NASA-CR-196852

NAG 5-995

p. 178

NCC Simulation Model: Phase II: Simulating the Operations of the Network Control Center and NCC Message Manual

Submitted to:

Networks Division
National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt Maryland 20770

Submitted by:

Computational Science and Engineering Research Center
Howard University
2216 6th Street N.W., Washington, D.C. 20001

August 25, 1994

(NASA-CR-196852) NCC SIMULATION MODEL. PHASE 2: SIMULATING THE OPERATIONS OF THE NETWORK CONTROL CENTER AND NCC MESSAGE MANUAL Final Report (Howard Univ.) 178 p

N95-11385

Unclas

G3/32 0022742

NCC Simulation Model: Phase II: Simulating the Operations of the Network Control Center and NCC Message Manual

Submitted to:

Networks Division
National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt Maryland 20770

Submitted by:

Computational Science and Engineering Research Center
Howard University
2216 6th Street N.W., Washington, D.C. 20001

Prepared by:

Norman M. Benjamin
Tepper Gill
Mary Charles

Table of Contents

1. Introduction

The simulation of the NCC is in the second phase of development. This phase seeks to further develop the work performed in phase one. Phase one concentrated on the computer systems and interconnecting network. The focus of phase two will be the implementation of the network message dialogues and the resources (i.e schedulable elements of the SN) controlled by the NCC. These resources are requested, initiated, monitored and analyzed via network messages. In the NCC, network messages are presented in the form of packets that are routed across the network. These packets are generated, encoded, decoded and processed by the network host processors that generate and service the message traffic on the network that connects these hosts. As a result, the message traffic is used to characterize the work done by the NCC and the connected network.

Phase one of the model development represented the NCC as a network of bi-directional single server queues and message generating sources. The generators represented the external segment processors. The server based queues represented the host processors. The NCC model consists of the internal and external processors which generate message traffic on the network that links these hosts. The external processors represented are the POCC, NASCOM, WSGT, FDF, SDPF and JSC. These connect to the internal processors which are the CCS, SPS and ITS. To fully realize the objective of phase two it is necessary to identify and model the processes in each internal processor. These processes live in the operating system of the internal host computers and handle tasks such as high speed message exchanging, ISN and NFE interface, event monitoring, network monitoring, and message logging. Inter process communication is achieved through the operating system facilities. The overall performance of the host is determined by its ability to service messages generated by both internal and external processors.

2. NCC Overview

The NCC is located at the Goddard Spaceflight Center and provides scheduling, monitoring and control of services to the NASA Space Network (SN). The SN provide tracking and data acquisition services to a large community of low-earth-orbiting spacecrafts. Spacecraft data is down linked through TDRS to the WSGT. The data is then quality checked at the NASA NGT which is co-located with WSGT. The data is then forwarded over

NASCOM's telecommunications links to the NCC at Goddard and relayed to the project users. User spacecraft telemetry data do not pass through the NCC.

The NCCDS is specifically responsible for scheduling, control, fault isolation and accountability to ensure that users receive their data as scheduled. The NCCDS is provided with support services from the FDF and the SDPF. The FDF provides tracking data analysis together with orbit determination for spacecraft vector generation. The SDPF receives and processes scientific data for users.

The NCCDS is divided into three major subsystems. The service planning segment (SPS), which schedules TDRSS services; the communication and control segment (CCS), which controls and monitors the quality of active services; and the intelligent terminal segment (ITS), which provides the interface between the console operators and the SPS and CCS. The specific functions of providing the management and resources for scheduling, controlling, and monitoring the performance of NASA's SN are summarized as follows:

- Scheduling of Network Resources

  Forecast Scheduling and conflict resolution
  Real-time scheduling and conflict resolution

- Network Performance Monitoring

  Ground equipment status messages from WSGT
  Data quality messages from NGT

- Acquisition Data Management (Secure and Un-secure)

  Non-secure- Routing of data from FDF to WSGT
  Secure: Generation of pointing data using the Acquisition Data facility software and subsequently transmitting vectors to WSGT.

- NCC Database Management

  Maintain a library of all current databases.
  Restores data as required.
  Troubleshooting of database anomalies
  Maintain all database documentation

- Network Fault Isolation

  Real-time fault isolation which is accomplished by teams of Performance Analysts and TDRSS Network Controllers.

Post-event analysis performed by teams of TDRSS Network Analysis who evaluate electronically logged messages in the NCC data systems.

- Network Accountability and Reporting

  Operation of a Service Accounting System (SAS) that counts the data messages received from user services and uses the data to compute time and resource usage for billing purposes. This data is also used by the SN Network Anomaly Committee to evaluate and resolve all documented network anomalies.

3. Measures of Performance for the NCC

3.1. Framework

Having decided to use a queuing model to represent the NCC, we are constrained by queuing theory to a limited set of performance measures. These measures are:

Mean waiting time
Mean service time
Mean delay (waiting time plus service time)
Mean queue length

These are fairly stable statistics that should not vary greatly from one run to the next, if, of course, the run-times are sufficiently long to attain stability in the process.

Other measures such as:

Maximum waiting time
Maximum service time
Maximum delay (waiting time plus service time)
Maximum queue length

can be observed, but they are likely to vary substantially from one run to the next, even if the run-times are long and the system attains stability.

Still other measures such as:

Probability that the waiting time exceeds some threshold value $P(W > T_W)$
Probability that the service time exceeds some threshold value $P(W > T_S)$
Probability that the delay exceeds some threshold value $P(W > T_d)$
Probability that the queue length exceeds some threshold value $P(Q > q)$

can be computed, but the problem would be to establish threshold values that have some physical or operational significance.

We have suggested one measure of performance that is not typical that is "the minimum capacity of the server required for zero wait or no queuing". This measure will help to establish the required capacity for the processors being modeled.

At the current stage of development, all of these measures can be observed for the NCC as a system, and the CCS, the SPS, and the ITS as subsystems. We are proposing that the next logical step in developing this model should be to separate the manual processing from the automated processing. This should have a tremendous impact on the fidelity of the model. It will allow us to take a more analytical look at different processes with the goal of optimizing the allocation of tasks and processes to manual and automated servers. It will allow us to conduct "What If" analyses. It will provide insights into enhancement options for the NCC, and it can be the basis for a subsequent elaboration of the model to include cost as a measure of performance.

3.2.Selected Indicators of the NCC'S Operational effectiveness

The NCC has established specific quantifiable requirements that it must achieve in providing services to SN users. These requirements have been analyzed as a basis for selecting the following indicators of the operational effectiveness of the NCC:

3.2.1. Utilization of NCC's Communications Capacity

- Average communications capacity utilized by incoming messages (single user)
- Percentage of times that incoming messages (single user) exceeds 56 kilobits per second
- Maximum communications capacity utilized by incoming messages (single user)
- Average communications capacity utilized by incoming messages multiple user
- Percentage of times that incoming messages (multiple user) exceeds 112 kilobits per second
- Maximum communications capacity utilized by incoming messages multiple user
- Average queue time at the CCS from incoming messages
- Minimum communication speed to ensure zero queue time at the CCS from incoming messages
- Maximum queue time at the CCS from incoming messages

- Average communications capacity utilized by outgoing messages (single user)
- Percentage of times that outgoing messages (single user) exceeds 56 kilobits per second
- Maximum communications capacity utilized by outgoing messages (single user)
- Average communications capacity utilized by outgoing messages multiple user
- Percentage of times that outgoing messages (multiple user) exceeds 112 kilobits per second
- Maximum communications capacity utilized by outgoing messages multiple user
- Average queue time at the CCS from outgoing messages
- Minimum communication speed to ensure zero queue time at the CCS from outgoing messages
- Maximum queue time at the CCS from outgoing messages

3.2.2. Operational Effectiveness Measures

The following are suggested as indicators of the operational effectiveness of the NCC:

- Average delay [processing time plus queuing time] at the NCC
- Average delay at the CCS
- Average delay at the ITS
- Average delay at the SPS
- Average processing time for the NCC
- Average processing time for the CCS
- Average processing time for the ITS
- Average processing time for the SPS
- Average queue time within the NCC
- Average queue time at the CCS
- Average queue time at the ITS
- Average queue time at the SPS
- Maximum queue time within the NCC
- Maximum queue at the CCS
- Maximum queue at the ITS
- Maximum queue at the SPS
- Average Utilization--NCC
- Average utilization--CCS
- Average utilization--ITS
- Average utilization--SPS
- Minimum capacity of the NCC to ensure zero queue
- Minimum capacity of the CSS to ensure zero queue
- Minimum capacity of the ITS to ensure zero queue
- Minimum capacity of the SPS to ensure zero queue

3.2.3. Acknowledgments and Response Time

- Percentage of times the NCC fails to send response to originator of specific schedule request within one (1) minute of receipt of request.

3.3. Operational Scenarios for the Model

o What is the effect on the NCC upon losing a particular SN service (SA or MA antenna(s))?

[Actual loss of the service will no be modeled, however, an estimation of the arrival distribution of the SDRs and SARs due to the loss will need to be identified]

o What is the effect on the NCC of adding new users?

[An increase in the number (and arrival rate) of all message types will need to be determined using the NPAS mission and extrapolation

o What will be the effect on NCC's performance of changing from WSGT/NGT to STGT

[The FIMS and NSS messages will be removed and the number of bursts of SHOs sent will be reduced but the size (number of SHOs) of the bursts be larger]

o What will be the effect on NCC's performance of adding another TDRS?

o What will be the effect on NCC's performance from installing faster computers at the CCS, ITS, and SPS?

[Change the service rates for the respective processors, and observe changes in the NCC's performance]

- o What will be the effect on NCC's performance from incorporating improved scheduling algorithms (more flexible generic request parameters, better conflict resolution, more automation)?
- o What will be the effect on the NCC's performance from incorporating improved priority schemes?
- o In the current configuration, at what point will the NCC become saturated?

3.4. Other Measures of Performance

The need to identify other measures of performance for the NCC arose from the inability of the simulation model to easily address issues as they relate to the end-user. These limitations inherent in the simulation model is due to some extent to the absence of the modeled network resources. Since the primary function of the NCC is to schedule user access to resources, and to monitor and maintain access schedules, measures associated with resource availability and allocation must be investigated.

Since resources are not implemented in the simulation model of the NCC, the definitions provided here will serve only to bring awareness of the existence of these measures, and to provide simplified computation methods of these measures where possible.

The NCC network performance criteria can be defined in terms of the following nine performance indicators. These indicators are used to assess how well a network handles information exchange. The performance indicators are:

1. Transfer Rate
2. Network Delay
3. Channel Establishment Time
4. Line Turn-around Time
5. Availability
6. Reliability
7. Accuracy
8. Maintainability
9. Security

The current NCC simulation model developed by the Howard University team addresses the first four indicators. The last five indicators will now be defined.

In section 3.1, titled Framework, Transfer Rate and Line Turnaround time are defined and implemented as constants in the simulation model. Although the Space Network uses circuit switching technology it was determined through a design decision to omit this criterion.

Network delay and message turnaround time are computed using the selected indicators of the NCC effectiveness.

3.4.1. Introduction to Measures of Service to Users

NASA's network, like any other network, must provide a service to its customers. The communications environment must match the customer's needs on a service by service basis. Starting with the deployment of appropriate architectures and reliable components, the network must have an effective operational support. Operations support must also include rapid recovery procedures.

Complex systems, like the NCC, are usually constructed using one or more simple systems. The simple systems are built using standard design principles and manufactured from well tested components. In such a complex system, the ability of the system to fulfill its mission goals is based on factors other than the system's performance. These additional factors that influence the systems ability to fulfill its design purpose are *the operational effectiveness, the operational readiness*, *the capacity*, and *the system cost*. These four factors can be used to define the system's measure of service. Welker and Horne defined this measure of service as the system effectiveness. A system's effectiveness is *the probability that the system can successfully meet an operational demand within a given time when operated under the conditions specified by the design* [Welker and Horne 1960]. This definition suggests that issues associated with design and manufacture, maintenance and administration and logistic support all impact the system's effectiveness.

3.4.2. Measure of System Service

The typical criteria used in the determination of a system's performance are *transfer rate, network delay, throughput and response time*. Most systems are designed to provide a service to a specific user community, and that community should be able to determine the level of service they are receiving. An indication of the level of service will act as an indication of how much of a return they are receiving on their investment in the service. The focus of these measures should be to provide indicators of the level of service that is provided to the users. Indicators of service that can give a better view of the system from the users perspective are *availability, reliability, transparency, security, fault and cost.* There are many varying definitions of these measures so it is important to define what these measures mean in the context of measuring system services of the NCC. The definition must fully define the scope of the measure and must allow the property to be described in quantitative terms. The following definitions were derived after extensive examination of the services provided by the NCC and from an extensive literature search.

3.4.3. Units of Measure

The primary goal is to determine, from the users perspective, how well the system provides the service for which it was intended. By design the system is intended to perform one or more services during a specific time period. The calendar time is usually chosen as the fundamental unit of measurement for the period during which the system is working. System service attributes will be defined over or within the operational period. The intervals of interest to this study are the ones during which the system is operating (operating time) and not operating (down time). System operating time is the time during which the system is operating in a manner that permits usage of all its resources. Down time is the total time during which the system is in an unacceptable operating condition. This means that all resources are not available.

Down time can be subdivided into repair time, logistic time and administrative time. Repair time is the time during which the repair crew is actively working to repair the system. Logistic time is the portion of the down time during which it is necessary to wait for a replacement part, or availability of repair crew. Administrative time is the necessary activities that are required for system accounting, management and the building of audit trails.

Indicators of system service may also be a measure of the system's capacity or the ability of the system to make available to the user some amount of usable capacity.

3.4.4. Indicators of NCC System Service

Through the use of the units of measures defined above, we can define the system service as the sum of operational readiness, operational effectiveness, measures of cost, and measures of capacity. These indicators will be defined in the context of the NCC.

3.4.4.1. NCC Availability

NCC operational availability is defined by NASA [534-FPR-NCC] as a measure of the system's availability during active service or standing ready to provide active service. This operational availability is determined for each function performed by the NCC using the following variables.

1. the mean time between failures (MTBF).

2. the mean time between maintenance (MTBM) [this includes both scheduled and unscheduled maintenance actions.]

3. the mean down time (MDT)[this includes schedule and unscheduled maintenance, logistics delays, and administration delays.]

Availability is that portion of the selected time interval during which the information path is capable of performing its assigned data communication function [GRUBB 75]. In addition to failure and repair times, the availability of the space network is affected by shuttle launches, high priority missions, and emergencies.

Space Network availability thus defines, in an operational manner, the time during which a user may have access to the network and its resources. This definition and all that follows exclude any reference to psychological variables. The definitions refer only to those variables that can be objectively quantified. The following appears to exhaust the independent elements that may affect availability:

1) Downtime (maintenance, failures, etc.)
2) Shuttle Launches
3) High Priority Missions
4) User Spacecraft Emergencies

This information suggests that availability must be defined in terms of the system's operating time and down time. From the users perspective, the system is available if it is capable of performing its intended function when called upon to do so. This view focuses on a point in time rather than on the interval between downtime and priority missions. The *availability is thus defined as the probability that the system is operating satisfactorily at the point in time when used under the predefined conditions*

3.4.4.2. Quantifying NCC Availability

Operational availability is defined using the system's down time and operational time.

$$Operational\_Availability = \frac{MTBF}{MTBM + MDT}$$

As noted earlier the normal operations of this system is affected by priority missions, administrative delays, emergencies, and logistics delays. This means that system availability will be significantly reduced during those periods of shuttle launches and spacecraft emergencies. This reduced availability is a result of an increase in priority traffic and unavailability of key network resources due to saturation. Since these high priority missions and emergencies do not indicate a system failure they can be characterized by increased network activity.

