periodica polytechnica

Transportation Engineering
36/1-2 (2008) 3–8
doi: 10.3311/pp.tr.2008-1-2.01
web: http://www.pp.bme.hu/tr

RESEARCH ARTICLE

Server Architecture Development for On-line Tracking of Large-sized Vehicle Fleet

Szilárd Aradi

Received 2007-03-03

Abstract

This article describes the structure of a fleet management system and its system elements. First, the schematic structure (central server and on-board computers) are outlined. Therefore the details of the communication, operation and the emerged problems are given. Later the development and testing of the central server, its software, and the database server are described. Finally the advantage of the system and the development possibilities are summarized.

Keywords

computer · communication · fleet management system

Szilárd Aradi

Department of Control and Transport Automation, BME, H-1111 Budapest Bertalan L. u. 2., Hungary

e-mail: szilard.aradi@auto.bme.hu

1 Introduction

Thanks to the rapid development of microelectronics and mobile telecommunications by the end of the 90's a wider range of fleet diagnostics and satellite tracking became possible. These innovations gave a technological background of the creation of fleet management systems. Economical demand for fleet management systems strengthened as an effect of the increased competitive situation in passenger and freight transport especially in road traffic. This effect was strengthened by the increase of traffic density and the intensification of transportation demand in Europe. One of the social impacts of this increase is an increase in the number of accidents that intensifies the demand for safer vehicles.

Spread in on-line fleet management systems was greatly aided by the constant decrease of communication charges and the increase of speed in data transmission.

For all these reasons in the recent years online fleet management systems in traffic spread rapidly.

Its advantages are the followings:

- a greater safety in delivery,
- aiding dynamic freight arrangement,
- constant tracking of the mechanical condition of the vehicles,
- easier documentation,
- aiding efficiency wages,
- developing safety of traffic,
- developing safety of freight,
- increased environmental protection.

This article presents the structure and elements of an online fleet management system, with a special regard on the invention of a server system of a higher reliability.

2 System structure

General construction of online fleet management systems are demonstrated on Fig. 1. The three main features are:

- on-board computer,

- central server,
- user computers.

The operation of the system is the following. The board instruments on the vehicle measure the operational parameters of the vehicle (state of the switches, energy consumption, motor parameters, etc.), and its position (aided by GPS based location), and they store the data given by the vehicle (the name of the actual activity, etc.). These parameters are sent to a central server at the actualization of previously defined events (alarm-signal, sudden decrease in fuel level, etc.) and in previously defined periods of time.

On-board computers communicate with the central server through mobile systems. The incoming data are evaluated and stored in a database. If necessary the central server can send an alarm to a given e-mail address or even a mobile phone. In this structure communication from the server towards the vehicle is plausible as well. Aided by this the incoming data packages can be confirmed, a written message can be sent to the driver and the parameters of the board unit can be set.

Vehicles are detectable and observable almost constantly (online) and the operating parameters (running performance of vehicles, energy consumption, activities and work time of drivers, delivery performance) can be followed by a later evaluation of data stored in the centre (offline).

3 On-board unit (OBU)

In the building of the on-board unit important aspects are a heavy-duty design (EMC protection, shake protection, fluctuation of environmental temperature, etc.) and modularity. Therefore one should use a system that is built up of individual units. The connection of these by a series communication connection is worth realizing for the sake of simplicity and easy expansion. For this the most appropriate is the Controller Area Network (CAN) bus system.

Board unit is made up of the following main units:

- GSM/GPS module,
- central unit,
- incoming unit,
- human interface device,
- diagnostic adapter,
- CAN bus,
- power supply unit and background batteries.

4 Communication

The communication system can be built up by OSI model as shown on Table 1.

The connection point between the OBU-s and the server is the session layer (UDP or TCP socket).

The first step is the determination of the session layer's protocol. There are a few key features that set TCP apart from User Datagram Protocol:

- Ordered data transfer,
- Retransmission of lost packets,
- Discarding duplicate packets,
- Error-free data transfer,
- Congestion/Flow control.

In the data block of TCP packet there must be built up a record structure (Table 2), which contains the vehicle data.

The record can contain BCD (Binary Coded Decimal) characters for numbers, and ASCII (American Standard Code for Information Interchange) characters for letters.

5 Central server

5.1 Tasks

The main tasks of the central server are:

- data receiving from the vehicles,
- piggybacking to the vehicles,
- identification of drivers,
- inserting data to the SQL database,
- sending alert, if necessary,
- setting the parameters of OBU-s,
- remote diagnostic handling
- software update.

