N93-32145

Process Control and Recovery in the Link Monitor and Control Operator Assistant

*Lorrine Lee
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109-8099
MS 525-3660
(818)-306-6178

Randall W. Hill, Jr.
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109-8099
MS 525-3651
(818)-306-6085

Abstract

This paper describes our approach to providing process control and recovery functions in the Link Monitor and Control Operator Assistant (LMCOA). The focus of the LMCOA is to provide semi-automated monitor and control to support station operations in the Deep Space Network. The LMCOA will be demonstrated with precalibration operations for Very Long Baseline Interferometry on a 70-meter antenna. Precalibration, the task of setting up the equipment to support a communications link with a spacecraft, is a manual, time consuming and error-prone process.

One problem with the current system is that it does not provide explicit feedback about the effects of control actions. The LMCOA uses a Temporal Dependency Network (TDN) to represent an end-to-end sequence of operational procedures and a Situation Manager (SM) module to provide process control, diagnosis and recovery functions. The TDN is a directed network representing precedence, parallelism, precondition and postcondition constraints. The SM maintains an internal model of the expected and actual states of the subsystems in order to determine if each control action executed successfully and to provide feedback to the user.

The LMCOA is implemented on a NeXT workstation using Objective C, Interface Builder and the C Language Integrated Production System.

Introduction

This paper describes our approach to providing process control and recovery functions in the Link Monitor and Control Operator Assistant (LMCOA). The focus of the LMCOA is to provide semi-automated monitor and control to support station operations for the Deep Space Network (DSN). The DSN is a world-wide network of antennas located in California, Spain, and Australia and is used for communicating with spacecraft. The antennas range from 26 to 70 meters in diameter. The 70-meter antennas are heavily oversubscribed because they are used to support communications for a large number of deep space missions. One way to increase the availability of the 70-meter antennas is to decrease the amount of time spent on overhead operations. Overhead operations are activities where the antenna is not used to support a mission operation. An example of an overhead activity is *precalibration*, which is the task of setting up the equipment at an antenna complex in order to support a communications link with the spacecraft. The process of establishing the link involves issuing control actions, or *directives*, to configure equipment at an antenna complex.

Currently, performing precalibration takes between 30 minutes to over two hours. The time varies according to operators' experience and the complexity of the operation. An example of a complex operation, which most operators don't have experience with, is Very Long Baseline Interferometry (VLBI) Delta Differential One-Way Ranging (DOR), a technique used to determine an accurate measurement of the

position of the spacecraft for navigation purposes. The actual task of performing VLBI measurements takes approximately 15 minutes, whereas precalibration for VLBI takes two to eight times as long. To address this problem, the demonstration domain for the LMCOA was selected to be precalibration for VLBI on a 70-meter antenna.

Precalibration is currently a time consuming process because of limitations in the existing operational monitor and control system. It is a command-line, keyboard-entry system that requires operators to manually send hundreds of directives to subsystems and monitor over a thousand incoming messages on a text-based scrolling log. The system lacks explicit, informative responses about the state of a directive and does not provide guaranteed communications between the monitor and control system and subsystems being controlled. For each directive sent by the operator, the subsystem returns a *directive response* which is simply an acknowledgment from the subsystem that the directive was received. A directive response does not indicate the successful or unsuccessful execution of a directive. The subsystem may also send out *event notice messages*, which relay information about the state of some device in a subsystem. However, these messages are not explicitly tied to any directive that was sent. The operator must rely on his/her experience to determine which directive was most likely to have caused the subsystem to send the event notice message. *Monitor data*, which are sent periodically by the subsystems, also provides information about device states. However, monitor data is never displayed automatically or tied to any directive. Instead, a subset of monitor data is formatted into pre-defined displays that the operator can invoke. The operator has to decide which piece of the data (s)he's interested in and which display contains that piece of information before (s)he can access it.

The absence of guaranteed communications between the existing monitor and control system and the subsystems cause false alarm situations when communication packets destined for the monitor and control system are dropped. The subsystem cannot resend dropped packets because it can't detect them. The number of false alarms inundates the user and hides real alarm situations. Finally, the system is prone to input errors. A simple precalibration pass requires over 200 directives to be issued. The operator must manually identify and type each directive and its parameters. Simple typographic errors can cause a subsystem to take several minutes to recover.