This suggests that variables associated with availability should be refined to include level of the system traffic. The volume of traffic in the system will determine the degree of congestion that exists. This suggests the following definition:

$$Operational\_Availability = (Level\_of\_network\_traffic) - \frac{MTBF}{MTBM + MTD}$$

in which Level of Network traffic is defined as the percentage of unused bandwidth. The level of network traffic is computed directly from the system utilization.

$$Level\_of\_network\_traffic = Network\_capacity - Fraction\_currently\_utilized$$

The required availability for critical NCC functions must be greater than 0.9998. Non-critical NCC functions availability must be greater than 0.9990[534-FPR-NCC].

3.4.4.3. NCC Reliability

A composite definition of reliability of a system that is made-up of multiple independently manufactured components, is as follows:

> Reliability is defined as the probability that every component of the system will perform satisfactorily for some given period of time when used under predetermined conditions. [Welker and Horne 1960].

What is of primary concern to the users is the satisfactory performance of the network. Here reliability is the probability of non failure of the network for the period of time that the network requires to complete its mission. Reliability issues are generally associated with the way a system is organized to serve the user. This organization is centered on the tolerance of the network to complete an initiated request.

The STDN 203.6/NCC definition of reliability is in terms of time critical and non-time critical functions. For NCC specific tasks, reliability must be measured by the network's ability to setup a circuit, transmit the data, then tear down the circuit. By definition, reliability can be improved by enhancing the network fault tolerance. This is accomplished by adding redundant network components.

3.4.4.4. Quantifying NCC Reliability

A classical approach to quantifying the reliability of the a system is through the process of defining a reliability function that expresses the probability of non failure as a function of the time period of operation. Here we are concerned with the user's perception of the network reliability. A user's perspective of the network reliability can be derived by examining the user's view of the service provided. *If a service works the way a user wants it to work when the user wants that service then that service is considered a reliable service.* Service reliability is thus only achieved through the ability of the system to continuously provide a high quality service, while gracefully absorbing failures or unauthorized intrusions into the system. This must be achieved with little or no impact on the level of service provided to the users.

Reliability can now be defined in terms of the quality of the service provided, the survivability of the system, and the security of the system.

Service Quality = the ability to provide a service that meet the needs of customers under pre-defined conditions.

System Survivability = the ability to maintain services in progress and to deploy new services to support customer survivability expectations.

System Security = the ability to minimize intrusions into the system components.

Quantifying system reliability thus becomes a matter of deriving measures that indicate the technical effect on the system of events such as lost capacity, and also deriving variables that indicate loss of utility to users. These variables can be obtained from parameters of the down time , and the workload characteristics. From the down time event we may obtain variables such as *length of failure*, *services affected*, and *reason for service failure*. From the workload characteristics we get the usage information at the time of the failure.

The loss of utility to users will be used to depict social impact measures. This is a measure of the impact of the system failure and lost workload capacity on the user community. The workload may be quantified as the traffic generated as a result of the service being provided to the user. This may be a linear measure similar to the computation of lost person hours as a result of a disaster situation.

$$Social\_impact = traffic\_rate * downtime$$

Using a logarithmic scale the social impact can be normalized.

$$Social\_Impact = Log_{10}(traffic\_rate * downtime)$$

3.4.4.5. NCC Accuracy

The ANSI X3.44 defines accuracy as a measure of the undected error rate. An error has occurred if information received at a node is incorrect, is incomplete, is duplicated, or transmitted but never received. History has taught us that errors occur in bursts. Space Network accuracy is enhanced by the encoding techniques employed in communication equipment. To some degree accuracy will be affected by reliability, since increased fault tolerance will also reduce the possibility of errors.

3.4.4.6. Quantifying NCC Accuracy

The task of quantifying accuracy is difficult because of its relationship to other measures of performance. These relationships must be objectively defined to determine their weight on the accuracy measure. When an error is detected, for example, that information must be retransmitted. The retransmission of the information results in an increase of the network traffic. The use of encoding adds some degree of delay to the overall network and also an additional level of complexity. The encoding function, may also be considered an additional point-of-failure (point of error generation). Network accuracy may be defined as *that percentage of the total bits transmitted that are not error-free*.

$$Accuracy = \frac{undetected\_errors}{total\_bits\_transferred}$$

3.4.4.7. NCC Maintainability

During system maintenance some or all the system resources are off-line. Maintenance is thus some fraction of the system downtime. The measure of *maintainability is thus the probability that the failed network can be restored in a specific downtime*. Down time is the sum of the repair time , logistic time and administrative time.

For the NCC maintenance is the total repair time due to relevant failures divided by the total number of relevant failures. The performance requirements for the NCC specify the mean time to repair (MTTR) should not exceed thirty minutes.

3.4.4.8. Quantifying NCC Maintainability

In complex systems down times are allocated to specific failures as a function of the complexity of the failed component. Since system maintainability measures the entire system, it forces the measure of maintainability to be a mean value. This is the total down time due to relevant failures divided be the total number of relevant failures.

$$Ma\,\mathrm{int}\,ainability = \frac{\sum down\_times}{\sum failures}$$

3.4.4.9. NCC Security

System security is a measure of how secure the system is to internal or external unauthorization access and subsequent illegal activities. The security subsystem may implement static and dynamic security facilities that may have a direct impact on the system performance. These include the use of security protocols, information encryption, and authentfication features. The implementation of these features increases the network overhead.

The interest here is to define a measure of confidence whereby the user feels assured that all information on the network is protected from unauthorized access.

3.4.4.10. Quantifying NCC Security

Security has the same type of attributes as accuracy and thus can be quantified in a similar way. System security is a measure of the systems resistance to intrusions. This is quantified by computing the percentage of unabated intrusions into the network.

$$Security = \frac{un\,\mathrm{det}\,ected\_\,\mathrm{int}\,rusions}{\mathrm{det}\,ected\_\,\mathrm{int}\,rusions}$$

3.4.5. Application of NCC System Service

The above definitions clearly show some relationship between the measurement components that can be viewed graphically as in figure 3.1.

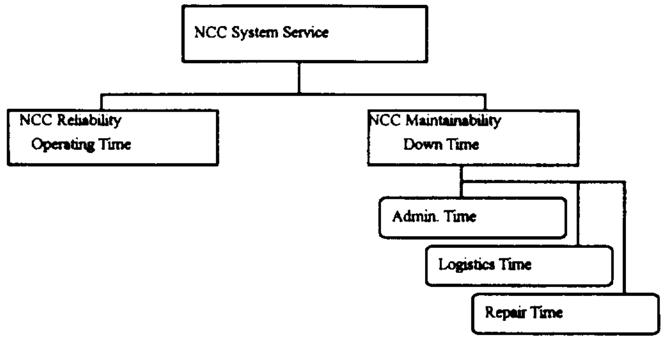


Figure 3.1: Relationship between measurement criteria

This view allows us to consolidate all these system properties into a measure of the *system's service.* The use of the *system service* measure can be used by system administrators to estimate how satisfied the user is with the level of service received from the network. System service is the sum of the operational time and the down time, and can be used to quickly estimate user satisfaction. This user satisfaction materializes as response time, unused capacity, and system reliability. It is important to note that these measures are designed to report only on that particular service in which the user has interest.

The NCC is a network that simultaneously services many user requests. A request for service in this network is represented by a collection of messages that are sent to specific nodes to gain access to network resources. An example of a service is Schedule Dissemination. During schedule dissemination to POCCs the *active scheduler* or the *forecast scheduler* process will initiate the dialogue with a user schedule message 9401, or a 9402, or a 9403. The POCC will respond with an 0360 acknowledgment.

Since each service is made up of a constant collection of messages that traverse the network through a fixed set of nodes, then each service can be represented by a graph that is made-up of those nodes and links that are essential to that service. Figure 3.2 show the graph of the Schedule Dissemination Service.

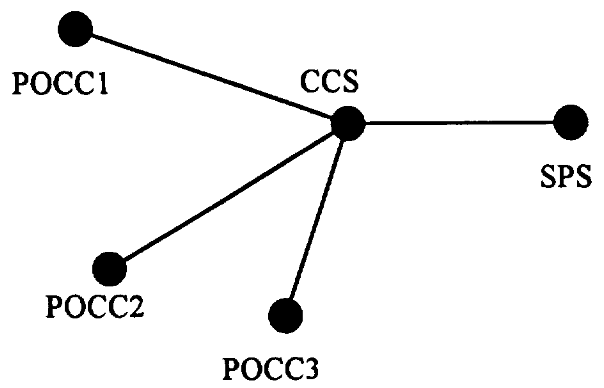


Figure 3.2 Schedule Dissimination Service

The input attributes for the service graphs will be obtained from the statistics obtained from the simulation model output or any other reliable source. Constant network parameters such as link capacity and service times are taken from the existing system.

Notation

N	Node set with |N| nodes
N_i, N_j	nodes i and j
L	link set with |L| links
$L_{i,j}$	link between nodes i and j
$c_{i,j}$	link capacity
p,q	node and link reliability, unreliability for all links; where p+q =1
G(N,L,p)	graph (N,L)
R(G)	reliability of G
A_i	event that node i is up
$B_{i,j}$	event that link i,j is up

3.4.5.1. Response Time

One of the primary concerns of the network user is the network response time. Users expect an almost instantaneous response to requests regardless of the complexity of the request. Service response times are computed by assigning mean link and node delay times obtained from simulation runs to the graph of the service.

$$Response\_time = \sum Link\_delay + \sum Node\_delay$$

3.4.5.2. Unused Service Capacity

A simplified application of the network availability is realized through the definition of Unused Service Capacity. Since the network is made-up of independent nodes with multiple links between the nodes, the network services can also be represented as a directed graph in which the edges represent the link capacity between nodes. The failure of a service or link is represented by removing nodes and links from the graph, using some probability of failure function. The indicators of system service will then be computed using the theorems of Network flows. The variables will be link capacity, capacity of a cut, maximum and minimum flows, and saturation.

The unused capacity of a link or node can be computed using simulation output statistics and our definition and constants of Operational Availability. The unused capacity of a link or node is defines as:

$$Unused\_capacity = Assigned\_capacity - Utilization$$

Unused capacity is more clearly defined as the fraction of the operational time during which a resource is idle. For any given service the unused service capacity can be computed by assigning to the graph of the service the appropriate values to the links and nodes.

$$Unused\_service\_capacity = Min\{Unused\_capacity$$

3.4.5.3. NCC Service Reliability

The reliability of a service is defined as the probability that any two nodes in each and every node-pair that constitute that service can communicate with each other when called upon to do so.

Assumptions

Maximum flow in a link $\leq c_{i,j}$

Let

A_i be the event that node i is up
Bj be the event that node j is up
$L_{i,j}$ be the event that link between i and j is up and is not saturated
p_i probability that node i is up
q_i probability that node j is down

p_{ij} probability that link ij is up

Since every event is preceded by a communication test, it can be assumed that a successful communication test will indicate that the required nodes and links are available and functioning. Probability of failure of links and nodes can be obtained from historical management data of the nodes and links. The probabilities that will be used are those obtained from performing a communication test. Therefore if $D_{i,j}$ is the event that the communication test was successful, then $Pr[D_{i,j}] = Pr[A_i \cap B_j \cap L_{i,j}] = p_i * p_j * p_{ij}$ defines the probability of the successful completion of the data transfer. The reliability of the service is then the reliability of the sub network G which is defined as

$$R(G) = \Pi \; Pr[D_{i,j}] * (1 - \Pi \; Pr[D_{i,j}])$$

where $1 - \Pi \; Pr[D_{i,j}]$ is the probability of failure.

4. Scope of Model

In this version of the simulation model of the NCC the objects that represent the NCC internal processors are being enhanced. The processors are the CCS, SPS, and ITS. These objects will now include representations of the major processes that run on these computers. Each processor, (CCS, ITS or SPS), will contain a single server that will act as the central processing unit (CPU). The CPU will be responsible for servicing the processes of these hosts on a priority basis. The CCS processes modeled are the interfaces to both the ISN and NFE, the high-speed-message-exchange, the network monitor, and the event monitor. The active and forecast schedulers, and the acquisition processes are represented on the SPS. The display request process is the only process of the ITS that is modeled.

NCC network dialogues are also implemented. Network dialogues are any specific set of messages that are necessary to complete a task. Dialogues take place between external segment processors and one or more of the NCCDS. An example of a dialogue is the schedule dissemination. During schedule dissemination to POCCs the *active scheduler* or the *forecast scheduler* process will initiate the dialogue with a user schedule message 9401, or a 9402, or a 9403. The POCC will respond with an 0360 acknowledgment. The following message dialogues between the NCC and the external segments have been implemented.

Message Dialogues:

Ground Control Message Requests
Schedule Dissemination
Performance Data Dissemination
Return Channel Time Delay Measurement
Schedule Services
Acquisition failure Notification

The arrival statistics of all messages will be provided by NASA/532. The dialogues will be initiated by setting up statistical distributions to trigger the first message in the dialogue sequence. The message statistics will be added to the model by the user at runtime.

5. System Abstraction and Model Description

5.1. System Abstraction

The system abstraction and model description that was done in the Phase I model was modified to eliminate the representation of the NFE. The NFE will be synthesized into the interface between the external segments and the NCC. This change will not affect the implementation accuracy of the model since the actions of the NFE is not considered in the problem analysis. These model abstraction changes will reduce the global view of the network to two distinct set of objects. The objects are classified as the *external segment objects* and the *internal segment objects*. A significant reduction in the runtime performance of the model is also expected.

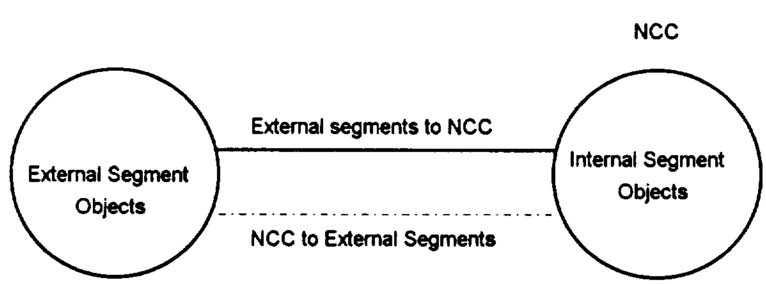


Figure 4.1: Model Abstraction: Network View.

The inter-relationships between the segments of the system as seen from a performance point of view is directly related to the message traffic generated by each object. This approach to abstraction will allow the objects to be represented as processes and generators and the message traffic will characterize the work done in the model.

The *external segment* is mainly concerned with the generation of messages that are sent by the external processors to the NCC. A message generator object will be use to generate messages to drive the simulation model based on data collected from NCC log tapes. The model will be required to handle the generation of messages for an eight hour shift. Stochastic distributions will be attached to each message during each shift to ensure that the rate at which messages are generated closely matches the actual message traffic pattern in the real system. The arrival rates of each message type/class will be computed from the sample data collected. The arrival rates will be used as input into the model as parameters of the stochastic distribution function. These functions are used to determine the times at which messages will be generated.

The *internal segment objects* represent the NCC. The model of the NCC is made-up of three fixed communication nodes that represent the CCS, ITS, and SPS, see Figure 6.2. Each of these fixed communication nodes has incorporated into it, a unique node object. This was necessary because of the functional difference between the CCS, ITS and SPS. The internal segment objects are externally linked to the external segment objects via the CCS by two message streams. The SPS and ITS are linked to the CCS by the ISN which is a fiber based Ethernet LAN. In this phase of the modeling effort, the ISN is represented by a virtual Ethernet backbone model. This virtual backbone is a media access protocol developed in the form of an OPNET process model. The virtual backbone does not require a physical bus implementation. However it, has all the attributes of a bus except for the collisions associated with this type of CSMA/CD protocol. All processors have the ability to initiate and process message dialogues.

