111175 N95- 17590

31995

A New Communication Protocol Family for a Distributed Spacecraft Control System

p. 9

Andrea Baldi, ESA/ESOC/FCSD

Marco Pace, Vitrociset Space Division

Abstract

In this paper we describe the concepts behind and architecture of a communication protocol family, which was designed to fulfil the communication requirements of ESOC's new distributed spacecraft control system SCOS II.

A distributed spacecraft control system needs a data delivery subsystem to be used for telemetry (TLM) distribution, telecommand (TLC) dispatch and inter-application communication, characterised by the following properties: *reliability,* so that any operational workstation is guaranteed to receive the data it needs to accomplish its role; *efficiency,* so that the telemetry distribution, even for missions with high telemetry rates, does not cause a degradation of the overall control system performance; *scalability,* so that the network is not the bottleneck both in terms of bandwidth and reconfiguration; *flexibility,* so that it can be efficiently used in many different situations.

The new protocol family which satisfies the above requirements is built on top of widely used communication protocols (UDP and TCP), provides reliable point-to-point and broadcast communication (UDP+) and is implemented in C++.

Reliability is achieved using a retransmission mechanism based on a sequence numbering scheme. Such a scheme allows to have cost-effective performances compared to the traditional protocols, because retransmission is only triggered by applications which explicitly need reliability. This flexibility enables applications with different profiles to take advantage of the available protocols, so that the best rate between speed and reliability can be achieved case by case.

Introduction and Context

SCOS II is a generic mission control system, providing a collection of buildings blocks upon which a custom control system can be implemented with moderate effort (ref. [1]). Basic services are provided by the Distributed Access Service layer (DAS) responsible for distribution, local caching, and retrieval of mission information (e.g. TLM and TLC) over the network. An Application layer (APP) provides basic building blocks for implementing mission applications.

A SCOS II system is distributed and is composed by several Unix workstations connected on a local area network. Each workstation or node has a role with associated communication requirements determined by the mission configuration. The role of a node and consequently its communication requirements are determined by the applications running on it. The following classification is useful to understand the different roles a node might play:

- *server*: a node that provides services, usually data to be consumed by clients.

Andrea Baldi (abaldi@esoc.bitnet) works within the Flight Control Systems Department at the European Space Operations Centre (ESOC), Robert Bosch Strasse 5, D-64293 Darmstadt, Germany. Marco Pace (mpace@esoc.bitnet) works for Vitrociset Space Division, Via Salaria 1027, I-00138 Rome, Italy. The work described in this article was carried out at ESOC under a contract with the European Space Agency.

- *replica*: a node that provides services, like a server, but is only available when the primary server is down.
- *client*: a node that makes use of the services provided by servers or replicas, usually data to be consumed.

Servers, replicas and clients can be classified as *Reliable*; in case of a server or replica *Reliable* means that the node *supports* reliable delivery of data; in case of a client it means that it *requires* reliable delivery of data.

A workstation may play more than one role at the same time (e.g. server and client, replica and client), therefore the communication requirements may change over time. Communication and information distribution is achieved using the services provided by the Inter Process Communication layer (IPC) which is part of the DAS.

The IPC services are used by the DAS when communication is required, but also by the APP layer directly as shown in Figure-1.

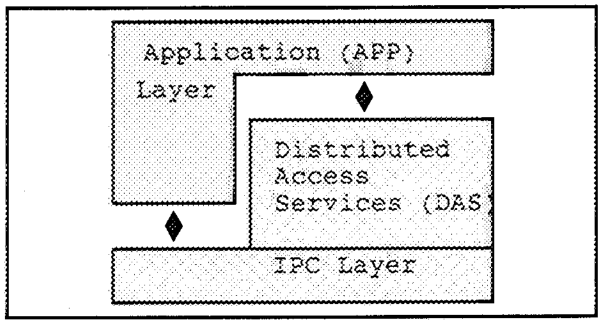

Figure-1 SCOSII Software Layering

The IPC layer has an important role because it supports the bulk of the information exchange among the different system components.

A typical SCOS II configuration (See Figure-2) will be composed by a number of servers, clients and replica nodes. The number of nodes may change dynamically according to the mission phase and configuration, to contingency conditions, and to the number of interactive users connected to the system.

TLM data is received at a central node and distributed to all the nodes by means of the IPC. TLC are dispatched using the IPC to a central node for uplink.