5.2 Challenges

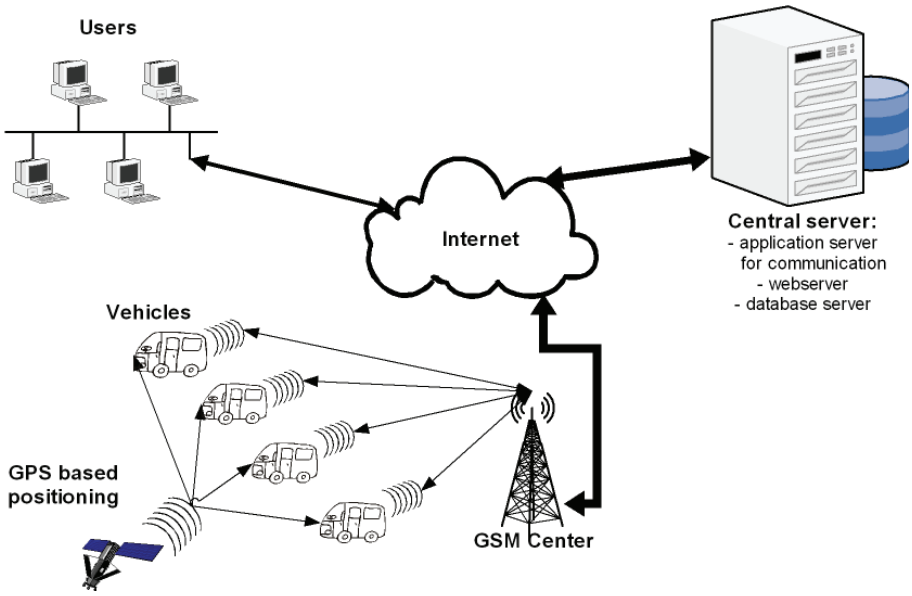
A computer of decent performance should be enough for dealing with the tasks of communication, because of the simple kind of raw data arriving from the vehicles, not larger than about 100 bytes/client. This amount of data can not cause any problem even in case of thousands of clients. On the other hand it is necessary to handle the clients carefully to avoid data loss. The bottleneck of such application is the database server, because of the enormous size of the tables and the amount of queries indicates the use of a powerful hardware.

For example let us consider 1000 vehicles and a 30 seconds data sending period:

$$1000 \times \frac{60}{0.5} \times 24 = 2880000 \frac{\text{messages}}{\text{day}}$$

By calculating with messages of 100 bytes, the overall storage size is only:
$\frac{2880000 \times 100}{1024^2} \cong 275 \frac{\text{Mb}}{\text{day}}$, but
$2880000 \frac{\text{rows}}{\text{day}}$ are inserted into SQL table.

Fig. 1. System structure



Tab. 1. Architecture of communication system

OSI model	Used protocol or service	Implementation in OBU	Implementation in central server
Physical layer	GSM		100BASE-TX
Data link layer	GPRS	GSM modem	Ethernet
Network layer	Internet Protocol (IP)		Operating system
Transport layer	Transmission Control Protocol (TCP) or Universal Datagram Protocol (UDP)	TCP stack in GSM modem or microcontroller's software	TCP or UDP server class (Server software)
Session layer	TCP socket or UDP socket		Client thread (Server software)
Presentation layer	Data exchange with pre-defined records	Microcontroller's software	Client thread and SQL thread (Server software)
Application layer	SQL server	–	SQL server (Oracle)

Tab. 2. Structure of the record

Start	Data type ID, Event ID	Ordinal number of record, Length	Vehicle ID, Driver ID	Date and Time
GPS data	Digital signals (switches, CAN data)	Analogue signals (fuel quantity, engine speed, temperature etc.)	Activity data	End

The number of the rows depends on the structure of the data record (see Section 4).

The example shows, that the size of the table grows dynamically. In an ordinary database, one *INSERT* statement runs for each new data-row. The user applications execute a lot of queries on the received data, as a result of the on-line vehicle following. These properties raise more problems:

- a lot of *INSERT* statement cause a large load
- the queries are slow,
- because of the load caused by queries, a timeout error may occur by the *INSERT* statement.

The identification of the drivers may cause further load.

5.3 Development of the server application

Considering the above mentioned problems, a Win32 server application was developed with the Borland Developer Studio 2006, Turbo Delphi for .NET rapid application developer.

The database server is an Oracle 10g SQL server, which runs on an older computer with an Intel Pentium 4 1.6 GHz processor, 768 Mb RAM, enough HDD space, and Windows XP Professional SP2.