5.2. Model Description

The model is represented by two sets of objects connected by two uni-directional streams. The objects are representations of the ***external segment*** and the ***internal segment***. The streams act as an interface between the external segments and the NCC. The *external segment* object will house the all the sources and sinks that are external to the NCC. The *external segments* represented are the FDF, NASCOM, NGT, POCCs, SDPF, and WSGT. The NCC houses the CCS, ITS, and SPS. The CCS, ITS, and SPS contain servers that are used to process incoming messages and generators for message creation. The *external segment* objects contain one or more message generators and a sink. Each generator will be linked to an object called the segment queue that is used as an interface between the CCS and the *external segments*. The link between the generator and the segment queue is a uni-directional stream. Uni-directional streams also link the segment queue to the sink process of the external segment object. All process timing is controlled by the generation of interrupts.

5.2.1. Server Process

The server processes perform all the required message processing within any processor. This object receives messages from any number of sources. The messages are held for a simulated duration that is equal to the *service time* specified for that message. The message is then forwarded to its destination module. The forwarding algorithm uses the message type_class as an index to determine where to send the message.

The possible message sources are streams and sub-queues that may be part of the server object. The server polls the sources, based on a predetermined priority, to access messages waiting to be processed. When a message is found it is assigned its simulated service time then moved from the source to a destination specified by the message path field in the message.

5.2.2. Generator Process

The generator processes are process objects that are responsible for the generation of messages within the system. The generator uses a message traffic definition record of the following format:

type_class	The message type/class
path	Message path through the processors within the NCC
mean	Message traffic mean
variance	Message traffic variance
destination	Message traffic destination
distribution_type	Message generation distribution type

The generator uses the *mean* and *variance* to generate messages of that specific type/class based on the outcomes of the distribution. The destination is the final destination of the message. The path is the route through the sub-processes of the CCS, ITS, and SPS that the message will take.

5.2.3. Link Facility

Connectivity between the objects is accomplished through point-to-point links and a virtual bus. The bus exist between the CCS, SPS, and ITS. The bus represents the ISN. All other objects are connected by uni-directional point-to-point links.

5.2.4. Interrupt Facility

The heartbeat of the model is based on four events. The first occurs when it is time to generate a new message. The inter-arrival times of these events are characterized by the parameters of the distribution function assigned to that message type. All message generators contain distribution functions that generate interrupts which initiate the creation of a message. The second type of events are the class of stream interrupts that are generated when a message arrives at an object. This type of event is automatically generated whenever a message sent from one object to another arrives at its destination. The third event is a service completion interrupt. This event is a server generated interrupt and it signals the completion of simulated service to a message. All post message servicing activities are initiated by this event. The final event is the dialogue response event. Messages that are a part of a dialogue sequence will either be destroyed or trigger another message. The receiving processor will determine if the message received needs a response or is triggering message. If either case is true the creation of the response message will be scheduled with the generator.

5.3. NCC Simulation Model Work Characterization

A measure of the work done by the system is obtained by summing the simulated processing time for each message by each processor. NCC host processors perform pre-determined operations on messages by allocating each message processing time on its server. These operations are associated with logging, monitoring, routing, and scheduling. The simulation of these operations is accomplished by holding the message in a queue representing the process for some specified time. The work done by any processor can be computed as a function of the total time spent servicing messages.

5.3.1. Internal Processes

The processes identified in the CCS are *the NFE and ISN interfaces, the high speed message exchange, the logger, the network monitor and the event monitor*. All messages entering the CCS travel to the high speed message exchange and simultaneously logged by the logger. The forwarding algorithm uses the path field of the message to determine the path the message takes through the processor. The ISN provides the interface between the CCS and the ITS and the SPS.

The SPS consists of four distinct processes. These are the acquisition process, the SAR router, the active schedule process and the forecast schedule process. Messages entering the SPS are routed to the acquisition process or the SAR router. From the SAR router the messages are passed to other SPS processes.

Messages to the ITS are destroyed. Those leaving the ITS are sent to the ISN were they are routed to their respective destinations.

5.3.2. Internal Message Paths

Messages generated by the external processors first enter the CCS internal processor. Messages entering the internal processors follow specific paths to their final destination. Messages generated by the internal processors pass through the CCS as they travel to the intended external processor. Combination of the possible routes of the messages through each internal processsor led to the identification of distinct paths.

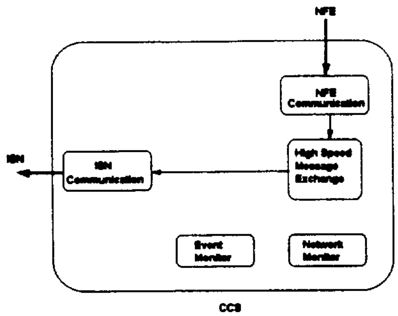
5.3.2.1. CCS Message Paths

Messages entering the CCS are created by the external segment processors, the ITS and SPS or the network and event monitors within the CCS. All messages entering the CCS go to the logger and the high speed message exchange. Messages to the logger are sent to a sink. Those to the high speed message exchange are routed to either the ISN, the network monitor or the event monitor. Messages to the ISN proceed to the SPS or the ITS for processing. Messages sent to the network monitor initiates dynamic display messages before they are destroyed. These dynamic displays are routed to the ISN to be displayed on the ITS. Messages sent to the event monitor are the GCMR's. These initiates GCM messages before being destroyed.

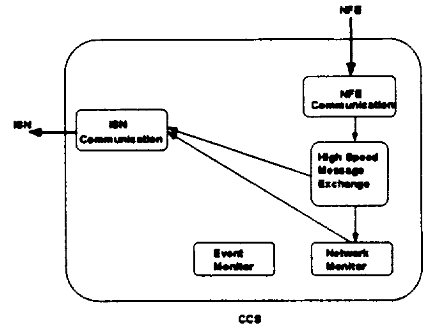
Messages generated by the ITS are sent to the CCS via the ISN. The ISN broadcasts these messages to the network monitor where they initiate the generation of other messages before being destroyed. The newly generated messages are sent to the high speed message exchange where they are routed to the respective external generators.

Messages generated by the SPS are also sent to the CCS via the ISN. The ISN broadcasts these messages to the high speed message exchange where they are routed to the respective external processors.

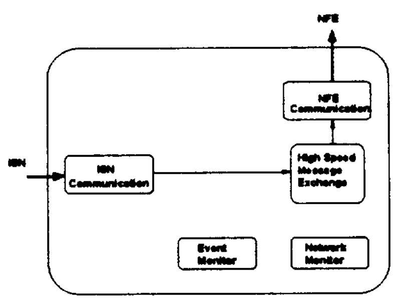
Seven distinct paths were identified for the CCS processor. These paths and the messages traversing them are described below:



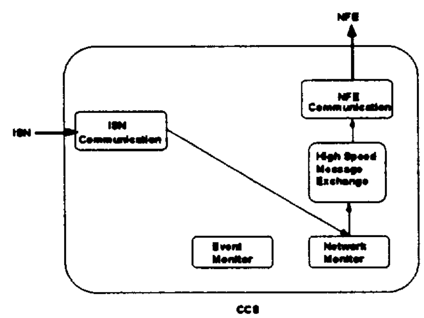
PATH 1 NFE -> high speed message exchange ->ISN



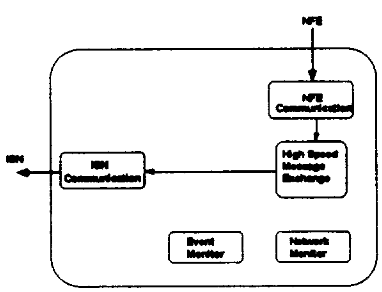
PATH 2 NFE -> high speed message exchange -> network monitor -> ISN



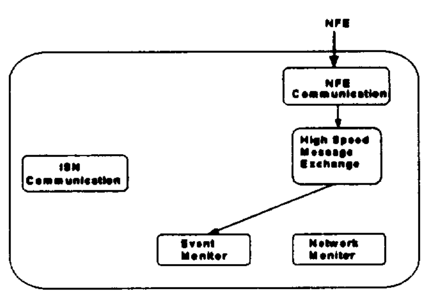
PATH 3 SPS -> ISN -> high speed message exchange ->NFE



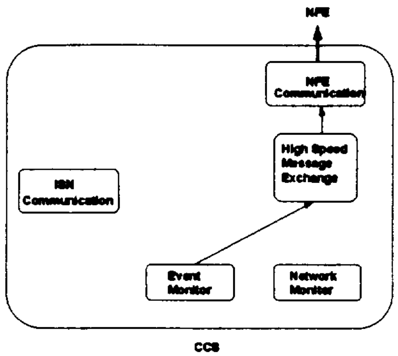
PATH 4 ITS -> ISN -> network monitor -> high speed message exchange ->NFE



PATH 5 NFE -> high speed message exchange ->ISN



PATH 6 NFE -> high speed message exchange -> event monitor



PATH 7 Event monitor -> high speed message exchange -> NFE

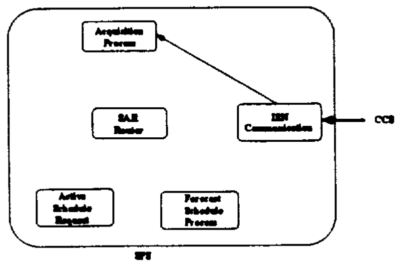
5.3.2.2. SPS Message Paths

Messages through the SPS can be generated by a SPS process or can be received via the ISN from the CCS. Messages received from the ISN travelled through the CCS according to path 1. As a result the path of all messages entering the SPS is defined as path 1. These messages are routed to the acquisition process or the SAR router.

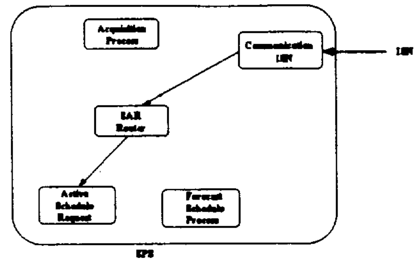
The acquisition process routes the messages it receives back to the ISN to be broadcast to the CCS. The SAR router routes the messages it receives to either the active schedule process or the forecast schedule process. The messages to these two processes from the SAR router initiates the generation of other messages. These generated messages are sent to the SAR router where they travel to the ISN to be routed to their various destinations.

Messages leaving the SPS processor enter the CCS via the ISN. These messages travel through the CCS in the path defined as path 3. As a result the path of all messages leaving the SPS is also defined as path 3. This path is independent of the possible routes travelled by messages through the SPS.

There are three possible routes for messages entering the SPS. Descriptions of these routes and messages are given below:

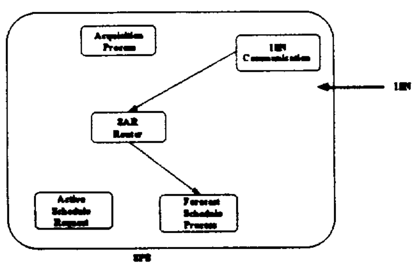


Route 1 ISN -> Acquisition Process



Route 2 ISN -> SAR Router -> Active Schedule Process

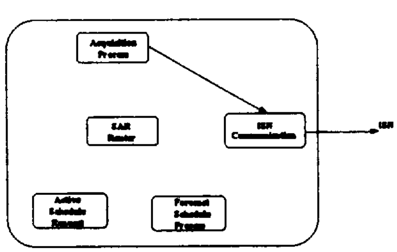
Messages for route 1 are: 0309, 0310, 0315, 0351, 0353, 0361, 0362, and 0365. .
Messages for route 2 are: 8651, 9910 and 9911.



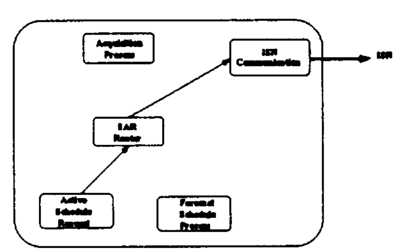
Route 3 ISN -> SAR Router -> Forecast Schedule Process

Message for this route is 9910.

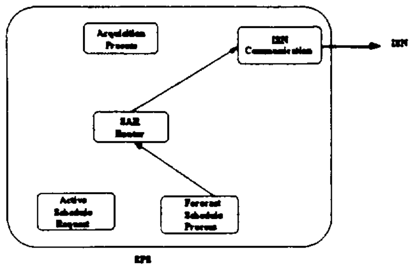
Three routes also exists for messages leaving the SPS. Diagrams of these routes are given below:



Route 4 Acquisition Process->ISN



Route 5 Active Schedule Request->SAR->ISN



Route 6 Forecast Schedule Process->SAR->ISN

5.3.2.3. ITS Message Paths

All messages to the ITS are destroyed at the sink. Display requests generated at the ITS are sent to the ISN where they are routed to their various destinations.

5.4. Message Dialogues

Ground Control Message Requests

TDRSS control includes the capability to modify certain parameters of an ongoing event or initiate special actions to support the event. Reconfigurations are initiated when an external processor sends the NCC a Ground Control Message Request (GCMR). This dialogue is summarized in table 5.1.

Table 5.1 Ground Control Message Dialogue

Message	**Response**
9803-9808 GCMR	03/02,03,04,06,07,11 - GCM
0360 - Acknowledgement	9801 - GCM Status
8604/9006 - NSS/NES	0362 - OPM Status
9802 - GCM Disposition	8654 - NSS

Schedule Dissemination

Schedules for use of the SN are transmitted to the users via type 94 messages. The schedule can be modified to reflect new specific schedule requests via Specific Schedule Request Messages. The NCC responds to a Schedule Specific Request with a Schedule Result Message. The dialogue for user scheduled messages are summarized in table 5.2.

Table 5.2 User Schedule Messages

Message	Response
94/01,02,03 - User Schedule	0360 - Acknowledgement
08/01,03,04,06 02xx - SHO	0360 - Acknowledgement
	0351 - SHO Status
86/01.02,03 - NSS	0360 - Acknowledgement
	8651 - Event Status
90/01/02/04/05 - NES	0360 - Acknowledgement

Performance Data Dissemination

The NCC receives system performance data for NCC operator review and dissemination in the form of Operations Data Messages (ODM) and Fault Isolation Monitoring System (FIMS) Reports. These messages are sent to the intended external processor as a routine requirement during mission support periods. The NCC forwards ODM performance data messages to external processors as requested. FIMS are also forwarded as required on a time available basis. This dialogue is summarized in table 5.3.

Table 5.3 Performance Data Dialogue

Message	Response
05, 06, 07 - ODM	0314 - Acknowledgement
8803 - FIMS	0314 - Acknowledgement
9204 - UPD Request	0314 - Acknowledgement
	9101 - UDP

Schedule Services

The schedule generation process produces a 7-day schedule based on spacecraft view predictions, TN equipment status, user requirements and /or specific support requests. This dialogue is summarized in table 5.4.

Table 5.4 Message Scheduling Dialogue

Message	Response

99/10/11 - Schedule Add/Delete	0314 - Acknowledgement
	9902 - Schedule Result
9901 - Deletion Notification	---

6. Simulation Program

The simulation program is an OPNET implemention of the abstracted model that uses discrete event simulation in a network of queues to represent the NCC and its external segments. The model is made up of fixed communication nodes connected by a virtual bus and point-to-point links. The virtual bus is a network bus implementation that does not allow collisions. This design decision was necessary to reduce the model runtime. Each node acts as a message generating and message processing server. The external nodes are made up of generator and sink processes. The internal nodes are single servers that service multiple subqueues within each node. These internal processors are represented by a combination of OPNET queues and processor models. The networks that connect the processors are represented by OPNET packet streams that are initiated and terminated by point-to-point transmitters and point-to-point receivers respectively. A complete set of model reports are provided in Appendix A.