In the existing system, the burden of filtering through the data returned by the subsystems and determining whether execution is proceeding as expected rests completely on the operator. By providing semi-automated control tools and improved data presentation, the LMCOA lightens the operators' load and frees them to do tasks that require human judgment and actions. The overall benefits of the LMCOA are increased DSN resource availability and operator efficiency. The LMCOA prototype will demonstrate the following key features: 1) closed loop control, which provides the operator with explicit and consistent feedback about the executing state of directives; 2) anomaly detection and explanation for directive rejections and anomalous device states; 3) recovery assistance by suggesting alternate plans of action; and 4) a consistent graphical user interface with multi-level displays. The prototype will be demonstrated within the framework of the existing monitor and control system using only the information that is available to the current system.

Temporal Dependency Network

The current system does not have any single source of documentation that outlines operational procedures for VLBI. Rather, it has many operations manuals which are sometimes inconsistent, out-of-date and difficult to use. Therefore, actual operations rely heavily on operator experience and expertise. Given this situation, our first task was to learn the procedures and lay them out in a coherent, consistent manner using a Temporal Dependency Network (TDN).

A TDN, shown in Figure 1, is a directed graph of interconnected nodes that represents an end-to-end operational sequence for VLBI Delta DOR. Sequential, parallel, and optional operation sequences are identified in the TDN. Each node in the TDN contains a block of directives that are sent to the subsystems sequentially. Nodes have precedence constraints where the block of directives in a node cannot be sent until all of its predecessor nodes' directives have successfully completed execution. Each node has associated precondition and postcondition constraints. These constraints define the state the system must be in before the start of each block of directives and after successful execution of those directives. Each node may also have time constraints which limit the start and completion of the directives to a specific time or time interval.

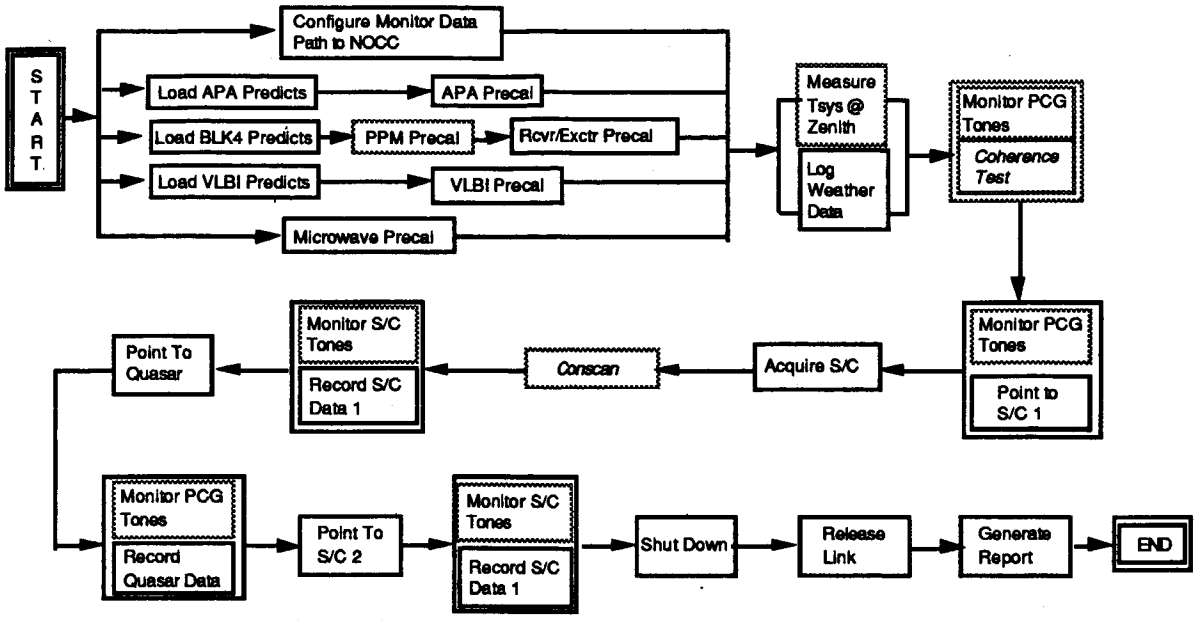

Figure 1. VLBI Temporal Dependency Network.

