Operating System of the Multi-machine Computer AS-6

I. Bourdonov, V. Ivannikov, A. Kossatchev, S. Kuznetsov, A. Tomilin

Operating System of the Multi-machine Computer AS-6

I.B. Bourdonov[1], V.P. Ivannikov[2], A.S. Kossatchev[3],
S.D. Kuznetsov[4], and A.N. Tomilin[5]

[1] Institute for System Programming of RAS, igor@ispras.ru
[2] Institute for System Programming of RAS, ivan@ispras.ru
[3] Institute for System Programming of RAS, kos@ispras.ru
[4] Institute for System Programming of RAS, kuzloc@ispras.ru
[5] Institute for System Programming of RAS, tom11@bk.ru

Abstract. The S.A. Lebedev Institute developed operating system for the AS-6 distributed computing system (OS AS-6) in the 1970s. The OS AS-6 consisted of peer operating systems of separate machines making up the AS-6 computer system. Those operating systems interacted through a uniform interface. The OS AS-6 facilitated interaction between computation processes on the nodes of the AS-6 system through a network together with their interaction with the global network. It provided the possibility for any process in the AS-6 system to use all the devices connected with the nodes of the AS-6 computer both with addressing and with usage of external devices. It also supported a pipeline operation of the computers in the AS-6 system performing real-time processing of large data streams of spacecraft missions. The operating systems of the nodes of the AS-6 system had a special means for parallel processes management, including task hierarchy organization and information processing management within a single task.

Keywords: Distributed computing system, operating system

1 Overview

In the 1970s, the S.A. Lebedev Institute of Fine Mechanics and Computer Engineering developed the operating system (OS AS-6) for the AS-6 distributed computing system [1]. The OS AS-6 consisted of peer operating systems of separate machines making up the AS-6 computer system. Those operating systems interacted through a uniform interface.

The OS AS-6 facilitated interaction between computation processes on the nodes of the AS-6 system through a network together with their interaction with the global network. It provided the possibility for any process in the AS-6 system to use all the devices connected with the nodes of the AS-6 computer both with addressing and with usage of external devices. It also supported a pipeline operation of the computers in the AS-6 system ("computer pipeline") performing real-time processing of large data streams of spacecraft missions. The operating systems of the nodes of the AS-6 system had a special means for parallel processes management, including

task hierarchy organization and information processing management within a single task.

There were two kinds of network software in OS AS-6: transport and functional. The purpose of the transport software was to transport data between operating systems of separate nodes and between different user tasks in the system. The functional software evolved from the transport software and it carried out a variety of operations such as resource request processing, data exchange with input-output devices, and invocation of middleware programs. Since the operating systems of all the nodes were peers, their transport software (in contrast with the functional one) was identical. If some computer within the AS-6 system failed, the network would interact and stopped it; it automatically resumed after restarting its operating system. We used transport software typically for interaction between processes on different computers, except when a special computer managed external devices ("peripheral machines"). For the operating system of a peripheral machine, the transport software also supported interaction between internal processes.

2 AS-6 Mail Functionalities

Interaction between processes in a distributed real-time system should be very effective and have minimum overhead. Because of this, we organized the data transfer mechanism between peer operating systems based on high-speed channels and two-level transport software. The first level, called the *physical mail*, was responsible for message transfer through the high-speed physical channels. The second level, called *user mail*, was responsible for message transfer between user tasks and system processes. The physical mail effectively handled hardware faults such as processing unit outage, incorrect transfer of message data (detected by monitoring circuits), and message loss in the physical channel. The physical mail service of one unit in AS-6 could interact with fifteen parties, which were the physical mail services of the other units.

The main elements of a physical mail service were the following.

- *Physical port*: These are the sending and receiving ports; when connected, these ports make up the data transfer lines.
- *Messages with responses*: A sending port can send a message to a receiving port and the latter can send a response to the former. Each message had a response mechanism.
- *Physical mail request*: Such a request contained a command of the physical mail service and its parameters, including the address of the message beginning and the message's length. Each request contained a reference to the port that should process this request.
- *Directives*: A directive is an interrupt word by which one unit in AS-6 system can send it to another unit in the system.

The physical mail service used two kinds of directives: direct and inverse. A direct directive contained an address of some request. An inverse directive contained a response to some direct directive. A positive response occurred if the request processed normally; otherwise, it was a negative response. The transfer commands

used by a physical mail request included, create or destroy the communication line, send a message and wait for response, receive a message, and send a response.

3 Communication Elements

Both sending and receiving ports could perform the creation of a communication line. Each port in AS-6 system had the unique number. Default or static mail subscribers such as resource managers had standard ports that statically defined numbers. Static subscribers could call occasional subscribers such as device control programs. At any moment of a call, they received ports with numbers generated on the fly.