6.1. Network Domain Description

The network domain is represented by the external subnet and the internal subnet. The external subnet is made up of one fixed communication node. This node contains packet generators, connected to a subnet that represent the NCC. The NCC subnet contains fixed communication nodes that represent the CCS, ITS, and SPS. These internal processors are connected by ISN which is represented by a virtual bus implementation that is collision free. Packets are moved from the external subnet to the internal subnet and back via two packet streams. The network level model is shown in Figure 6.1.

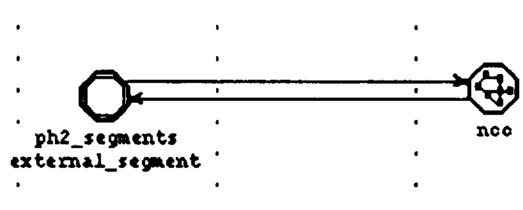


Figure 6.1: NCC Model - Network Level

A view of the NCC subnet is shown in Figure 6.2.

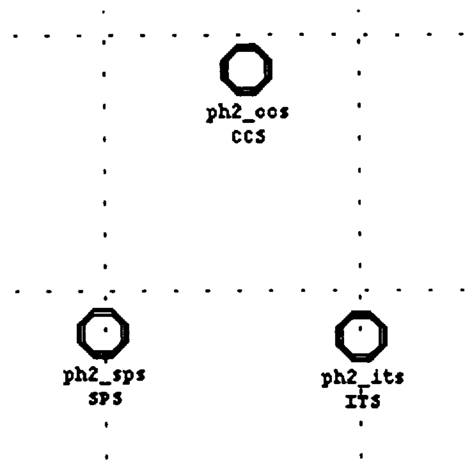


Figure 6.2: NCC Internal Subnet

6.2. Node Domain Description

The node model domain, like the network domain, contain node models that are associated with both the internal and external segments. The node models are the representation of server, sink, generation and communication elements. In the external segment node model, attributes are used to make each segment (node) unique. This is to facilitate creating one generic process model for all seven segments. The node model attributes will be read by using the OPNET kernel functions. The additional attributes are:

NAME	TYPE
number of messages	integer
station address	integer
message info file	data file

The "message info file" will contain information necessary for generating messages for a given distribution. The details of the file format will be explained in the process models discription.

6.2.1. External Subnet Node Models

Each external segment node is a combination of a message generator and a message sink. The message generator sends packets via a stream to the external segment queue. The function of this queue is to collect messages from the external segment message generators and pass them on to the CCS within the NCC. The segment queue is used only as a funnel for messages and is not an abstraction of any subsystem being modeled. The External Segment node models are shown in Figure 6.3.

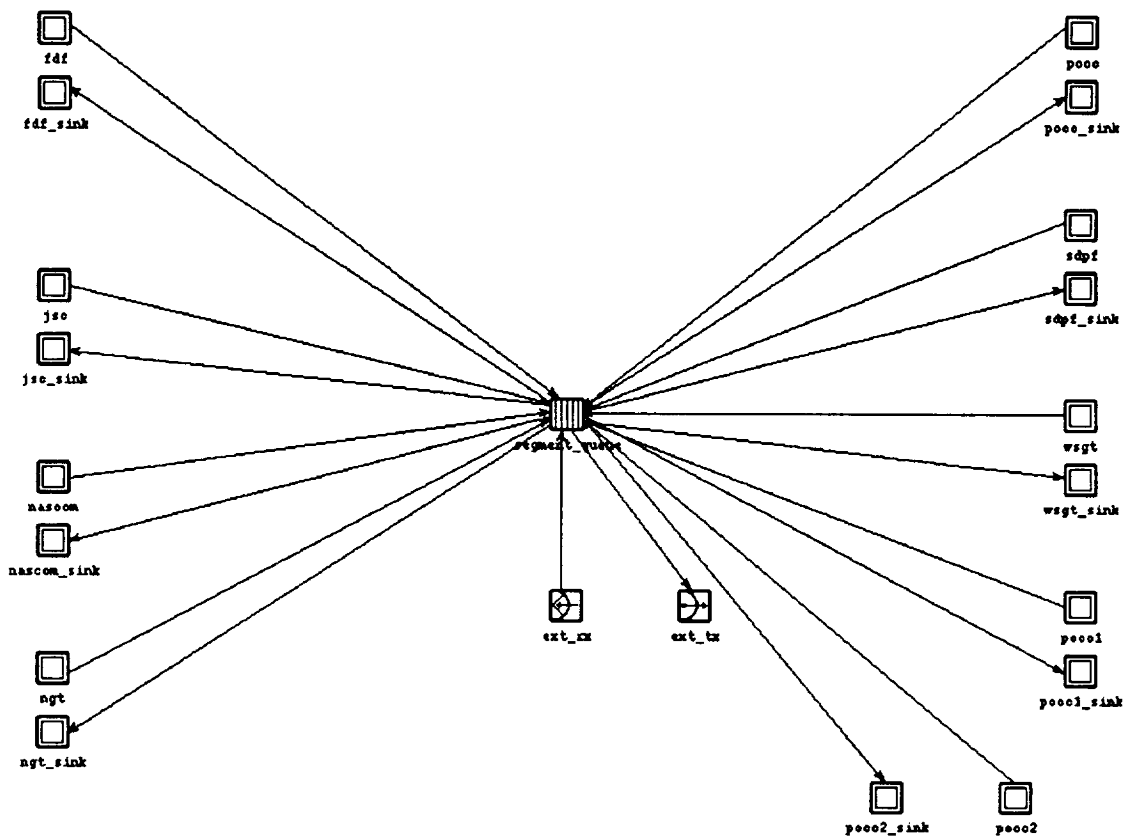


Figure 6.3: External Segment Node Models

6.2.2. Internal Subnet Node Models

Three primary node models make up the internal processors. These are the ph2_ccs, ph2_its, and ph2_sps. The ph2_ccs represent the CCS and is the interface to the external segments. Communication with the external segment objects is achieved through the use a transmitter, ext_int_tx, and a receiver, ext_int_rx. The at the heart of this object is a server process, server_router, in which all sub-processes are represented as sub-queues. Two generators are present in this node, and are used to generate messages for the event monitor and the network monitor. The ISN is the implementation of the virtual bus used in communication with the ITS and SPS. A sink is provided to destroy messages that exit the system. Figure 6.4 show the CCS node model. All communications between the elements of this model is by packet streams.

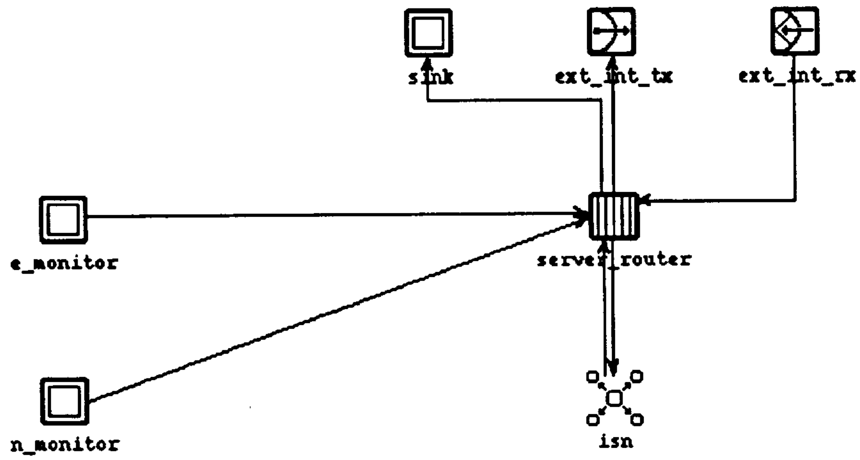
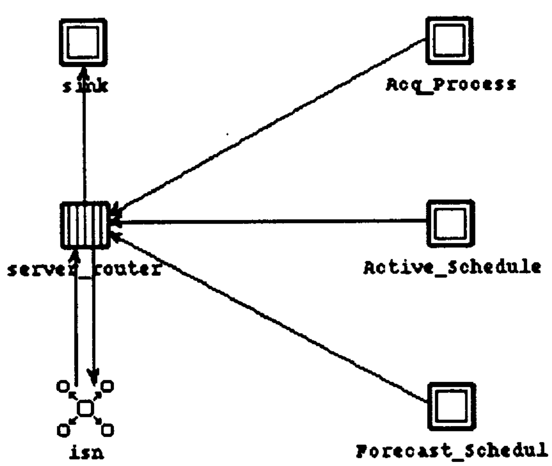


Figure 6.4: CCS Node Model

The SPS node model uses an architecture similar to that used by the CCS. In the SPS node model a server element that represents the CPU of the SPS is fed packets from an implementation of the Acquisition Process, the Active Scheduler and the Forecast Scheduler. Messages arriving from the ISN are also accepted. The messages entering the SPS are processed in accordance with the rules that govern the paths taken by specific messages through the SPS. These rules are defined in the section labelled SPS MESSAGE PATHS. Figure 6.5 show the elements that make up the SPS Node Model.



The ITS messages like the SPS and CCS messages are governed by the definitions given in the section named ITS MESSAGE PATHS. The ITS is a server based model that transmits display requests the Network Monitor located in the CCS.

6.2.2.1. Process Domain Description

The models on this domain are used to describe the behavior of the processor and queue models within the node model domain. The process models used are of two basic types, communicating processes, and server processes. The communicating processes are logical entities that interact to accomplish some common goal. The server processes are the entities at process the messages in an effort to impose the notion of work being done within the system being modeled. The logic of the process is specified in the FSM's. A FSM may be generally defined as an automaton which has states, inputs, and outputs. The FSM models its process by responding to changes in its inputs, modifying its state, and producing new outputs. In OPNET the primitives for building process models are states and transitions. OPNET employs the concept of event scheduling and interrupts, in which each incident is called a *simulation event* and an *interrupt* represents the actual execution of the scheduled event. See Appendix A for the process models used in this model.

7. Model Verification and Validation

This is the process of establishing that the computer implementation of the model is error-free and represents a correct implementation of the logical behavior of the conceptional model. During the verification of the model the following potential sources of errors were examined.

1. Numerical data errors -- distribution parameters, probability values, number of servers and inital values.
2. Unexpected random variate.
3. Inconsistency in units of measurements.
4. Entity flow problems.
5. Entity deadlocks.
6. Errors in statistical specification.

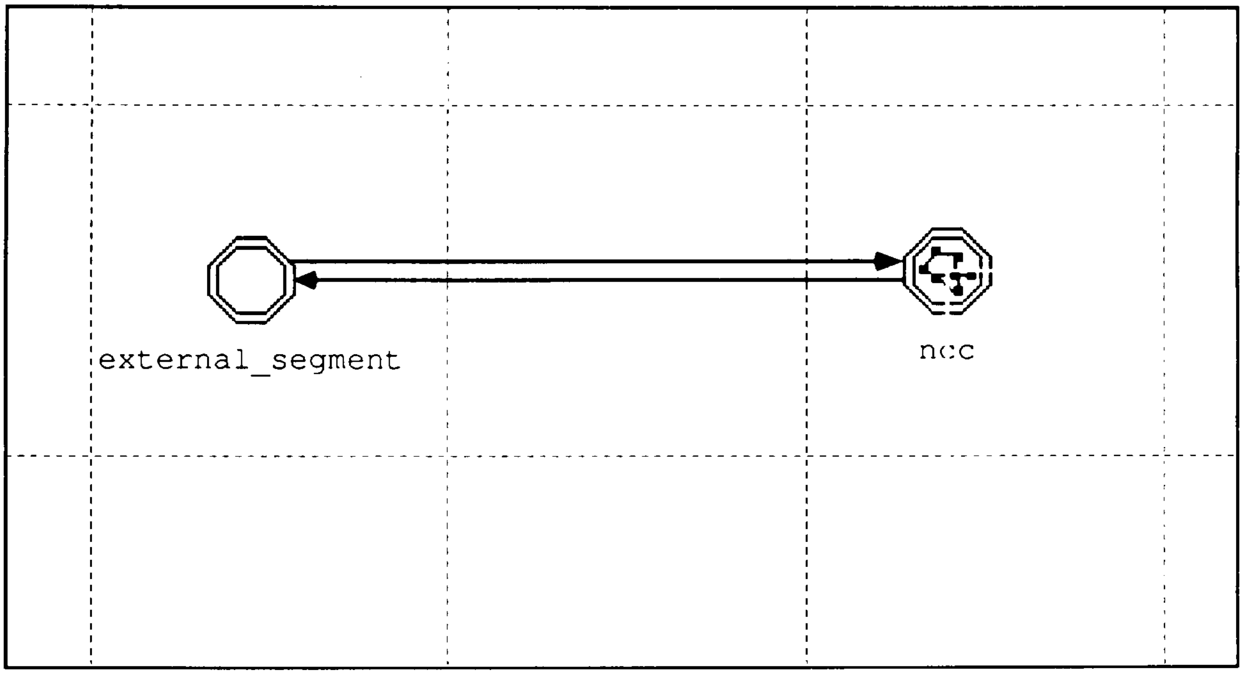
Verification was performed on a module by module basis. Numerical evaluations were performed on each module to ensure that all data logically represented a true abstraction of the system. Infrequent events were forced to verify correctness of operation. Event anamation was performed using the OPNET debugging interface. This was verified by performing a trace and tracking some critical variables.

Validation is the process of establishing that the model correctly represents the important aspects of the system being simulated. The most definitive test of the validity of the simulation model will be to establish that its output data closely resemble the output data expected from the actual system. However, the NCC is a highly secured system and output data from the actual system are not available for comparison, since this modeling is being performed at an unclassified level. It would therefore be impossible to validate the correctness of the model without compromising the security of the NCC.

For every service to be modeled a statistical profile must first be constructed for the messages that make up that service. This statistical profile will be used to identify a a statistical distribution and its parameters that closely match the behavior of this service. The distributions are then used in the appropiate message generators. The output from the model must be compared to the output from the real system. Differences resulting from a comparison can be minimized by varying the following model parameters: server processor speed, and message generatior distribution parameters.