Architecture

Our goal is to provide both closed-loop control and closed-loop communications for the operator. There are two major modules in the LMCOA which control the TDN execution and close the communications and control loops. These are the TDN Execution Manager and the Situation Manager (SM). Other modules which will be discussed are the Block Execution module, Router, Timed-Events Manager, Monitor Data Handler and DSN Data Simulator. An overview of the architecture is presented in Figure 2.

The LMCOA is an UNIX based, object-oriented, multi-*threaded* architecture. A thread is a very lightweight process that only carries the basic information about the processor state, such as a program counter and hardware registers, that is necessary for independent execution. Many independently executing threads can make up one program. In a single processor system, threads are useful for providing concurrent logical control. The modules in the LMCOA are implemented as individually executing threads that cooperate through message passing and manipulation of shared data objects.

TDN Execution Manager and Block Execution Modules

The TDN execution manager is responsible for traversing the TDN in order to identify blocks and directives that are ready to execute. The execution manager identifies the blocks whose precedence constraints, as laid out in the TDN, are satisfied. (Blocks and nodes will be used interchangeably since a node consists of a block of directives.) Each block that is identified as having its precedence constraints satisfied is executed concurrently as separate instances (or threads) of the block execution module. Multiple block execution threads may be active at the same time because of parallelism in the TDN. By executing each block as a separate thread, we allow the operating system to schedule execution of parallel blocks on a single processor system. A block that is waiting for some event (e.g. a subsystem's response to a directive) does not hold up any other blocks except for its own successor blocks.

When a block first starts execution, it asks the SM to evaluate its block preconditions. A block's directives are sent only after the SM verifies that the preconditions are satisfied. After a directive is sent, a directive response must be received before the next directive in the block is sent to a subsystem. After the last directive is sent and its corresponding response is received, the block execution module asks the SM to evaluate the block's postconditions. If the postconditions are satisfied, the block of directives is considered

