anderson Consulting

NSB hired a skilled project leader from Anderson Consulting as their project leader.

5.3 Norsk Data

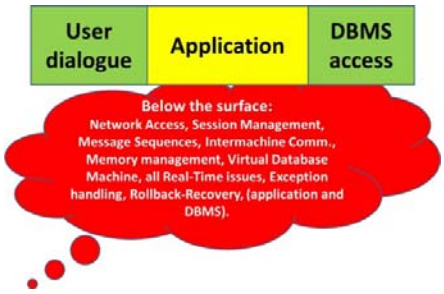
Norsk Data appointed Harald Eide as its general project leader, with Dag Spilde being responsible for the TP-monitor and networking. Lars Lind was appointed responsible for application program development, while Peter Bonne designed the database.

5.4 Computas

Norsk Data hired Computas, a subsidiary of Det norske Veritas, for the networking and TP monitor part. The author worked there at the time.

5.5 ADB System Logikk (ASL)

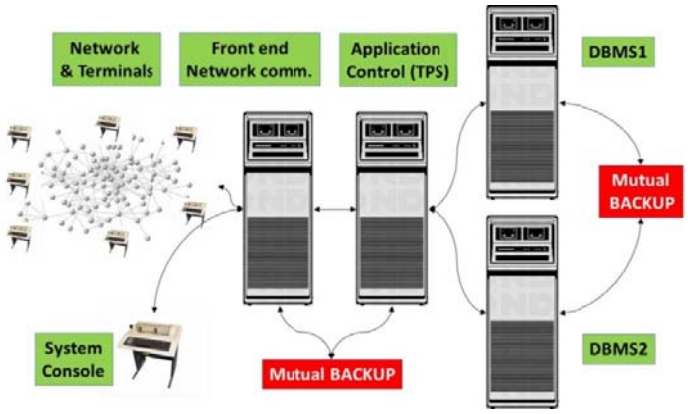
ASL was hired as developers of the application part. In order to let the application development start immediately, one decided to use time-sharing for the application development, in a simulated TPS environment. Thus, the TPS development could take place undisturbed in the background, and application development and database testing could take place in parallel.



6 System Solutions

6.1 The Hardware Structure

As previously stated, a group of 4 CPUs was selected:

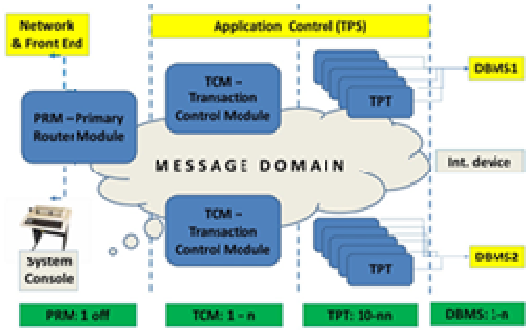


Two pairs of CPUs were mutual backup for each other. In normal operation one CPU interfaced to the NSB network, the next one contained all application programs, and the remaining two CPUs were both database machines. The database machines split the database between themselves in order to share the load, and trains would literally move between the database machines as they moved along the rail network in real life.

6.2 The Software Structure

In software the components looked slightly different. The software structure allowed a customer organization to utilize the TP monitor in the most flexible way, with as many elements as possible being automated.

The software structure allowed the users to organize all the components in one single CPU, or spread the components across any practical number of CPUs.

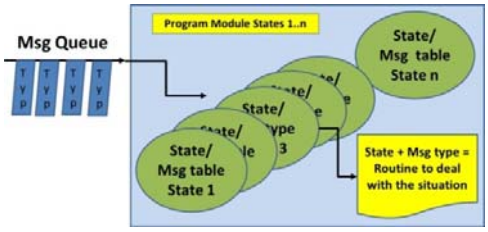


During normal operation on a full hardware configuration, the GTL system's network CPU contained the Network & Front End module, the next CPU contained the Application Control part, and the remaining two CPUs contained one SIBAS DBMS system each. For communication between the CPUs, a mini version of the previously mentioned Nordnet was used.

6.3 Program Module Structure

The main challenge was, as it always is in large project, to maintain simplicity and order. In a message-based system, this even includes the messaging structure.