8. Appendix A: Model Reports

NCC Network Level Model



...
...

fixed node **external segment**			
attribute	*value*	*type*	*default value*
name	external_segment	string	f
model	ph2_segments	typed file	ex_node
user id	0	integer	0
priority	0	integer	0
condition	enabled	toggle	enabled
x position	-70.2 (deg.)	double	0.0 (deg.)
y position	-22.5 (deg.)	double	0.0 (deg.)
altitude	0.0 (m)	double	0.0 (m)
threshold	0.0 (pixels/deg.)	double	0.0 (pixels/de...
icon name	fixed_comm	icon	fixed_comm
jsc.number of messages	2	integer	0
jsc.message info file 1	jsc_ncc1.gdf	string	
jsc.message info file 2	jsc_ncc1.gdf	string	
jsc.message info file 3	jsc_ncc1.gdf	string	
jsc.station_address	12	string	
nascom.number of messages	3	integer	0
nascom.message info file 1	nsc_ncc1.gdf	string	
nascom.message info file 2	nsc_ncc1.gdf	string	
nascom.message info file 3	nsc_ncc1.gdf	string	
nascom.station_address	13	string	
ngt.number of messages	5	integer	0
ngt.message info file 1	ngt_ncc1.gdf	string	
ngt.message info file 2	ngt_ncc1.gdf	string	
ngt.message info file 3	ngt_ncc1.gdf	string	
ngt.station_address	14	string	
pocc.number of messages	8	integer	0
pocc.message info file 1	poc_ncc1.gdf	string	
pocc.message info file 2	poc_ncc1.gdf	string	
pocc.message info file 3	poc_ncc1.gdf	string	
pocc.station_address	15	string	
sdpf.number of messages	1	integer	0
sdpf.message info file 1	sdp_ncc1.gdf	string	
sdpf.message info file 2	sdp_ncc1.gdf	string	
sdpf.message info file 3	sdp_ncc1.gdf	string	
sdpf.station_address	16	string	
wsgt.number of messages	11	integer	0
wsgt.message info file 1	wsg_ncc1.gdf	string	
wsgt.message info file 2	wsg_ncc1.gdf	string	
wsgt.message info file 3	wsg_ncc1.gdf	string	
wsgt.station_address	17	string	
fdf.number of messages	2	integer	0
fdf.message info file 1	fdf_ncc1.gdf	string	
fdf.message info file 2	fdf_ncc1.gdf	string	
fdf.message info file 3	fdf_ncc1.gdf	string	
fdf.station_address	11	string	

pt-to-pt simplex link **ls_3**			
attribute	*value*	*type*	*default value*
name	ls_3	string	ls

transmitter	external_segment.ex...	enumerated	
receiver	ncc.CCS.ext_int_rx	enumerated	
delay	0.0 (sec.)	double	0.0 (sec.)
ber	0.0 (err/bit)	double	0.0 (err/bit)
condition	enabled	toggle	enabled
user id	0	integer	0
cost	1.0	double	0.0
txdel model	dpt_txdel	typed file	dpt_txdel
propdel model	dpt_propdel	typed file	dpt_propdel
error model	dpt_error	typed file	dpt_error
ecc model	dpt_ecc	typed file	dpt_ecc
color	RGB030	color	RGB233

subnet **ncc**

attribute	*value*	*type*	*default value*
name	ncc	string	n
priority	0	integer	0
user id	1	integer	0
x position	17.55 (deg.)	double	0.0 (deg.)
y position	-21.3 (deg.)	double	0.0 (deg.)
x center	98.775 (deg.)	double	0.0 (deg.)
y center	-55.65 (deg.)	double	0.0 (deg.)
x span	162.45 (deg.)	double	0.0 (deg.)
y span	68.7 (deg.)	double	0.0 (deg.)
threshold	0.0 (pixels/deg.)	double	0.0 (pixels/de...
map	NONE	typed file	NONE
icon name	subnet	icon	subnet
outline color	RGB000	color	RGB133

pt-to-pt simplex link **ls_2**

attribute	*value*	*type*	*default value*
name	ls_2	string	ls
transmitter	ncc.CCS.ext_int_tx	enumerated	
receiver	external_segment.ex...	enumerated	
delay	0.0 (sec.)	double	0.0 (sec.)
ber	0.0 (err/bit)	double	0.0 (err/bit)
condition	enabled	toggle	enabled
user id	0	integer	0
cost	1.0	double	0.0
txdel model	dpt_txdel	typed file	dpt_txdel
propdel model	dpt_propdel	typed file	dpt_propdel
error model	dpt_error	typed file	dpt_error
ecc model	dpt_ecc	typed file	dpt_ecc
color	RGB300	color	RGB233

fixed node **ncc.ITS**

attribute	*value*	*type*	*default value*
name	ITS	string	f
model	ph2_its	typed file	ex_node

...
...

user id	0	integer	0
priority	0	integer	0
condition	enabled	toggle	enabled
x position	400.78125 (m)	double	0.0 (m)
y position	318.75 (m)	double	0.0 (m)
altitude	0.0 (m)	double	0.0 (m)
threshold	0.0 (pixels/m)	double	0.0 (pixels/m)
icon name	fixed_comm	icon	fixed_comm
Display_req.number of messages	0	integer	0
Display_req.message info file 1	its_dr1.gdf	string	
Display_req.message info file 2	its_dr1.gdf	string	
Display_req.message info file 3	its_dr1.gdf	string	
Display_req.station_address	4	string	

fixed node **ncc.SPS**

attribute	*value*	*type*	*default value*
name	SPS	string	f
model	ph2_sps	typed file	ex_node
user id	0	integer	0
priority	0	integer	0
condition	enabled	toggle	enabled
x position	299.21875 (m)	double	0.0 (m)
y position	317.1875 (m)	double	0.0 (m)
altitude	0.0 (m)	double	0.0 (m)
threshold	0.0 (pixels/m)	double	0.0 (pixels/m)
icon name	fixed_comm	icon	fixed_comm
Acq_Process.number of messages	2	integer	0
Acq_Process.message info file 1	sps_ap1.gdf	string	
Acq_Process.message info file 2	sps_ap1.gdf	string	
Acq_Process.message info file 3	sps_ap1.gdf	string	
Acq_Process.station_address	3	string	
...e_Schedule.number of messages	10	integer	0
..._Schedule.message info file 1	sps_as1.gdf	string	
..._Schedule.message info file 2	sps_as1.gdf	string	
..._Schedule.message info file 3	sps_as1.gdf	string	
Active_Schedule.station_address	3	string	
...st_Schedul.number of messages	10	integer	0
...t_Schedul.message info file 1	sps_fs1.gdf	string	
...t_Schedul.message info file 2	sps_fs1.gdf	string	
...t_Schedul.message info file 3	sps_fs1.gdf	string	
Forecast_Schedul.station_address	3	string	

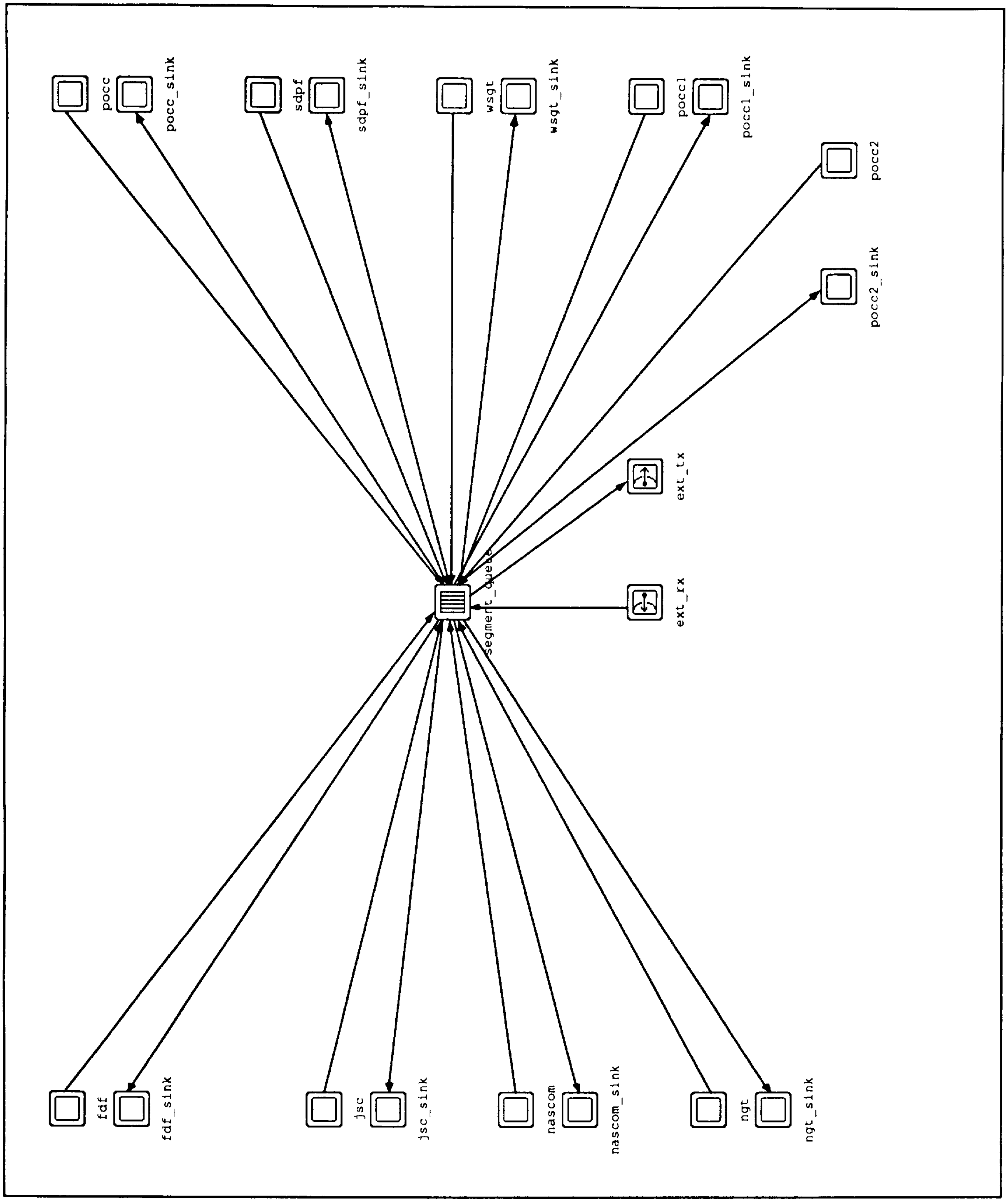
fixed node **ncc.CCS**

attribute	*value*	*type*	*default value*
name	CCS	string	f
model	ph2_ccs	typed file	ex_node
user id	0	integer	0
priority	0	integer	0

...
...

condition	enabled	toggle	enabled
x position	356.25 (m)	double	0.0 (m)
y position	214.0625 (m)	double	0.0 (m)
altitude	0.0 (m)	double	0.0 (m)
threshold	0.0 (pixels/m)	double	0.0 (pixels/m)
icon name	fixed_comm	icon	fixed_comm
e_monitor.number of messages	11	integer	0
e_monitor.message info file 1	em_ccs1.gdf	string	
e_monitor.message info file 2	em_ccs1.gdf	string	
e_monitor.message info file 3	em_ccs1.gdf	string	
e_monitor.station_address	2	string	
n_monitor.number of messages	7	integer	0
n_monitor.message info file 1	nm_ccs1.gdf	string	
n_monitor.message info file 2	nm_ccs1.gdf	string	
n_monitor.message info file 3	nm_ccs1.gdf	string	
n_monitor.station_address	2	string	

External Segment Node Model



...
...

processor **fdf_sink**			
attribute	*value*	*type*	*default value*
name	fdf_sink	string	p
process model	ph2_sink	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	disabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

processor **jsc**			
attribute	*value*	*type*	*default value*
name	jsc	string	p
process model	ph2_msg_gen	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

packet stream **jsc [0] -> segment_queue [1]**			
attribute	*value*	*type*	*default value*
name	strm_1	string	strm
src stream	0	enumerated	0
dest stream	1	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB030	color	RGB030

processor **jsc_sink**			
attribute	*value*	*type*	*default value*
name	jsc_sink	string	p
process model	ph2_sink	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	disabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

...
...

processor **nascom**			
attribute	*value*	*type*	*default value*
name	nascom	string	p
process model	ph2_msg_gen	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

packet stream **nascom [0] -> segment_queue [2]**			
attribute	*value*	*type*	*default value*
name	strm_3	string	strm
src stream	0	enumerated	0
dest stream	2	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB030	color	RGB030

processor **nascom_sink**			
attribute	*value*	*type*	*default value*
name	nascom_sink	string	p
process model	ph2_sink	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	disabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

processor **ngt**			
attribute	*value*	*type*	*default value*
name	ngt	string	p
process model	ph2_msg_gen	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

packet stream **ngt [0] -> segment queue [3]**			
attribute	*value*	*type*	*default value*
name	strm_5	string	strm
src stream	0	enumerated	0
dest stream	3	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB030	color	RGB030

processor **ngt sink**			
attribute	*value*	*type*	*default value*
name	ngt_sink	string	p
process model	ph2_ngt_sink	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

processor **pocc**			
attribute	*value*	*type*	*default value*
name	pocc	string	p
process model	ph2_msg_gen	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

packet stream **pocc [0] -> segment queue [4]**			
attribute	*value*	*type*	*default value*
name	strm_7	string	strm
src stream	0	enumerated	0
dest stream	4	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB030	color	RGB030

processor **pocc sink**			
attribute	*value*	*type*	*default value*
name	pocc_sink	string	p

...
...

process model	ph2_pocc_sink	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

processor **sdpf**

attribute	*value*	*type*	*default value*
name	sdpf	string	p
process model	ph2_msg_gen	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

packet stream **sdpf [0] -> segment_queue [5]**

attribute	*value*	*type*	*default value*
name	strm_9	string	strm
src stream	0	enumerated	0
dest stream	5	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB030	color	RGB030

processor **sdpf_sink**

attribute	*value*	*type*	*default value*
name	sdpf_sink	string	p
process model	ph2_sink	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	disabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

...
...

processor **wsgt**

attribute	*value*	*type*	*default value*
name	wsgt	string	p
process model	ph2_msg_gen	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

packet stream **wsgt [0] -> segment_queue [6]**

attribute	*value*	*type*	*default value*
name	strm_11	string	strm
src stream	0	enumerated	0
dest stream	6	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB030	color	RGB030

processor **wsgt_sink**

attribute	*value*	*type*	*default value*
name	wsgt_sink	string	p
process model	ph2_wsgt_sink	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

processor **fdf**

attribute	*value*	*type*	*default value*
name	fdf	string	p
process model	ph2_msg_gen	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

...
...

packet stream **fdf [0] -> segment_queue [0]**			
attribute	*value*	*type*	*default value*
name	strm_15	string	strm
src stream	0	enumerated	0
dest stream	0	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB030	color	RGB030

processor **pocc1**			
attribute	*value*	*type*	*default value*
name	pocc1	string	p
process model	ph2_msg_gen	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

packet stream **pocc1 [0] -> segment_queue [8]**			
attribute	*value*	*type*	*default value*
name	strm_16	string	strm
src stream	0	enumerated	0
dest stream	8	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB030	color	RGB030

processor **pocc1_sink**			
attribute	*value*	*type*	*default value*
name	pocc1_sink	string	p
process model	ph2_pocc_sink	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	disabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

processor **pocc2**			
attribute	*value*	*type*	*default value*
name	pocc2	string	p

...
...

process model	ph2_msg_gen	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	disabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

packet stream **pocc2 [0] -> segment_queue [9]**

attribute	*value*	*type*	*default value*
name	strm_17	string	strm
src stream	0	enumerated	0
dest stream	9	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB030	color	RGB030

processor **pocc2_sink**

attribute	*value*	*type*	*default value*
name	pocc2_sink	string	p
process model	ph2_pocc_sink	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	disabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

queue **segment_queue**

attribute	*value*	*type*	*default value*
name	segment_queue	string	p
process model	ph2_ext_svr	typed file	pc_fifo
subqueue count	2	integer	1
subqueue	(See below.)	object list	(See below.)
intrpt interval	-1.0 (sec.)	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	queue	icon	queue

...
...

subqueue **segment_queue.subqueue [0]**			
attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

subqueue **segment_queue.subqueue [1]**			
attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

packet stream **segment_queue [0] -> fdf_sink [0]**			
attribute	*value*	*type*	*default value*
name	strm_0	string	strm
src stream	0	enumerated	0
dest stream	0	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB300	color	RGB030

packet stream **segment_queue [1] -> jsc_sink [0]**			
attribute	*value*	*type*	*default value*
name	strm_2	string	strm
src stream	1	enumerated	0
dest stream	0	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB300	color	RGB030

packet stream **segment_queue [2] -> nascom_sink [0]**			
attribute	*value*	*type*	*default value*
name	strm_4	string	strm
src stream	2	enumerated	0
dest stream	0	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB300	color	RGB030

packet stream **segment_queue [3] -> ngt_sink [0]**			
attribute	*value*	*type*	*default value*
name	strm_6	string	strm
src stream	3	enumerated	0
dest stream	0	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB300	color	RGB030

packet stream **segment_queue [4] -> pocc_sink [0]**			
attribute	*value*	*type*	*default value*
name	strm_8	string	strm