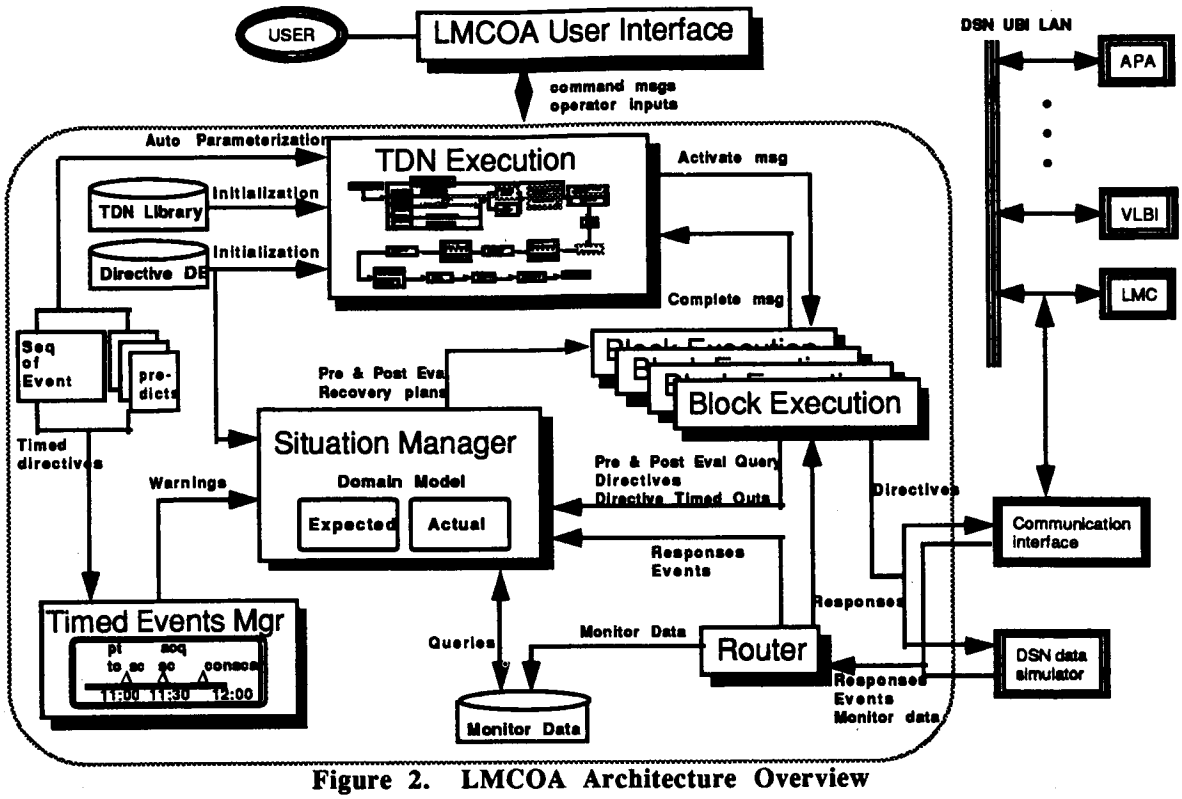

Figure 2. LMCOA Architecture Overview

completed. For each completed block, the TDN execution manager identifies all successor blocks. Each successor block that satisfies its precedence constraints is forked as a separate thread. In this manner, the TDN is traversed and all of the directives in the TDN blocks are sent to the subsystems.

Situation Manager Device Model

The Situation Manager (SM) is an AI-based module that controls the start and completion of each block of directives, verifies correct execution by checking postcondition constraints on a block of directives, detects problems and provides recovery assistance. The goal of the SM is to maintain situational awareness. In order to accomplish this, the SM keeps an internal model of all the monitorable hardware and software devices in the system. Each device represented in the model has attributes that reflect the state of the device. Each attribute has a pair of values: an *expected value* and an *actual value*. The expected value of an attribute is set when a directive is sent to the subsystem. The actual value of an attribute is set when the subsystem sends messages noting state changes in the subsystem.

Every directive sent to a subsystem is expected to effect certain known changes on the states of the devices in the subsystem. In the SM, each directive has a production rule associated with it which sets the expected states of the affected device attributes to the values expected if the directive executes successfully. Thus each time a directive is sent, the expected values of the attributes in the device model are updated. Several data sources are used to set the actual values of the device attributes: event notice messages, monitor data and the operator. All three provide information about the actual state of a device but no explicit information about whether a directive was executed successfully. However, by using information about the expected and actual states of devices, the success of a directive can be determined.

Event notice messages explicitly give the actual states of devices. Since the numbers and types of event notice messages are finite and known, we know in advance which device attributes the message refers to. Therefore, for each event notice message, the SM has a production rule that updates the actual values of the relevant device attributes.

Monitor data are blocks of status information that are sent periodically by the subsystems. Monitor data usually provides more information than event notice messages. Since monitor data is sent only at set intervals by a subsystem, the information is stored in a Monitor Data Database. When the SM needs monitor data to verify the state of a device, it queries the Monitor Data Database for the latest value. The time of the returned monitor data value is checked and if it is more recent then the time at which the directive was sent, this value is used as the actual value.

Finally, the operator is provided a set of pre-defined monitor displays. The information in these displays is not always available from the monitor data blocks. These displays are generated as bit-map displays at the subsystem level and are unavailable to the LMCOA because of format and DSN operational restrictions. Therefore, for certain directives, the operator must obtain information from the displays and enter it into the LMCOA. This information is used to set the actual value of an attribute in the SM internal device model.

Because the kinds of information available in event notice messages and monitor data are known, we can associate with each device attribute the data source which is used to set its actual state. The primary data source is always event notice messages. If there are no event notice messages, the secondary data source is the Monitor Data Database. Finally, if monitor data doesn't provide the necessary information, the final data source is the operator.

Process Control, Anomaly Detection and Recovery

The SM provides control over the start and completion of each block in the TDN. A TDN block must wait for a message from the SM indicating preconditions are satisfied before it can send the first directive. Likewise, a TDN block cannot mark itself as completed until it receives a message from the SM indicating postconditions are satisfied. Once that message is received, the successor blocks of the just completed block can start execution. The SM also controls continuation of the TDN block after an anomaly is detected. It detects the anomaly and constructs a recovery plan to insert into the TDN. There are several methods for detecting anomalies during directive execution: evaluating directive responses and event notice messages, comparing expected and actual values in the device model, and evaluating precondition and postcondition constraints.