The communication runs on a self-made protocol based on TCP. The data record contains only BCD and ASCII bytes, except the start and stop bytes. It makes the syntactic and semantic checks easier, and reduces the cost of the data transmission. The record also contains an internal 16 bits checksum.

The application works with more simultaneous threads. The threads used in the software are the following:

- Central server thread for listening socket (*ServerThread*)
- Client threads for each clients (*ClientThread*)
- Database threads (*SQLThread*)

The base functionality of the application is the following. At the starting of the program the *ServerThread* (*TThread* class) creates the *Server* object as an instance of the *TTcpServer* class. The *ServerThread* is listening continuously on a specified port by the *Server* object. If an incoming connection occurs, the *ServerThread* creates a *ClientThread* entity (*TThread* class), which takes over the handling of the connection. The *ClientThread* communicates by the *Client* object which is an instance of the *TTcpClient* class. The *ClientThread* is the most complex part of the software. It has to deal with the following tasks:

- data receiving
- data checks (syntactic, semantic, checksum)
- data conversion and passing to the *SQLThread*
- piggybacking to the clients
- identification of the drivers
- sending the parameters of the OBU

The *SQLThread* is also an instance of the *TThread* class. The .NET assembly of the Oracle Client 10g is used for connecting to the Oracle database server and running PL/SQL commands. The tasks of the *SQLThread* are keeping the database connection, inserting the data into a table (*OBU_Table*), and querying the driver's ID and the OBU's parameters. In section 5 it was appointed, that the huge number of *INSERT* statements may overload the database server. To avoid it, a stored procedure, named *OBU_InsertProc* was created by the following scheme:

CREATE OR REPLACE PROCEDURE "TEST"."OBU_INSERTPROC" (PNumber1 NUMBER, . . ., PNumberN NUMBER, PString1 VARCHAR2, . . ., PStringN VARCHAR2)

BEGIN

INSERT INTO "TEST"."OBU_TABLE" (FNumber1, . . .,FNumberN, FString1, . . . ,FStringN) VALUES (PNumber1, . . ., PNumberN, PString1, . . ., PStringN);

END

Where the *FNumberN* shows the *NUMBER(x,y)* type fields and the *FStringN* shows the *VARCAHR2(x)* type fields in the *OBU_Table*. With this procedure the data inserting is more effective, because the Oracle Client's OracleCommand class supports the use of so-called "bind variables" meaning, that the stored procedure can accept also arrays as input parameters, if the required properties (CommandType, ArrayBindCount etc.) of the class are properly configured. With this method, it is avoidable to insert the records one by one, but several records can be collected to one array and handed it over to the stored procedure. The filling up of the *OBU_Table* is more effective and faster with this solution.

The other problem is the slowness of the queries because of the huge size of the OBU_Table. The solution is that the *OBU_Table* contains only the data of one day (24 hour). The other part of the data can be sorted by a vehicle property or date, and it can be stored in other tables or databases. If the GPS data (necessary for on-line vehicle following) is continuously copied to another table, the effectiveness will increase. An SQL trigger is the most suitable for this task. It must run after insert on each new row. The following trigger was created:

CREATE OR REPLACE TRIGGER GPS2TOPO_TABLE

AFTER INSERT ON "TEST"."OBU_TABLE" FOR EACH ROW

DECLARE

BEGIN

END

It refreshes the GPS data for each vehicle in a table called TOPO_Table. TOPO_Table contains only one row for each vehicle and field of driver ID is indexed. So the queries of topographic applications run faster and does not load the OBU_Table.

Fig. 2. System structure

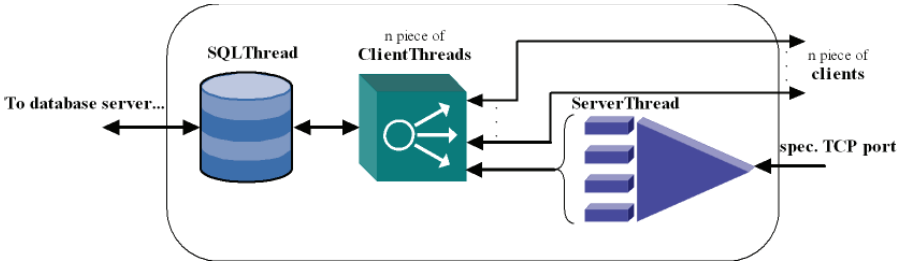
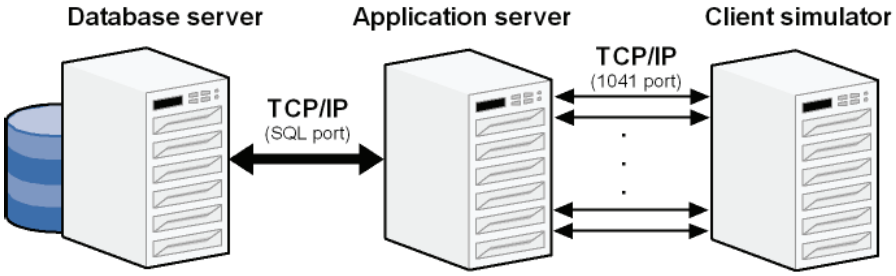