...
...

src stream	4	enumerated	0
dest stream	0	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB300	color	RGB030

packet stream **segment_queue [5] -> sdpf_sink [0]**

attribute	*value*	*type*	*default value*
name	strm_10	string	strm
src stream	5	enumerated	0
dest stream	0	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB300	color	RGB030

packet stream **segment_queue [6] -> wsgt_sink [0]**

attribute	*value*	*type*	*default value*
name	strm_12	string	strm
src stream	6	enumerated	0
dest stream	0	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB300	color	RGB030

packet stream **segment_queue [7] -> ext_tx [0]**

attribute	*value*	*type*	*default value*
name	strm_13	string	strm
src stream	7	enumerated	0
dest stream	0	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB033	color	RGB030

packet stream **segment_queue [8] -> pocc1_sink [0]**

attribute	*value*	*type*	*default value*
name	strm_18	string	strm
src stream	8	enumerated	0
dest stream	0	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB300	color	RGB030

packet stream **segment_queue [9] -> pocc2_sink [0]**

attribute	*value*	*type*	*default value*
name	strm_19	string	strm
src stream	9	enumerated	0
dest stream	0	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB300	color	RGB030

...
...

pt-to-pt receiver **ext_rx**			
attribute	*value*	*type*	*default value*
name	ext_rx	string	pr
channel count	1	integer	1
channel	(See below.)	object list	(See below.)
ecc threshold	0.0 (err/bit)	double	0.0 (err/bit)
icon name	cable_receiver	icon	pt_rx

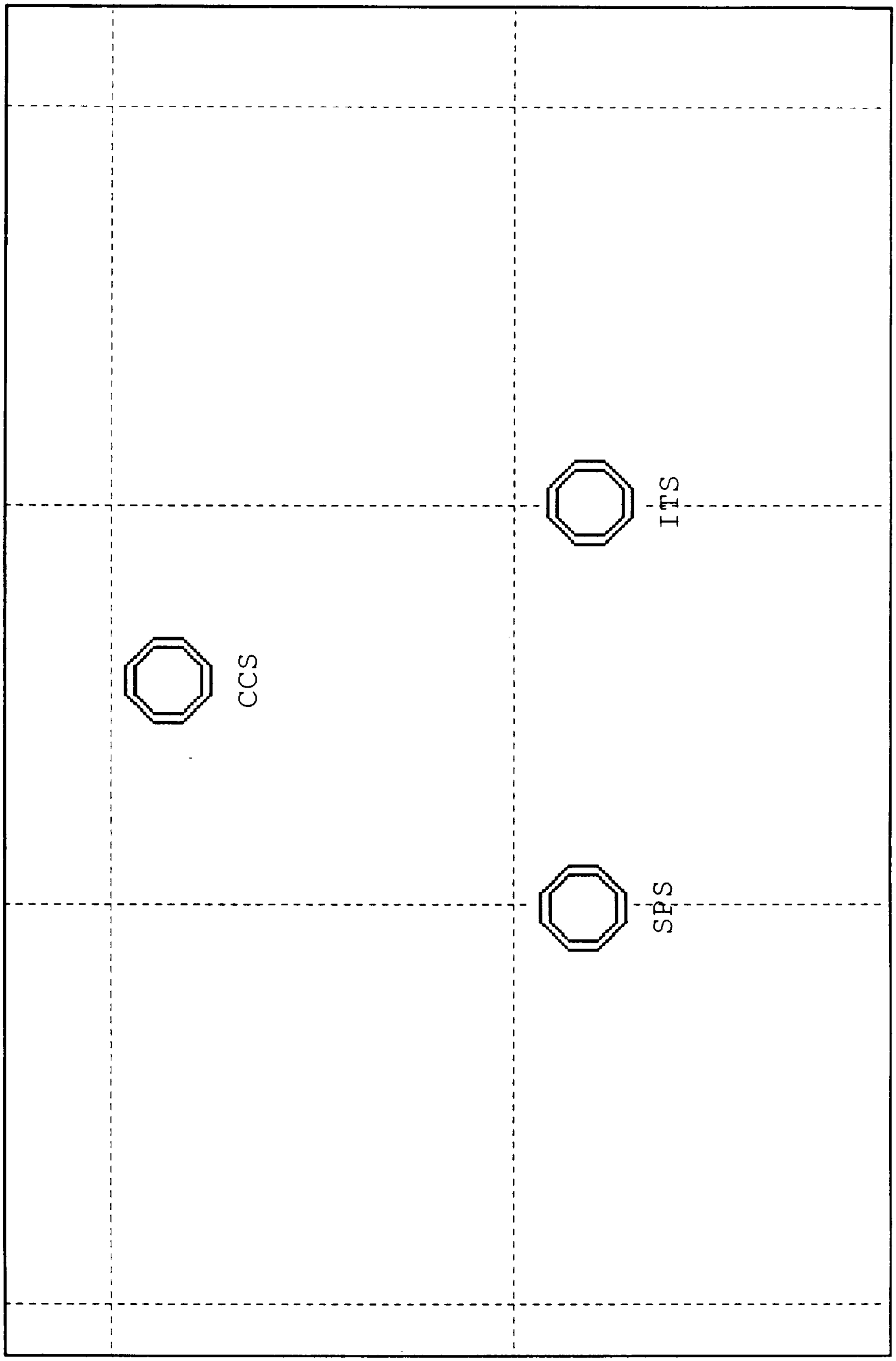
channel **ext_rx.channel [0]**			
attribute	*value*	*type*	*default value*
data rate	1,024 (bps)	double	1,024 (bps)

packet stream **ext_rx [0] -> segment_queue [7]**			
attribute	*value*	*type*	*default value*
name	strm_14	string	strm
src stream	0	enumerated	0
dest stream	7	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB333	color	RGB030

pt-to-pt transmitter **ext_tx**			
attribute	*value*	*type*	*default value*
name	ext_tx	string	pt
channel count	1	integer	1
channel	(See below.)	object list	(See below.)
icon name	cable_transmitter	icon	pt_tx

channel **ext_tx.channel [0]**			
attribute	*value*	*type*	*default value*
data rate	1,024 (bps)	double	1,024 (bps)

NCC Internal Segment Node Model



...
...

***processor* sink**

attribute	*value*	*type*	*default value*
name	sink	string	p
process model	ph2_enm_sink	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

***processor* e_monitor**

attribute	*value*	*type*	*default value*
name	e_monitor	string	p
process model	ph2_msg_gen	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

***packet stream* e_monitor [0] -> server_router [4]**

attribute	*value*	*type*	*default value*
name	strm_5	string	strm
src stream	0	enumerated	0
dest stream	4	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB300	color	RGB030

***processor* n_monitor**

attribute	*value*	*type*	*default value*
name	n_monitor	string	p
process model	ph2_msg_gen	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

...
...

packet stream **n monitor [0] -> server router [5]**			
attribute	*value*	*type*	*default value*
name	strm_6	string	strm
src stream	0	enumerated	0
dest stream	5	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB300	color	RGB030

queue **isn**			
attribute	*value*	*type*	*default value*
name	isn	string	p
process model	ph2_isn	typed file	pc_fifo
subqueue count	2	integer	1
subqueue	(See below.)	object list	(See below.)
intrpt interval	-1.0 (sec.)	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	module_connect	icon	queue
station address	2	integer	2

subqueue **isn.subqueue [0]**			
attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

subqueue **isn.subqueue [1]**			
attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

packet stream **isn [1] -> server router [1]**			
attribute	*value*	*type*	*default value*
name	strm_0	string	strm
src stream	1	enumerated	0
dest stream	1	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB300	color	RGB030

queue **server router**			
attribute	*value*	*type*	*default value*
name	server_router	string	p

...
...

process model	ph2_ccs_svr	typed file	pc_fifo
subqueue count	12	integer	1
subqueue	(See below.)	object list	(See below.)
intrpt interval	-1.0 (sec.)	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	queue	icon	queue
service rate	67,200 (bits/sec)	double	0.0 (bits/sec)

subqueue **server router.subqueue [0]**

attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

subqueue **server router.subqueue [1]**

attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

subqueue **server router.subqueue [2]**

attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

subqueue **server router.subqueue [3]**

attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

subqueue **server router.subqueue [4]**

attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

subqueue **server router.subqueue [5]**

attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

subqueue **server router.subqueue [6]**

attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

...
...

subqueue **server router.subqueue [7]**			
attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

subqueue **server router.subqueue [8]**			
attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

subqueue **server router.subqueue [9]**			
attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

packet stream **server router [1] -> isn [1]**			
attribute	*value*	*type*	*default value*
name	strm_1	string	strm
src stream	1	enumerated	0
dest stream	1	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB030	color	RGB030

packet stream **server router [2] -> ext int tx [0]**			
attribute	*value*	*type*	*default value*
name	strm_2	string	strm
src stream	2	enumerated	0
dest stream	0	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB300	color	RGB030

packet stream **server router [0] -> sink [0]**			
attribute	*value*	*type*	*default value*
name	strm_4	string	strm
src stream	0	enumerated	0
dest stream	0	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB300	color	RGB030

pt-to-pt receiver **ext int rx**			
attribute	*value*	*type*	*default value*
name	ext_int_rx	string	pr
channel count	1	integer	1
channel	(See below.)	object list	(See below.)

...
...

ecc threshold	0.0 (err/bit)	double	0.0 (err/bit)
icon name	cable_receiver	icon	pt_rx

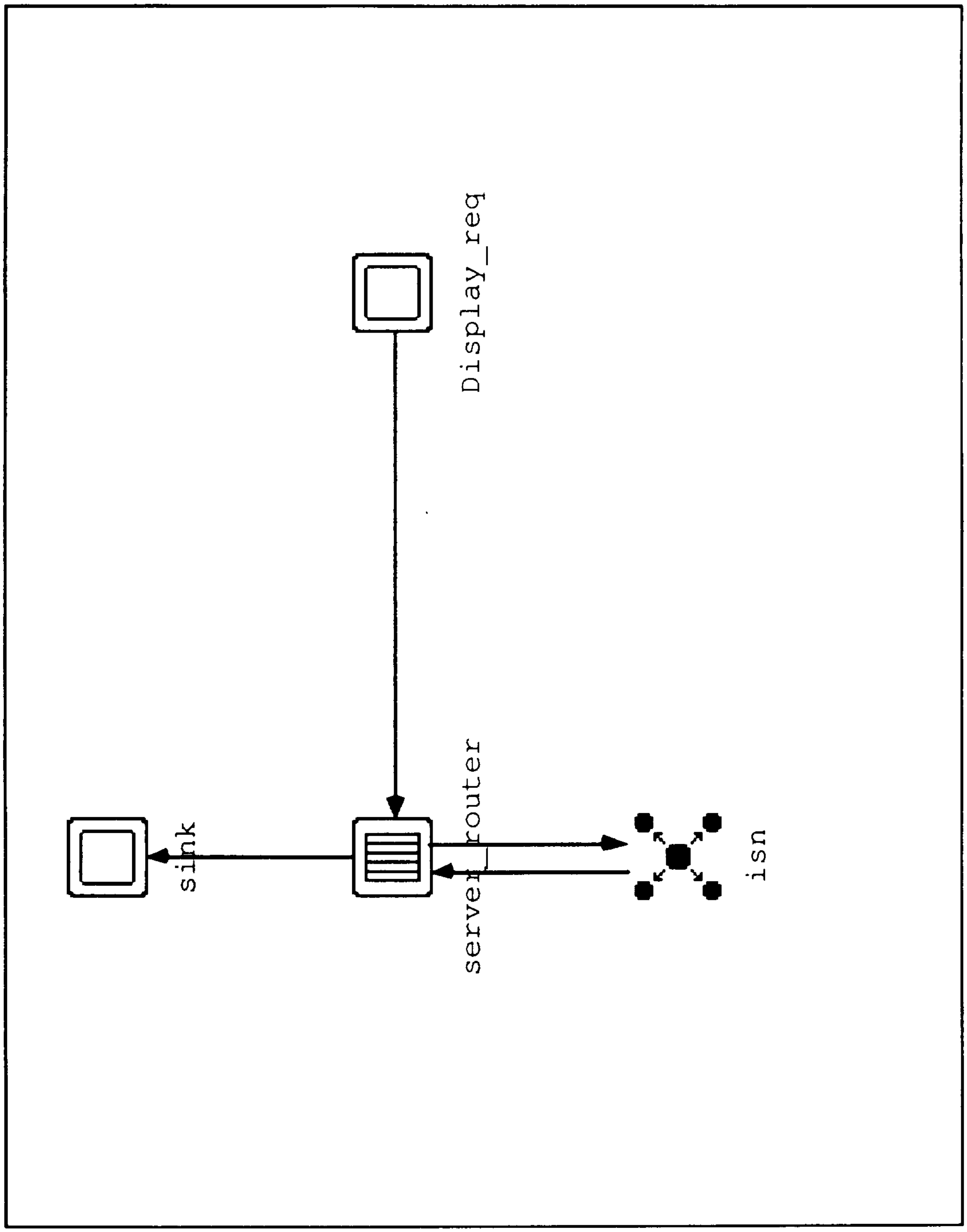
channel **ext_int_rx.channel [0]**			
attribute	*value*	*type*	*default value*
data rate	1,024 (bps)	double	1,024 (bps)

packet stream **ext_int_rx [0] -> server_router [0]**			
attribute	*value*	*type*	*default value*
name	strm_3	string	strm
src stream	0	enumerated	0
dest stream	0	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB033	color	RGB030

pt-to-pt transmitter **ext_int_tx**			
attribute	*value*	*type*	*default value*
name	ext_int_tx	string	pt
channel count	1	integer	1
channel	(See below.)	object list	(See below.)
icon name	cable_transmitter	icon	pt_tx

channel **ext_int_tx.channel [0]**			
attribute	*value*	*type*	*default value*
data rate	1,024 (bps)	double	1,024 (bps)

ITS Node Model



...