The subsystems send several types of directive responses: PROCESSING, COMPLETED, and REJECTED. The PROCESSING directive response means that the directive was received but has not started executing yet. The COMPLETE directive response means that the directive was received and is currently being executed, and when the subsystem cannot interpret a directive, a REJECTED response is received. Except for a few known exceptions, the subsystems will respond to every directive sent with either a COMPLETE or a REJECTED response. In addition, some subsystems may precede the COMPLETE response with a PROCESSING response. The REJECTED response signals an incorrectly formatted or parameterized directive. Again, the numbers and types of REJECT responses are known and recovery is based on existing plans documented in the operations manual. Each REJECT response has a rule associated with it which selects the appropriate recovery plan.

To address the problem of dropped communication packets, the LMCOA sets a time limit on when a response must be received. If the response is not received within a specific interval, recovery is started. Recovery from a timed out directive involves querying the Monitor Data Database or the operator for the actual value of a device. If the actual does not match the expected, it is an indication that the directive was not received and the directive is resent.

Event notice messages provide information about the state of a device including whether or not the device is in an erroneous state. Each time an event notice message is used to update an actual value of an attribute, it is compared to the expected value. If the actual and expected values do not match, then the system is in an erroneous state. Associated with each event notice message that describes an erroneous state is a rule that selects a known recovery plan. Precondition and postcondition evaluations are handled the same way. The actual values of the device states are compared to those defined by the preconditions or postconditions. If the states do not match, then a recovery plan to correct the state is created.

Once a recovery plan is created, it must be integrated into the TDN in order to be executed. A recovery subnet containing blocks of directives, control logic and display directives is created. This subnet carries

the same features as the TDN such as time, precedence, precondition and postcondition constraints and allows representation of parallel and optional sequences. It is inserted immediately after the anomalous block and becomes an integral part of the TDN. Recovery directives are sent and monitored in the same way as any other directive in the TDN.

The communications loop for a directive is closed when the TDN block execution module receives a directive response or completes recovery from a reject message. The control loop for a directive is closed when its block postconditions are satisfied and any recovery subnets are completed. The operator is thus guaranteed appropriate action will be taken for every directive sent.

Router, Timed-Events Manager, Monitor Data Handler and DSN Subsystem Simulator

In addition to the TDN execution, block execution and SM modules, there are several other supporting modules. The Router is responsible for all communications between the LMCOA and the DSN subsystems. It receives and decodes input from the DSN subsystems and directs the input to either the TDN Execution Manager or the SM or both. It also formats the directives into communication packages that are sent to the subsystems. The Timed-Events Manager maintains a list of directives in the TDN with time constraints. Two warning periods are associated with each of these timed directives. As the time nears the first warning period, the operator is advised to take an action such as skipping optional sequences. As the time nears the second warning period, the SM executes a known recovery plan to skip any optional sequences still pending. The Monitor Data Handler receives Monitor Data blocks from the subsystems and stores them in the Monitor Data Database. It also handles queries from the SM and returns the latest value for the requested monitor data element. Because access to the operational environment is limited, a DSN Subsystem Simulator was implemented to simulate the directive responses and event notice messages from the subsystems.

Automated parameterization and Automated Report Generation

One of our goals is to decrease the amount of keyboard entry required of the operator. In the existing system, the operator looks at several sets of paper listings to identify critical directive parameters. In the LMCOA, these listings are accessed in electronic form, parsed, and used to parameterize the directives defined in the TDN. In the existing system, the operator jots down the time at which certain directives were executed and their results. At the end of the pass, the operator writes a set of paper reports. In the LMCOA, we internally log the time, parameters and responses for each directive and the reports are automatically generated at the end of the pass.

User Interaction and Status Displays

Another one of our goals is to provide consistent interaction and meaningful displays to keep the user aware of what's going on in the system. The primary method of interaction is via menu or button selections with a mouse. Occasionally the operator may be asked to enter a value or response. The primary interaction window is a block-level display of the TDN which provides a high-level end-to-end sequence of operations. A color bar in each block is used to show the status and progress of each block: a gray bar means the block is inactive, a green bar means the directives are executing, a red bar means an anomaly has occurred and a blue bar means the directives have successfully completed. The portion of the color bar that is green is proportional to the number of executed directives in the block.