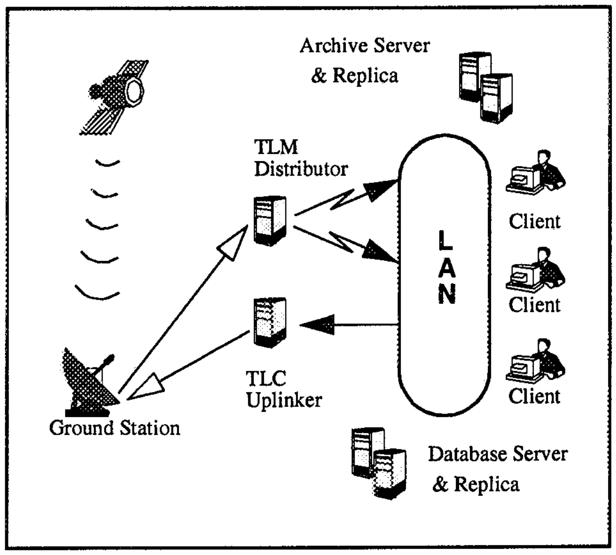

Figure-2 A Typical SCOS II Configuration

In a such context, where applications have different communication requirements, classical protocols like UDP and TCP are not able to cope efficiently with all the possible situations. The IPC tries to fill in the gap existing between UDP, a fast but not reliable protocol, and TCP, a reliable but not efficient protocol, defining the UDP+ protocol.

The protocol family available to SCOS II users extends the IP family and provides:

1. a reliable broadcast service (UDP+), with performance not too far from UDP.
2. an integrated environment where applications with different communication requirements can coexist without imposing overhead to each other.
3. the possibility to select the protocol that best fits the application's communication requirements.
4. compatibility with the already existing IP protocols.
5. support for fast local communication (FIFO).

Requirements

As introduced before, the IPC layer has to cope with many different situations and it is clear that no unique protocol can be designed to fulfil all the application requirements simultaneously. A protocol family in fact best satisfies multiple and sometimes conflicting needs. The following considerations describe the trade-offs made in order to satisfy as many requirements as possible.

Communication Schemes

Allowing different communication schemes to coexist in an integrated environment is fundamental for achieving flexibility so that applications can use different approaches to data distribution such as *point-to-point* and *broadcast*.

The IPC layer supports all these schemes providing a single unified abstraction called *Channel* available for any supported protocol. A *Channel* can be seen as an endpoint for communication which an application can use to send or receive data.

Protocol Scalability

Scalability is another key requirement and the IPC layer fully scales with respect to the communication data volume by means of the broadcast communication schema. Moreover it tries to avoid situations where the unreliability of the used IP services, which triggers packet retransmission requests, might cause a network congestion.

The retransmission algorithm already tries to optimize the policy of lost packets retransmission using the most appropriate communication scheme; broadcast is used for instance in the case it is detected that a lost packet is requested by several applications. The algorithm is tunable and it is driven by application *Hints* and information piggy-backed into retransmission requests.

Hints are used to instruct a server about the application reliability requirements. They can be used to avoid or force retransmission of data case by case as well as determine the number of attempts the IPC layer carries out before giving up the retransmission.

Protocol Reliability and Speed

The reliability of the protocol together with the speed necessary to cope with a high TLM delivery rate is a primary issue for SCOS II applications. Reliability and speed are tightly related and an effort to meet both the requirements is made.

Within the IP family, TCP is a fully reliable protocol where the speed is inversely proportional to the network load, while UDP is a fast one with a reliability inversely proportional to the network load.

UDP+ stays in between, is highly tunable and tries to fill in the gap existing between TCP and UDP (See Figure-3).

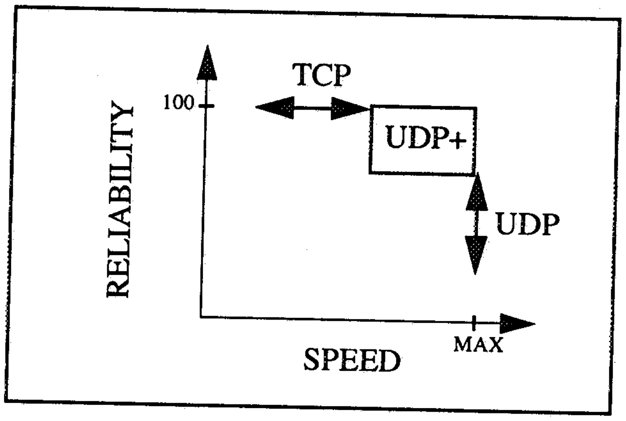

Figure-3 Speed vs Reliability Diagram

Transparent Reliable Data Delivery

Giving the user the responsibility of retransmitting or receiving data lost due to a network problem is not acceptable for client applications. The recovery of lost data is managed automatically by the IPC, without forcing the application to use any recovery policy.