...

processor **sink**

attribute	value	type	default value
name	sink	string	p
process model	ph2_sink	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	disabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

processor **Display_req**

attribute	value	type	default value
name	Display_req	string	p
process model	ph2_packet_gen	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

packet stream **Display_req [0] -> server_router [2]**

attribute	value	type	default value
name	strm_3	string	strm
src stream	0	enumerated	0
dest stream	2	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB030	color	RGB030

queue **isn**

attribute	value	type	default value
name	isn	string	p
process model	ph2_isn	typed file	pc_fifo
subqueue count	2	integer	1
subqueue	(See below.)	object list	(See below.)
intrpt interval	-1.0 (sec.)	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled

icon name	module_connect	icon	queue
station address	4	integer	4

subqueue **isn.subqueue [0]**			
attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

subqueue **isn.subqueue [1]**			
attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

packet stream **isn [1] -> server router [1]**			
attribute	*value*	*type*	*default value*
name	strm_1	string	strm
src stream	1	enumerated	0
dest stream	1	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB300	color	RGB030

queue **server router**			
attribute	*value*	*type*	*default value*
name	server_router	string	p
process model	ph2_its_svr	typed file	pc_fifo
subqueue count	3	integer	1
subqueue	(See below.)	object list	(See below.)
intrpt interval	-1.0 (sec.)	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	queue	icon	queue
service rate	57,600 (bits/sec)	double	0.0 (bits/sec)

subqueue **server router.subqueue [0]**			
attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

subqueue **server router.subqueue [1]**			
attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

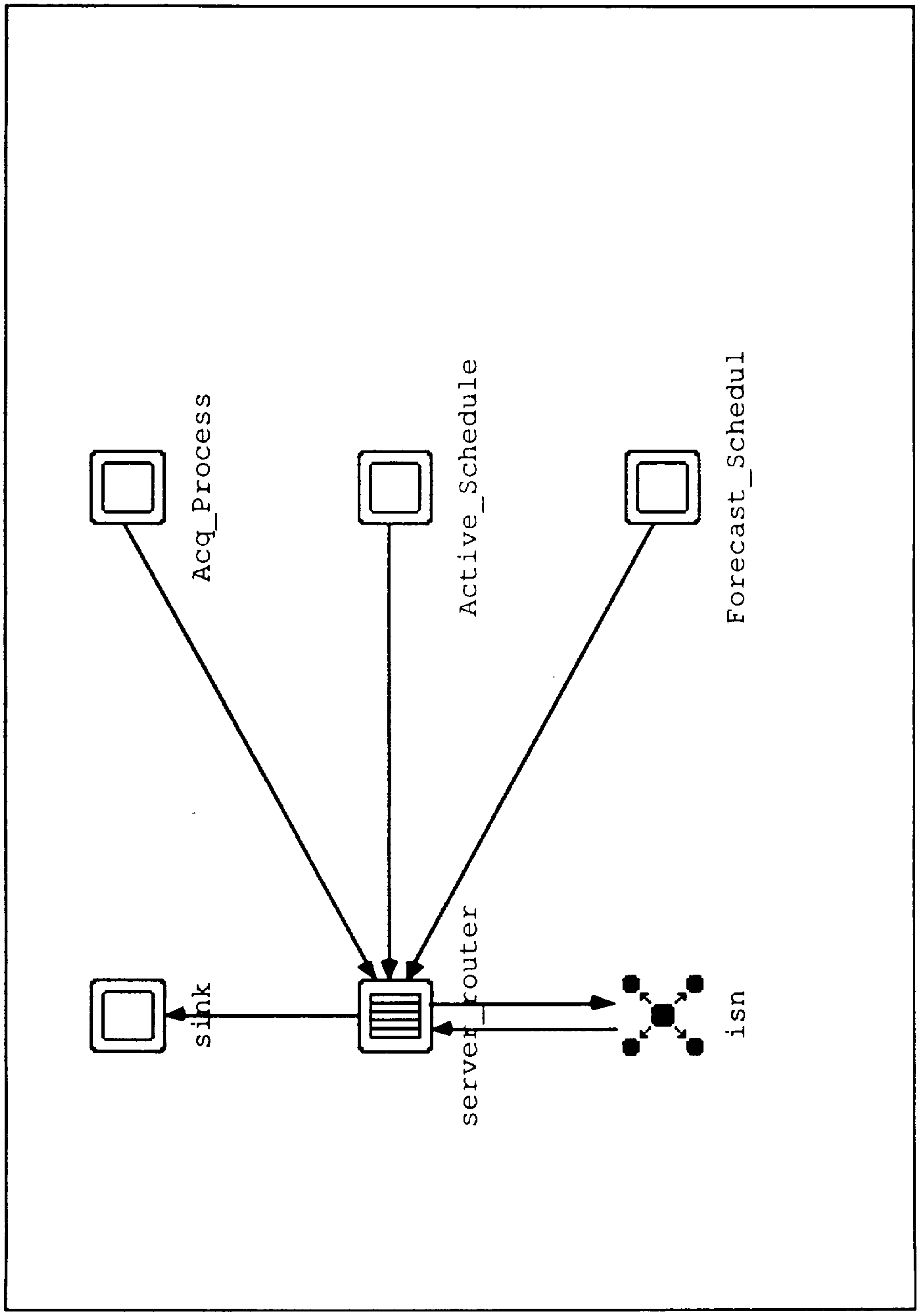
...
...

subqueue **server router.subqueue [2]**			
attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

packet stream **server router [1] -> isn [1]**			
attribute	*value*	*type*	*default value*
name	strm_0	string	strm
src stream	1	enumerated	0
dest stream	1	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB030	color	RGB030

packet stream **server router [0] -> sink [0]**			
attribute	*value*	*type*	*default value*
name	strm_2	string	strm
src stream	0	enumerated	0
dest stream	0	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB030	color	RGB030

SPS Node Model



...
...

processor **sink**

attribute	*value*	*type*	*default value*
name	sink	string	p
process model	ph2_sps_sink	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

processor **Acq Process**

attribute	*value*	*type*	*default value*
name	Acq_Process	string	p
process model	ph2_msg_gen	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

packet stream **Acq Process [0] -> server router [2]**

attribute	*value*	*type*	*default value*
name	strm_2	string	strm
src stream	0	enumerated	0
dest stream	2	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB033	color	RGB030

processor **Active Schedule**

attribute	*value*	*type*	*default value*
name	Active_Schedule	string	p
process model	ph2_msg_gen	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

...
...

packet stream **Active Schedule [0] -> server router [3]**			
attribute	*value*	*type*	*default value*
name	strm_3	string	strm
src stream	0	enumerated	0
dest stream	3	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB033	color	RGB030

processor **Forecast Schedul**			
attribute	*value*	*type*	*default value*
name	Forecast_Schedul	string	p
process model	ph2_msg_gen	typed file	sink
intrpt interval	disabled	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	processor	icon	processor

packet stream **Forecast Schedul [0] -> server router [4]**			
attribute	*value*	*type*	*default value*
name	strm_4	string	strm
src stream	0	enumerated	0
dest stream	4	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB033	color	RGB030

queue **isn**			
attribute	*value*	*type*	*default value*
name	isn	string	p
process model	ph2_isn	typed file	pc_fifo
subqueue count	2	integer	1
subqueue	(See below.)	object list	(See below.)
intrpt interval	-1.0 (sec.)	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	module_connect	icon	queue
station address	3	integer	3

subqueue **isn.subqueue [0]**			
attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

subqueue **isn.subqueue [1]**			
attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

packet stream **isn [1] -> server_router [1]**			
attribute	*value*	*type*	*default value*
name	strm_1	string	strm
src stream	1	enumerated	0
dest stream	1	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB300	color	RGB030

queue **server_router**			
attribute	*value*	*type*	*default value*
name	server_router	string	p
process model	ph2_sps_svr	typed file	pc_fifo
subqueue count	12	integer	1
subqueue	(See below.)	object list	(See below.)
intrpt interval	-1.0 (sec.)	toggle double	disabled
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
recovery intrpts	disabled	enumerated	disabled
priority	0	integer	0
super priority	disabled	toggle	disabled
icon name	queue	icon	queue
service_rate	57,600 (bits/sec)	double	0.0 (bits/sec)

subqueue **server_router.subqueue [0]**			
attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

subqueue **server_router.subqueue [1]**			
attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

subqueue **server_router.subqueue [2]**			
attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)

...
...

pk capacity	infinite (pks)	double	infinite (pks)

subqueue **server_router.subqueue [3]**

attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

subqueue **server_router.subqueue [4]**

attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

subqueue **server_router.subqueue [5]**

attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

subqueue **server_router.subqueue [6]**

attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

subqueue **server_router.subqueue [7]**

attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

subqueue **server_router.subqueue [8]**

attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

subqueue **server_router.subqueue [9]**

attribute	*value*	*type*	*default value*
bit capacity	infinite (bits)	double	infinite (bits)
pk capacity	infinite (pks)	double	infinite (pks)

packet stream **server_router [1] -> isn [1]**

attribute	*value*	*type*	*default value*
name	strm_0	string	strm
src stream	1	enumerated	0
dest stream	1	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB030	color	RGB030

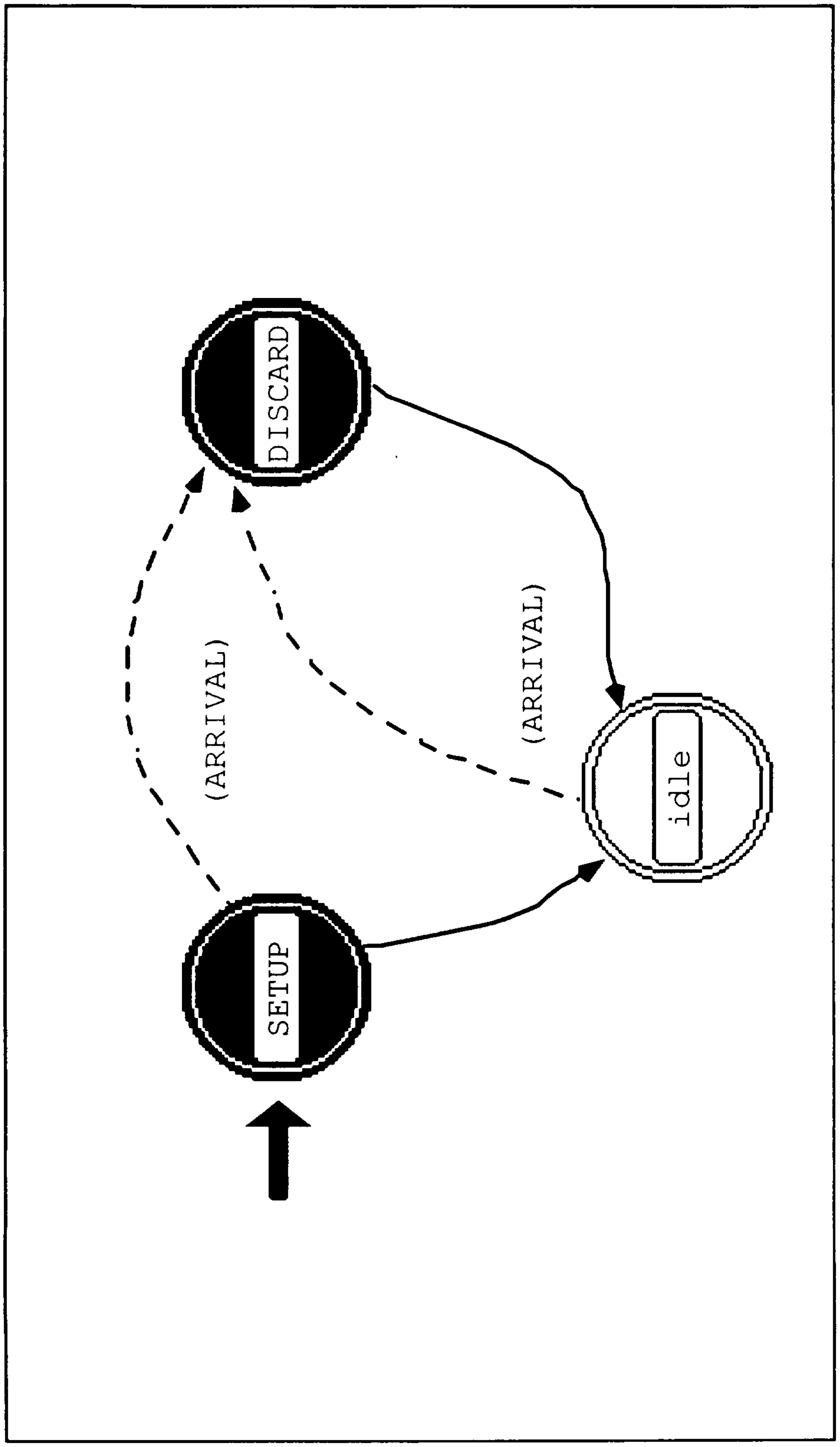
packet stream **server_router [0] -> sink [0]**

attribute	*value*	*type*	*default value*
name	strm_5	string	strm

...
...

src stream	0	enumerated	0
dest stream	0	enumerated	0
intrpt method	scheduled	integer	scheduled
delay	0.0 (sec.)	double	0.0 (sec.)
color	RGB030	color	RGB030

SPS Sink



...
...

Header Block

```
#define ARRIVAL     op_intrpt_type() == OPC_INTRPT_STRM
```

State Variable Block

```
Objid    \node;
Objid    \acq_proc;
Objid    \active_proc;
Objid    \forecast_proc;
Objid    \subnet;
```

Temporary Variable Block

```
Packet*    pkptr;
int        message_type;
int        code;
```

forced state SETUP

attribute	*value*	*type*	*default value*
name	SETUP	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs SETUP

```
/* get the object id of the network and event monitors */
subnet = op_id_from_name(0, OPC_OBJTYPE_SUBNET, "ncc");
node = op_id_from_name(subnet, OPC_OBJTYPE_NODE_FIXED, "SPS");
acq_proc = op_id_from_name(node,OPC_OBJTYPE_PROC, "Acq_Process");
active_proc = op_id_from_name(node,OPC_OBJTYPE_PROC, "Active_Schedule");
forecast_proc = op_id_from_name(node,OPC_OBJTYPE_PROC, "Forecast_Schedul");
```

transition SETUP -> DISCARD

attribute	*value*	*type*	*default value*
name	tr_1	string	tr
condition	ARRIVAL	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

transition SETUP -> idle

attribute	*value*	*type*	*default value*
name	tr_2	string	tr
condition		string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

...
...

forced state **DISCARD**			
attribute	*value*	*type*	*default value*
name	DISCARD	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **DISCARD**

```

/* get packet */

pkptr = op_pk_get (op_intrpt_strm());

/* print packet contents */
op_pk_print(pkptr);

op_pk_nfd_get(pkptr,"type_class",&message_type);

switch (message_type)
{

	case 9910:
		{	code = 9901;
			op_intrpt_schedule_remote(op_sim_time(),code,forecast_proc);
		}
	break;
	case 9911:
		{	code = 9902;
			op_intrpt_schedule_remote(op_sim_time(),code,active_proc);
	}
	break;
	case 363:
		{	code = 314;
			op_intrpt_schedule_remote(op_sim_time(),code,acq_proc);
		}
	break;
	default: ;
} /* end switch */


/* destroy */
op_pk_destroy(pkptr);
```

transition **DISCARD -> idle**			
attribute	*value*	*type*	*default value*
name	tr_3	string	tr
condition		string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

...
...