A command to create a communication line for a receiving port did not cause any data transfer; it only created a record in the rendezvous table of the physical mail service. The service then waited for the create command from a paired sending port. A command to create a communication line for a sending port generated a direct directive referring to this request. When the receiving unit obtained such a directive, its physical mail service looked through the rendezvous table for a request to create a communication with the corresponding sending port. If it found such a request, the inverse directive contained a positive response along with the address of the receiving port; otherwise, it contained a negative response.

After creating a communication, the receiving port could process the commands to receive a message and to send a response. The sending port could send a message while waiting for a response. A command to receive a message created the corresponding direct directive to the sending port. After receiving this directive, the sending port would send a message.

A command to send a message generated a corresponding direct directive to the receiving party, if it had already sent the directive of the kind described above. The directive to receive a message forced the receiving party to read the message data and to synchronize with the subscriber, which then issued a command to receive a message. The processing of a command to send a message ended only after receiving a response to the message. A command to send a response generated a corresponding direct directive. A response could come asynchronously; that is, responses for two messages sent in some order could arrive in reverse order. To match the responses with their corresponding messages, each response contained the address of the request containing its message.

A special initial message sent to another party marked the initial availability of a physical mail service. To monitor the availability of each other, the physical mail service would send each party a special availability message. If for a definite time the service did not send availability message to its peers, it destroyed all the communication lines. The same thing occurred if other parties obtained an initial message from the service. This signaled a quick restart of the corresponding node due to the absence of the availability messages.

The user mail service helped to exchange messages for the user or system tasks when performed on any node of the AS-6 system. The interacting tasks by user mail could occur on one node, on different directly connected nodes, or on nodes having no direct connections. Commands performed by user mail service did not depend on

the location of the interacting task, so the user mail service was a universal task interaction medium.

Subscribers were tasks interacting through user mail service. Each active (online) subscriber had an identifier unique for the entire network. Subscribers could be standard or temporary. A standard subscriber had one permanent identifier; a temporary identifier could have different identifiers on different activation periods. An identifier used by a temporary subscriber could transfer to another temporary subscriber if the first one became inactive. Some examples of standard subscribers include subscriber activation tasks, file transfer tasks, and network output buffering tasks.

A message exchange between two user mail subscribers occurred through a virtual communication line called an association. One subscriber could participate in many associations and in general, there was no limit to the number of associations between two subscribers. A port defined the point of interaction between a subscriber and an association. Two connected ports define an association, which always provided a full-duplex communication.

Data transfer between different tasks could be asynchronous or session-based. The first kind of interaction, called "mail-like", created an association between interacting tasks only at the moment of data transfer. Interaction of the second kind, called "phone-like", preserved this association throughout the session. Mail-like messages transferred independently on each other and in general, they did not preserve their order. A subscriber could receive messages in a different order than the order of their sending. Some mail-like messages could require approval of their receipt.

In the phone-like mode, before the transfer of any data, the system would commutate the association. Special commands closed a communication session. During a session, the user mail service could perform the following tasks.

- Automatic receipt approval of all messages,
- Automatic support of message order preservation,
- Message transfer only if the input buffers of the receiving party have the appropriate empty space,
- Interrupt transfer.

Messages were not sufficient to perform all practical tasks during phone-like communication. Interrupts were used for fast transfer of data of bounded volume (with a speed higher than the speed of a message) and to remove some messages from the association. The system would use the mail-like mode for occasional interactions and used the phone-like mode for intensive interaction on a considerable time interval.

4 System Devices

The operating system of the AS-6 computer directly connected with the corresponding device and for a device, it could assign tasks on different computers. It had special modules called resource administrators that could interact with each other with the help of the transport software (physical mail). Specialized modules of two

operating systems supported the use of a device; one module was on the node where the client task was working and the other module was on the node connected to the device used.

A user would have the possibility to address the devices in the network. This occurred in two ways—by the administrative division where the device was located or by the direct device address. One would use the first mode, for example, to send an output of some task to a device close to the one from which it had taken an input, preferably located in the same room. It was possible to connect different devices from one administrative division with different computers. However, different administrative divisions had no device in their intersection; thus, any device could contain only one administrative division.

5 Conclusion

In this brief paper, we have described only few, very basic features of the OS AS-6. Here we would like to highlight some other aspects that are important in the historical perspective.

At first, as we know the OS AS-6 was the first heterogeneous distributed operating system. Their component OSs based on different internal architectural principals and worked together based only on uniform protocols and interfaces. Moreover, these component OSs were relatively autonomous, and some of them provided services even without any support of other ones.

Second, the above architectural principles were very interesting in their own right. For instance, in the design of some of these OSs we actively used principles that later were called *object-orientation*.

And last but not the least: all this work was done with very close cooperation and collaboration with our customers and friends from Space Control Centers. These joint activities allowed to use OSs and applications to support the unique Soviet-American space mission Apollo-Soyuz and many other critically important projects.

References

1. AS-6 Data Processing System. Russian Virtual Computer Museum. http://www.computer-museum.ru/histussr/as6.htm (in Russian)