Asynchronous Communication

When applications are data driven an asynchronous communication mechanism is very useful.

The IPC layer provides the concept of a *Notify Channel*: a channel marked as *Notify* does not require the attention of the application; the application only needs to define the handler to be called on data arrivals.

The IPC automatically gives control to the

To keep this risk at a minimum a *piggy-backed* acknowledgement of the already received packets is used in the retransmission requests. It is important to note that for broadcast communication the *piggy-backed* information provides just an indication and there is no guarantee that a packet will not be requested in the future.

Under normal operation, when a packet is received the client protocol verifies that its sequence number is correct and then it delivers it to the application. The following anomalous situations are recognised and handled by the client side of the protocol:

- *loss of packets*: when a gap in the sequence is detected the retransmission of the missing packets is requested. In the meantime out of order packets can be received; they are discarded or stored in the client buffer according to their sequence number.
- *duplication of packets*: duplicated packet are always discarded.
- *delay in packet delivery*: if a delayed packet is received before the retransmitted one, it is returned immediately to the application, otherwise it is discarded.

Protocol Family Design

This section provides the description of the architecture of the IPC layer. The architecture described is a simplified one showing the main classes relevant to the problem domain. Some implementation classes have been omitted for the sake of simplicity and clarity.

The description uses a Rumbaugh Object Diagram (ref. [6]) where classes have been grouped into 3 *subjects (Data Handling, Transport Mechanism* and *Statistics*) according to the responsibilities they fulfil in the problem domain, as shown in Figure-5.

Data Handling Subject

Data handling groups together the classes dealing with the SCOS II transfer unit (STU). They implement respectively the *header* and the *data part* of an STU, the fragmentation and reassembly of STUs and the storage of STUs for the client and server side of the reliable protocol.

The *StuHeader* class specifies the information needed by the IPC layer to perform the transport of the packets on the network, the sequence number used by the UDP+ protocol,