Fig. 3. Test system



6 Testing

The server application was tested by a simulator, which is developed specially for this work. This application can simulate a predefined number of clients and it can communicate with the protocol used by the server. The period of data sending, the size of the data record and the distribution of the load can be configured. The simulator saves all of the sent data for each simulated clients to files, so the test results can be evaluated by comparing the sent data with the OBU_Table.

The test system was built as seen on Fig. 3. The server application and the test software run on two separate computers with the same hardware configuration: Intel Pentium M 1.8 GHz processor and 512 Mb RAM, Windows XP Professional SP2. The specification of the database server can be read in the Section 5.3

In this phase of the development two case studies were completed. The first test was accomplished with the following parameters:

- 200 clients,
- 100 byte data record
- 20 field OBU_Table
- 5 s data sending period for each clients,
- normal data records,
- 24 hours testing time.

During the test:
$\frac{1}{5} \times 60^2 \times 24 \times 200 = 3456000$ rows were inserted.

During the evaluation of the test results, it has been proven, that the 100% of data got into the OBU_Table.

The second one was a worst-case test. The assumption of this test is that the largest load occurs after a long time server shutdown (for example by a hardware error or maintenance). In this case all of the clients are sending their data stored in the memory of the OBU continuously, so that the normal functionality of the system can be restored as soon as possible. The parameters of the test were the following:

- 200 clients,
- 100 bytes data record
- 20 field OBU_Table
- 5 s data sending period at normal mode,
- continuous data sending after server start
- normal data records
- 2 hours server shutdown

The test executed as long as all of the data (theoretically collected during 2 hours) were got into the database. During the test all of the clients were needed to send
$\frac{1}{5} \times 60^2 = 720$ messages, so
$720 \times 200 = 144000$ rows were inserted into the OBU_Table.

The data received the database in 9 minutes without any data loss.

Also the performance test log file of the Oracle server was appraised at the end. It showed that the triggers and stored procedures worked normally, only the CPU and memory usage was at critical level, and the usage of pagefile became increasingly frequent. However it was prospective because of the database server's hardware configuration.

One must take into account that the client bandwidth (100 Mbit / 200 clients) was much larger, than a GPRS connection's maximal bandwidth, which indicated a much larger load in the second test case, than in practice.

Finally these tests indicated that the basic principles of such application are suitable. In accordance with the server application's average CPU and memory usage it can be stated that there is a lot of reserve in the software.

7 Summary

The article presented the theoretical structure of an on-line fleet management system containing a huge number of vehicles. A general purposed server system was developed, able to satisfy the claims of any on-line fleet management system. The system was designed for a fleet containing 1000 vehicles. Also an OBU simulator software was developed for testing the server. The results of the test are promising. The tests also show that the designing, managing, and maintaining of the database server needs a lot of consideration.

But on the other hand, more tests will be necessary in the future. The more real-like simulation and emulation, and the increase of the number of clients are necessary. A well-configured and more powerful hardware based database server will have to be built. The simulation of the queries of the user applications is also important.

References

1 *Scientific Association for Infocommunications: Telecommunication Networks and Informatics Services*, available at `http://www.hte.hu`. On-line book.

2 **Aradi Sz**, *Telemonitoring System with Locomotive On-Board Computer*, Hungarian Rail Technology Journal (2007), no. 1, 27-28.

3 *Oracle Database Express Edition 10g Release 2 (10.2) Documentation Library*, available at `http://www.oracle.com/pls/xe102/`.

4 **Tóth B, Tamás P**, *Programozzunk Turbo Delphi rendszerben!*, Computerbooks, 2007.

5 *Microsoft .NET Framework Documentation, .NET Framework Developer Center*, available at `http://msdn2.microsoft.com/en-us/library/aa139615.aspx`.