For each of the modules in the system, a message/state diagram was used for maintaining the simplicity, the so-called state-vector principle:



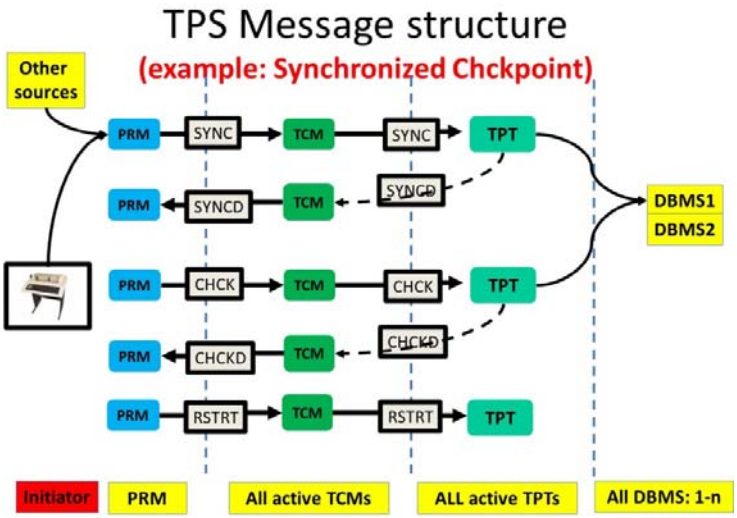
Any arriving message was handed over to a specific program element for treatment. Selection was done via a message/state vector. Modules would always be in a defined state, such as, e.g., "in normal session with a user in the network, synchronized checkpoint is going on". Upon receipt of a message, a state vector identified which piece of code to invoke at the receipt of this particular message type while in the particular state. The resulting code was extremely compact and easy to maintain.

All aspects of the software solutions were subject to thorough code inspections, where the author had to describe his/her solution to the scrutiny of critical fellow project participants.

In hindsight, the combination of state vectors and code inspections were the vital reason for the success of this project.

6.4 Message Structure

The effect of the state-vector principle was that most of the complexity resided in the message system, while the software itself was very simple and "if-then-else-less". Here is an example of the message sequences for "Make a synchronized checkpoint"



In order to leave control with the user developers, TPS initiated so-called "Special Applications" at distinct points during the operation. Such Special Applications were either Global, i.e. common for all users, or Local, i.e. meant for a specific user session only.

The user organization decided, what should happen when these defined incidents occurred, by modifying the content of the standard Special Applications.

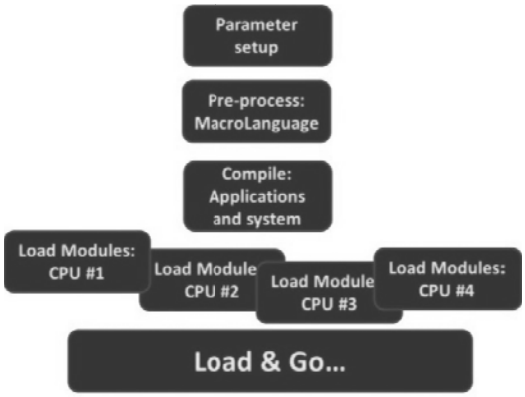
6.5 The System Builder

TPS and its components were not trivial, yet it managed to hide a large amount of complexity for the users, the application programmers, and the operators of the system, once it went into operation.

The final element in this strategy was to make a system builder, which reduced complexity for the personnel that already had many problems to solve and had no need for new problems to deal with.

The developer called the system builder "The Queen of the Night", since heavy batch processing capacity was required once it started.

Actually, there was a complete version (as seen below), and a short-cut version, which skipped some of the more time-consuming parts, when they were not needed.



The output of the system builder was load images for each of the CPUs at hand, and the total system was initiated by loading the CPUs and typing the “initialize” command on the system console.

7 Epilogue

The NSB GTL system was a success. It was delivered as agreed to the customer site close to the rail lines in Oslo, although the delivery time was delayed 2-3 months due to new functionality wanted by the customer. Actual operation started a few days over the agreed time, and had very few operational 'hickups' during its entire lifetime. Some strange hardware errors occurred, though. The memory of one of the computers broke down on several occasions. Later it was explained by the passing of nearby electric trains, during which they disturbed the power lines, thereby causing breakdowns in the sensitive MOS memory circuits.

Summing it all up: ND possessed a few ingredients that might be used in the total solution, yet a formidable job remained for a complete solution to be put together, let alone to be delivered on time. Still, the system was delivered according to the agreed on time, it functioned as planned, and served with an impressive stability and performance during its entire lifetime. The required up-time for the system was 99,75%, yet 99,98% was achieved during its lifetime.

The system was delivered as planned, on time, and without altering the original number of project participants.

In ND’s history, this was one of the more successful projects. TPS was used only a handful of times in addition to the NSB GTL system. Once it was used in a vital project for the Swedish Defense LEO system (6 CPUs in the entire configuration), and once for the Swedish Foreign Debt Organization, “Riksgälden”.

The project opened the road for ND to be a serious vendor even in the administrative market. It also illustrates how ND could become such a tremendous success in a short time.

We all know how it ended, but that is another story.

Acknowledgements

Many thanks to Dave Walden, Bo Lewendal, Dag Spilde, Lars Lind, Peter Bonne, Harald Eide, Per Ivar Danielsen, and Stig Nyberg, for their valuable contributions and inputs.