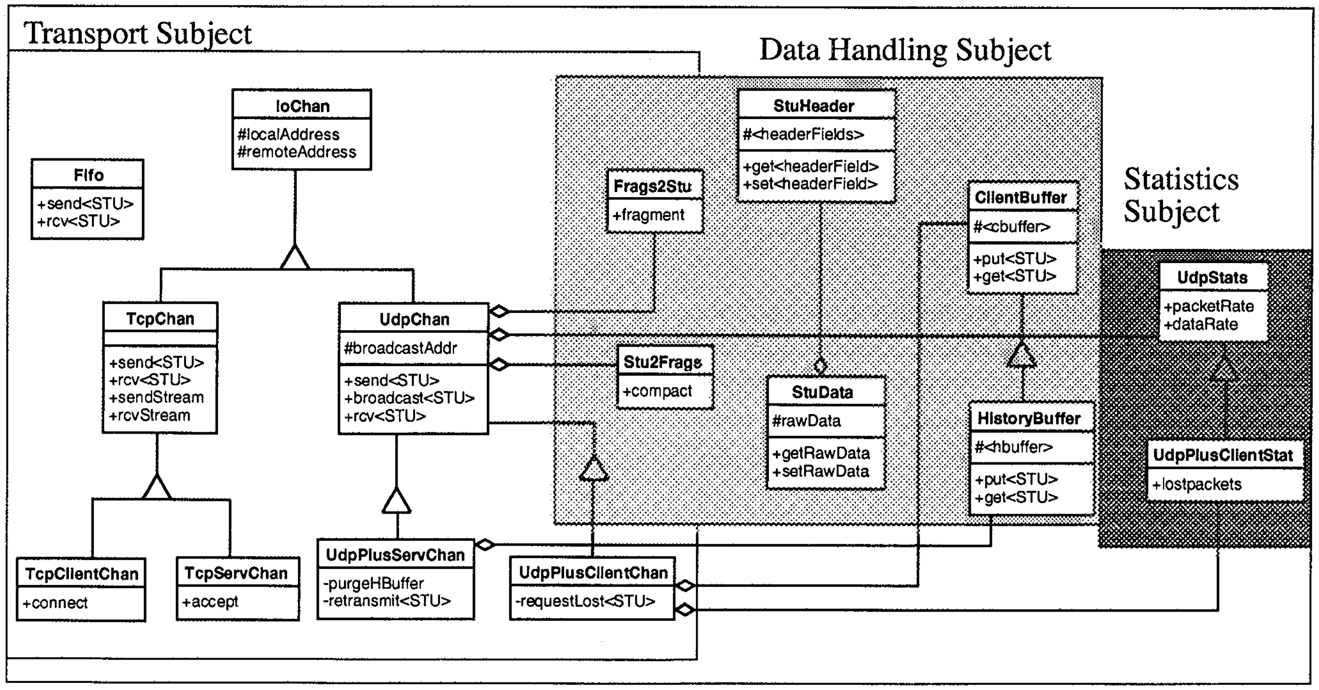

Figure-5 IPC Object Oriented Model

the fragmentation information and client hints. Services for getting and setting the value of the header fields are used by the IPC internally, but can also be used by applications to manipulate fields containing application-defined values. Those fields allow the STU to be tagged as belonging to a specified category.

The *StuData* class contains the application data, of whatever type. Such data are appended to the header and sent over the network when a send operation is performed.

The fragmentation layer (*Frags2Stu* and *Stu2Frags*) takes care of the splitting of large packets into smaller ones and of their reassembly at the destination. In this way the IPC layer is able to support the transmission of packets of virtually any size.

The *HistoryBuffer* and *ClientBuffer* classes implement data structures used by the UDP+ transport classes for the storage and retrieval of STUs. In particular, *ClientBuffer* is used to temporary store packets which cannot be delivered to the application because the retransmission of a lost packet is in progress; the *HistoryBuffer* is used to store transmitted packets which might be requested when a packet is lost.

Transport Mechanism Subject

Transport mechanism groups together the classes which deal with data delivery for the IPC layer protocol family. With the exception of the *Fifo* class introduced for fast local communication all the classes support remote communication.

The *Channel* class defines attributes and services which are common to all the supported communication schemes in the IP domain. *Channel* specializes into two branches respectively responsible for a TCP-based and an UDP-based communication. New IP based protocols can be derived from the *Channel* class specialising it at the most suitable level of the hierarchy.

Inheriting from the *TcpChan* class, two specific classes are defined to model the client and server side of a connection based communication. Both of them provide services for sending and receiving STUs and byte streams: the *TcpClient* adds connection establishment capability and the *TcpServer* adds connection acceptance capability.

Also inheriting from *Channel*, the class *UdpChan* provides services for receiving and sending STUs, both with point-to-point and broadcast connectionless transmission.

Classes *UdpPlusServChan* and *UdpPlusClientChan*, reliable components of the IPC layer, are derived from the *UdpChan* class. In principle, they could have been grouped together in a single *UdpPlusChan* class because they provide the same interface as *UdpChan*. The reasons for such separation are:

- typically applications behave either as clients or as servers, not as both.
- the resulting implementation is less complex and easier to maintain.
- the resulting application overhead is reduced because applications will only include the minimal amount of data structures instantiated by the relevant classes, being such data structures different in the two cases.

Class *Fifo*, at last, supports fast local communication of STUs through a UNIX FIFO, maintaining a similar interface to the one provided by the remote communication classes.

Statistics Subject

Statistics groups the classes responsible for gathering information on volume of data sent and received on any UDP based channel. They have been introduced to tune and debug the internals of the IPC layer, since they provide a complete snapshot of the behaviour of the protocol including all the information on lost and retransmitted STUs.

Moreover, the Statistics classes have been used to implement a performance monitoring tool which reports on data and packet rate. Statistics are also available to an application for any possible usage it envisages.

Building Applications

The IPC layer offers application writers a flexible solution for the exchange of data. Applications can in fact exchange STUs using the protocol that best fits their communication requirements. Moreover applications using one of the UDP based protocols can select the reliability level as they like. It is important to notice that applications can be configured to use any of the available UDP based protocols, and still be able to communicate with each other.

Data transmitted by an application using a reliable server channel (*UdpPlusServChan*) can be received by an application using a non reliable channel (*UdpChan)* with the only difference that lost packets will not be detected and consequently not requested.