The operator can bring up a lower level display for each block that shows the state of the block, the state of each directive in the block (inactive, executing, paused, anomaly, etc.) and lists the preconditions and postconditions for each block. There are TDN-level controls to pause, resume and stop execution. Block-level and directive-level controls exist to pause, resume and skip execution. Icons are used to show the user whether a block is paused or skipped.

SM anomaly messages that require a user response are displayed in a separate window. A synopsis is displayed in a scrolling portion at the top of the window. By selecting a synopsis, the operator brings up a description of the anomaly in the bottom portion of the window. The operator can then enter the requested input or select a default option.

Other available displays are a scrolling event log, the link configuration display and the device state display. The scrolling event log lists all the input and output to and from the LMCOA system. The link configuration display lists the equipment being used for the pass. (A link is the set of equipment assigned to communicate with the spacecraft. There may be duplicate pieces of the same kind of equipment.) The device state display shows the expected and actual state of each piece of equipment. It provides the operator with a view of the SM device model.

Knowledge Engineering

A major effort in the LMCOA is the process of gathering the information for the TDN. When we started there was no definitive answer as to what is the correct sequence of operations for doing VLBI on a 70-meter antenna, Rather, each operator had his/her own "cheat" sheet of what needed to be done. There were also official documentation, engineers' procedures for VLBI and scientists' procedures for VLBI; all of which differed from group to group and from person to person. The approach for knowledge engineering was to first learn about the system through existing documentation, noting inconsistencies and missing information. The next step was to discuss the procedure with the operators, engineers, technicians and scientists to get their viewpoints and to clear up inconsistencies as much as possible. This information led to our initial TDN. The TDN became the needed common language between our knowledge engineers and our knowledge sources. The LMCOA knowledge engineering effort is the only known attempt, within the DSN, to document a single, coherent and consistent base-line operational sequence for precalibration that merges the viewpoints of the users.

Current Implementation Status

The LMCOA is implemented using C, Objective-C and the C Language Integrated Production System (CLIPS) on a NeXT workstation running Mach. Most of the features described above have been implemented and successfully tested and demonstrated in a laboratory environment with the DSN Subsystem Simulator. Items currently being implemented are the integration of the recovery subnet, the Timed-Events Manager, automated parameterization and report generation, the link configuration display and the device state display. Our plan is to demonstrate the LMCOA prototype at the Goldstone Deep Space Communications Complex's 70-meter antenna in September, 1992.

Conclusion

Knowledge-based systems will play a major and enabling role in improving future ground information systems at the DSN. The LMCOA prototype demonstrates the feasibility and benefits of AI-based automation in DSN operations. The benefits of an operational semi-automated monitor and control system are 1) reduction in precalibration time by at least 50% which could provide an extra 24 hours of antenna availability per week; 2) reduction in keyboard entry by 80% which reduces occurrences of typographic errors; 3) enablement of parallel operations ; and 4) increased operator efficiency via closed loop control. Future efforts will include extending the LMCOA to control multiple activities, demonstrating the LMCOA at a 34-meter experimental antenna complex, extending the LMCOA to support other operations and establishing guidelines for subsystem design to facilitate automated monitor and control.

Acknowledgments

The work described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. The other members of the LMCOA team are Sanguan Chow, Lynne Cooper, Kristina Fayyad and Juan Urista. We would like to acknowledge the contributions of Pam Wolken, Terry Dow, Dave Girdner, Elmain Martinez and the many operations personnel at Goldstone Deep Space Communications Complex.

References

1. Cooper, Lynne P, Desai, Rajiv and Martinez, Elmain, "Operator Assistant to Support Deep Space Network Link Monitor and Control," SOAR Symposium, Houston, TX, 1991.

2. Deep Space Network Information System Architecture Study", JPL Publication D-8916, Jet Propulsion Laboratory Internal Document, September 27, 1991.

3. Miller, Gary and Walrath, Kathy, "NeXT Operating System Software", NeXT Computer Inc., Redwood City, California, 1990.

4. Cooper, Lynne P, "Operations Automation Using Temporal Dependency Networks," Technology 2001, San Jose, CA, December 3-5, 1991.