Reliable clients (*UdpPlusClientChan*) can also receive data sent over a non reliable channel, but in this case the protocol does not perform any check on the sequence number, and just delivers the data it receives to the application.

The following list shows some of the SCOS II applications or system components together with their communication requirements (see also Figure-2):

- *TLM Receiver and Broadcaster:* it receives the telemetry from the ground station and broadcasts it to the system. It is a reliable server and satisfies retransmission requests coming from reliable clients.
- *History File Archiver (HFA)*: it archives the received telemetry and retrieves it on application demand. As a consequence that all the telemetry coming from the ground station need to be archived, the HFA is a reliable client. At the same time, it satisfies retrieval requests on the network, so it is a reliable server.
- *TLM Cache*: it receives real time telemetry and makes it locally available to the applications. It can be configured either as a reliable or a non reliable client according to the role the node has in the system.

Conclusion

The use of the IPC layer for more than one year in the SCOSII system has shown that the initial objectives have been achieved:

- the retransmission approach together with the almost full reliability of the network hardware make the degree of reliability high enough to guarantee that any node receives the data it needs to accomplish its role.
- UDP+ efficiency compares favourably with UDP and definitely well with TCP. The overhead introduced by the retransmission mechanism is a fraction of the benefits obtained, especially when considering reliable broadcast. The results collected using the IPC statistics are summarized in Figure-6 where the data rates achieved are shown. Such figures may vary, however, depending on the dynamic tuning the applications perform on the IPC using hints.

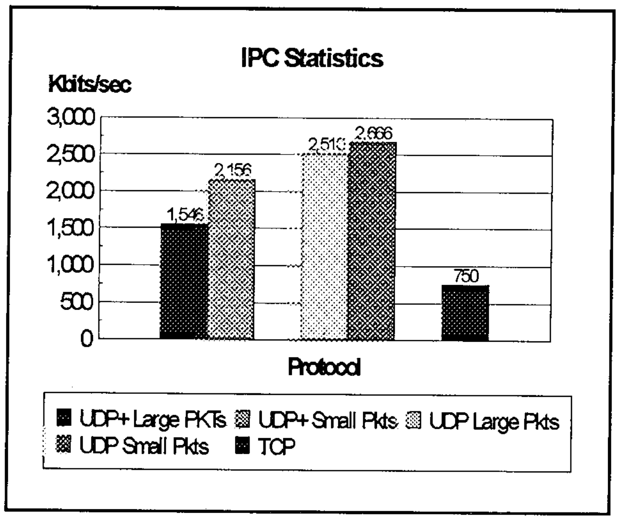

Figure-6 Protocols Statistics for a Typical Mission Configuration Consisting of 20 Nodes.

- the protocol scalability guarantees that adding new client workstations does not require any reconfiguration and does not impose unacceptable network overhead.
- the protocol family has shown to be flexible enough to satisfy different communication requirements for a wide range of applications that need to exchange the same data

using different protocols. This is achieved by the IPC layer through the introduction of a common exchange data unit (the STU) together with a continuous range of performances both for point-to-point and broadcast communication. It is important to note that SCOS II applications can communicate even with already existing software not supporting a STU based data exchange.

The IPC layer is now complete and stable in the interfaces, although its implementation evolves as a result of a continuous life cycle which includes analysis of statistics, tuning and test.

New generations of Unix already support multiprocessor hardware and the time to make the IPC layer fully reentrant is mature as multi-threaded SCOS II applications are under development. To have a full coverage of commonly used protocols the IPC layer will be augmented in the future to support non IP based protocols, like Unix datagram, streams and X25.

References

[1] K.Keyte: SCOS II - *A distributed Architecture for Ground System Control* - International Symposium on Spacecraft Ground Control and Flight Dynamics SCD1. - Feb. 94 - Sao Jose dos Campos - Brazil.

[2] M.Frans Kaashoek et al. - *An Efficient Reliable Broadcast Protocol* - Operating System Review- Volune 23 Num. 4, October 1989.

[3] T.A. Joseph and K.B. Birman - *Reliable broadcast protocols* - Lecture Notes of Artic 88, Tromso, Norway, July 1988.

[4] Douglas E.Comer - *Internetworking with TCP/IP* - Vol.I - Prentice Hall.

[5] W.Richard Stevens - *UNIX Network Programming*- Prentice Hall - 1990.

[6] Rumbaugh et al. - *Object Oriented Modelling and Design* - Prentice Hall - 1991.