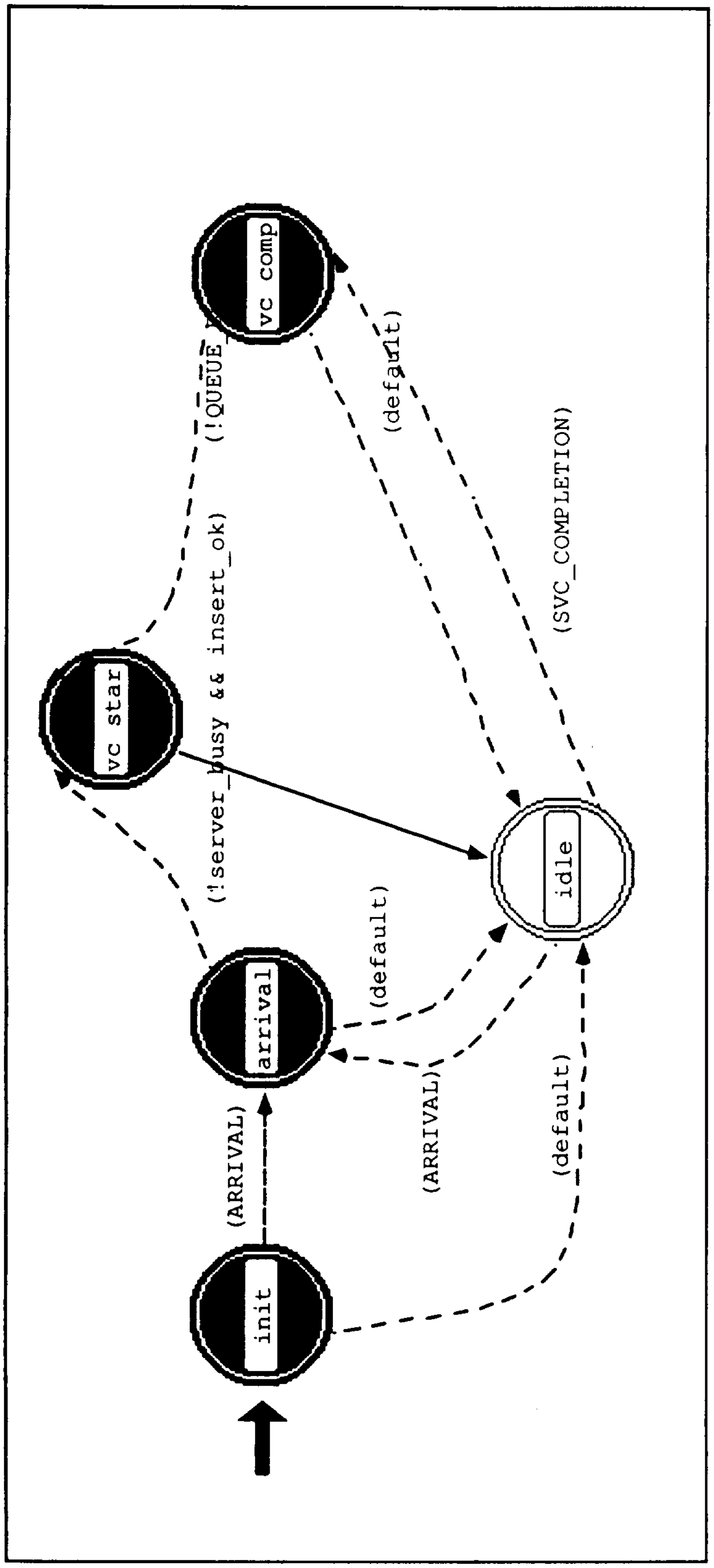
unforced state **idle**

attribute	*value*	*type*	*default value*
name	idle	string	st
enter execs	(empty)	textlist	(empty)
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

transition **idle -> DISCARD**

attribute	*value*	*type*	*default value*
name	tr_4	string	tr
condition	ARRIVAL	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

SPS Server


init
arrival
vc star
vc comp
idle
(ARRIVAL)
(default)
(ARRIVAL)
(default)
(!server_busy && insert_ok)
(!QUEUE_
(default)
(SVC_COMPLETION)

...
...

Process Model Attributes

attribute	*value*	*type*	*default value*
service_rate	promoted	double	9,600 (bits/sec)

Header Block

```
#define QUEUE_EMPTY              (op_q_empty ())
#define SVC_COMPLETION           op_intrpt_type () == OPC_INTRPT_SELF
#define ARRIVAL                  op_intrpt_type () == OPC_INTRPT_STRM

#define SERVER_ROUTER_OUT_STREAM   1
#define MAC_SOURCE_IN_STREAM     2
#define MAC_ROUTER_IN_STREAM     1
#define ROUTER_SINK_OUT_STREAM   0
#define ROUTER_EXT_OUT_STREAM    2

/* CCS Stream indexes */
#define SPS_SINK                 0
#define ISN_IN                   1
#define ISN_OUT                  1
#define ACQ_OUT                  2
#define ACTIVE_OUT               3
#define FORECAST_OUT             4

/* CCS SUB Queues */
#define Q_ISN_IN                 1
#define Q_ISN_OUT                2
#define Q_ACQ_IN                 3
#define Q_ACQ_OUT                4
#define Q_SAR_IN                 5
#define Q_SAR_OUT                6
#define Q_ACTIVE_IN              7
#define Q_ACTIVE_OUT             8
#define Q_FORECAST_IN            9
#define Q_FORECAST_OUT           10
```

State Variable Block

```
int          \server_busy;
Objid        \own_id;
Objid        \mac_objid;
Objid        \source_objid;

int          \station_addr;
int          \dest_addr;
int          \source_addr;
double       \service_rate;
```

Temporary Variable Block

```
Packet*          pkptr;
Packet*          cp_pkptr;

int              message_type;
int              message_path;
double           service_time;
```

...
...

```
double    pk_svc_time;
int       insert_ok;
int       stream_index;
int       queue_index;
int       temp,temp1;
int       int_ext;
int       pk_len;
int       Q;
```

forced state **init**			
attribute	*value*	*type*	*default value*
name	init	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **init**

```
/* initially the server is idle */
server_busy = 0;

/* get queue module's own object id */
own_id = op_id_self ();

op_ima_obj_attr_get(own_id, "service_rate", &service_rate);

/* determine id of the mac object */
mac_objid=op_topo_out_assoc(own_id,ISN_OUT);

/* get the station address from the mac */
op_ima_obj_attr_get(mac_objid,"station_address",&station_addr);
```

transition **init -> arrival**			
attribute	*value*	*type*	*default value*
name	tr_1	string	tr
condition	ARRIVAL	string	
executive		string	
color	RGB300	color	RGB333
drawing style	line	toggle	spline

transition **init -> idle**			
attribute	*value*	*type*	*default value*
name	tr_2	string	tr
condition	default	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

...
...

forced state **arrival**			
attribute	*value*	*type*	*default value*
name	arrival	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **arrival**

```
/* Determine the stream source of the interrupt */
stream_index = op_intrpt_strm();


/* acquire the arriving packet */
/* multiple arriving streams are supported. */
pkptr = op_pk_get (stream_index);

/* POSSIBLE STREAMS ARE 1-4 */

switch(stream_index)
{
      case 1:    /* packet is from outside SPS */
                 {

                       /* attempt to enqueue the packet at tail of subqueue ISN  */
                 /*------------- queue insertion machenics------------------------- */
                 if (op_subq_pk_insert (Q_ISN_IN, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
                       {
                       /* the inserton failed (due to to a full queue) */
                       /* deallocate the packet */
                       op_pk_destroy (pkptr);

                       /* set flag indicating insertion fail */
                       /* this flag is used to determine transition */
                       /* out of this state */
                       printf("[Queue --> %d is full]\n",Q_ISN_IN);
                       insert_ok = 0;
                       }
                 else{
                       /* insertion was successful */
                       insert_ok = 1;
                       }
                 /* ---------------------------------------------------------------*/
          break;
                 } /* end case 1 */
      case 2:    /* packet is from Acquisition Process */
                 {

                       /* attempt to enqueue the packet at tail of subqueue ACQ_OUT */
                 /*------------- queue insertion machenics------------------------- */
                 if (op_subq_pk_insert (Q_ACQ_IN, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
                       {
                       /* the inserton failed (due to to a full queue) */
                       /* deallocate the packet */
                       op_pk_destroy (pkptr);

                       /* set flag indicating insertion fail */
                       /* this flag is used to determine transition */
```

```
                /* out of this state */
                printf("[Queue --> %d is full]\n",Q_ACQ_IN);
                insert_ok = 0;
                }
            else{
                /* insertion was successful */
                insert_ok = 1;
                }
            /* -----------------------------------------------------------*/
        break;
            } /* end case 2 */

    case 3: /* event from stream active scheduler*/
            {
            /*------------- queue insertion machenics------------------------- */
            if (op_subq_pk_insert (Q_ACTIVE_OUT, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
                {
                /* the inserton failed (due to to a full queue) */
                /* deallocate the packet */
                op_pk_destroy (pkptr);

                /* set flag indicating insertion fail */
                /* this flag is used to determine transition */
                /* out of this state */
                printf("[Queue --> %d is full]\n",Q_ACTIVE_OUT);
                insert_ok = 0;
                }
            else{
                /* insertion was successful */
                insert_ok = 1;
                }
            /* -----------------------------------------------------------*/

            break;
            } /* end case 3 */
    case 4: /* event from stream FORECAST scheduler*/
            {
            /*------------- queue insertion machenics------------------------- */
            if (op_subq_pk_insert (Q_FORECAST_OUT, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
                {
                /* the inserton failed (due to to a full queue) */
                /* deallocate the packet */
                op_pk_destroy (pkptr);

                /* set flag indicating insertion fail */
                /* this flag is used to determine transition */
                /* out of this state */
                printf("[Queue --> %d is full]\n",Q_FORECAST_OUT);
                insert_ok = 0;
                }
            else{
                /* insertion was successful */
                insert_ok = 1;
                }
            /* -----------------------------------------------------------*/

            break;
            } /* end case 4 */
```

...
...

```
}    /* end of enqueue switch on stream index */
```

transition **arrival -> svc_start**			
attribute	*value*	*type*	*default value*
name	tr_3	string	tr
condition	!server_busy && ins...	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

transition **arrival -> idle**			
attribute	*value*	*type*	*default value*
name	tr_4	string	tr
condition	default	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

unforced state **idle**			
attribute	*value*	*type*	*default value*
name	idle	string	st
enter execs	(empty)	textlist	(empty)
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

transition **idle -> arrival**			
attribute	*value*	*type*	*default value*
name	tr_5	string	tr
condition	ARRIVAL	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

transition **idle -> svc_compl**			
attribute	*value*	*type*	*default value*
name	tr_6	string	tr
condition	SVC_COMPLETION	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

forced state **svc_start**			
attribute	*value*	*type*	*default value*
name	svc_start	string	st
enter execs	(See below.)	textlist	(See below.)

exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **svc_start**

```
/* find the queue with the most packets */
queue_index = op_subq_index_map(OPC_QSEL_MAX_PKSIZE);


/* get a handle on packet at head of subqueue with most packets */
/* (this does not remove the packet) */
pkptr = op_subq_pk_access (queue_index, OPC_QPOS_HEAD);


/* get the service time from the packet */
op_pk_nfd_get(pkptr,"service_time",&service_time);

    /* determine the time required to complete */
    /* service of the packet */
    pk_svc_time = service_time;

/* schedule an interrupt for this process */
/* at the time where service ends. */
op_intrpt_schedule_self (op_sim_time () + pk_svc_time, queue_index);

/*------------ the server is now busy.-------------- */
server_busy = 1;
```

transition **svc_start -> idle**

attribute	*value*	*type*	*default value*
name	tr_7	string	tr
condition		string	
executive		string	
color	RGB300	color	RGB333
drawing style	line	toggle	spline

forced state **svc_compl**

attribute	*value*	*type*	*default value*
name	svc_compl	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **svc_compl**

```
/* what is the interrupt code */
queue_index = op_intrpt_code();

/* extract packet at head of queue */
/* this is the packet just finishing service */
pkptr = op_subq_pk_remove (queue_index, OPC_QPOS_HEAD);
```

...
...

```
/* acquire the packet parameters for processing */
op_pk_nfd_get(pkptr,"type_class",&message_type);
op_pk_nfd_get(pkptr,"message_path",&message_path);

/* pass packet to next process */
switch(queue_index)
{
    case 1: { /* next queue is ACQ OR SAR ROUTER */
            if (message_type == 8651 || message_type == 9910 || message_type == 9911)
                Q = Q_SAR_IN;
            else Q = Q_ACQ_IN;

    /*------------- queue insertion machenics------------------------ */
            if (op_subq_pk_insert (Q, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
                {
                /* the inserton failed (due to to a full queue) */
                /* deallocate the packet */
                op_pk_destroy (pkptr);

                /* set flag indicating insertion fail */
                /* this flag is used to determine transition */
                /* out of this state */
                printf("[Queue --> %d is full]\n",Q);
                insert_ok = 0;
                }
            else{
                /* insertion was successful */
                insert_ok = 1;
                }
            /* ------------------------------------------------------------*/
            break;
            }  /* end case 1 */
    case 2:   /* send packet to ISN */
            {
            op_pk_send(pkptr,ISN_OUT);
            break;
            } /* end case 2 */
    case 4: /* enqueue in ISN_OUT */
            {
            /*------------- queue insertion machenics------------------------ */
            if (op_subq_pk_insert (Q_ISN_OUT, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
                {
                /* the inserton failed (due to to a full queue) */
                /* deallocate the packet */
                op_pk_destroy (pkptr);

                /* set flag indicating insertion fail */
                /* this flag is used to determine transition */
                /* out of this state */
                printf("[Queue --> %d is full]\n",Q_ISN_OUT);
                insert_ok = 0;
                }
            else{
                /* insertion was successful */
                insert_ok = 1;
                }
            /* ------------------------------------------------------------*/

            break;
```

...
...

```
            } /* end case 4 */
    case 5: { /* next queue is forecast or active scheduler */
            if (message_type == 9910)
                Q = Q_FORECAST_IN;
            else Q = Q_ACTIVE_IN;

    /*------------- queue insertion machenics------------------------ */
            if (op_subq_pk_insert (Q, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
                {
                /* the inserton failed (due to to a full queue) */
                /* deallocate the packet */
                op_pk_destroy (pkptr);

                /* set flag indicating insertion fail */
                /* this flag is used to determine transition */
                /* out of this state */
                printf("[Queue --> %d is full]\n",Q);
                insert_ok = 0;
                }
            else{
                /* insertion was successful */
                insert_ok = 1;
                }
            /* --------------------------------------------------------------*/
            break;
            } /* END CASE 5 */

    case 6: /* enqueue in ISN_OUT */
            {
            /*------------- queue insertion machenics------------------------ */
            if (op_subq_pk_insert (Q_ISN_OUT, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
                {
                /* the inserton failed (due to to a full queue) */
                /* deallocate the packet */
                op_pk_destroy (pkptr);

                /* set flag indicating insertion fail */
                /* this flag is used to determine transition */
                /* out of this state */
                printf("[Queue --> %d is full]\n",Q_ISN_OUT);
                insert_ok = 0;
                }
            else{
                /* insertion was successful */
                insert_ok = 1;
                }
            /* --------------------------------------------------------------*/

            break;
            } /* end case 6 */

    case 8: /* enqueue in SAR_OUT */
            {
            /*------------- queue insertion machenics------------------------ */
            if (op_subq_pk_insert (Q_SAR_OUT, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
                {
                /* the inserton failed (due to to a full queue) */
                /* deallocate the packet */
                op_pk_destroy (pkptr);
```

```
                /* set flag indicating insertion fail */
                /* this flag is used to determine transition */
                /* out of this state */
                printf("[Queue --> %d is full!\n",Q_SAR_OUT);
                insert_ok = 0;
                }
            else{
                /* insertion was successful */
                insert_ok = 1;
                }
            /* ------------------------------------------------------------*/

            break;
            } /* end case 8 */
    case 10: /* enqueue in SAR_OUT */
            {
            /*------------- queue insertion machenics------------------------ */
            if (op_subq_pk_insert (Q_SAR_OUT, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
                {
                /* the inserton failed (due to to a full queue) */
                /* deallocate the packet */
                op_pk_destroy (pkptr);

                /* set flag indicating insertion fail */
                /* this flag is used to determine transition */
                /* out of this state */
                printf("[Queue --> %d is full!\n",Q_SAR_OUT);
                insert_ok = 0;
                }
            else{
                /* insertion was successful */
                insert_ok = 1;
                }
            /* ------------------------------------------------------------*/

            break;
            } /* end case 10 */

    default:        /* send packet to SPS_SINK works for 3 7 and 9 */
            {
            op_pk_send(pkptr,SPS_SINK);
            break;
            } /* end case 3 */

    } /* end switch on queue index */


/* server is idle again. */
server_busy = 0;
```

transition **svc_compl -> svc_start**			
attribute	*value*	*type*	*default value*
name	tr_8	string	tr
condition	!QUEUE_EMPTY	string	
executive		string	

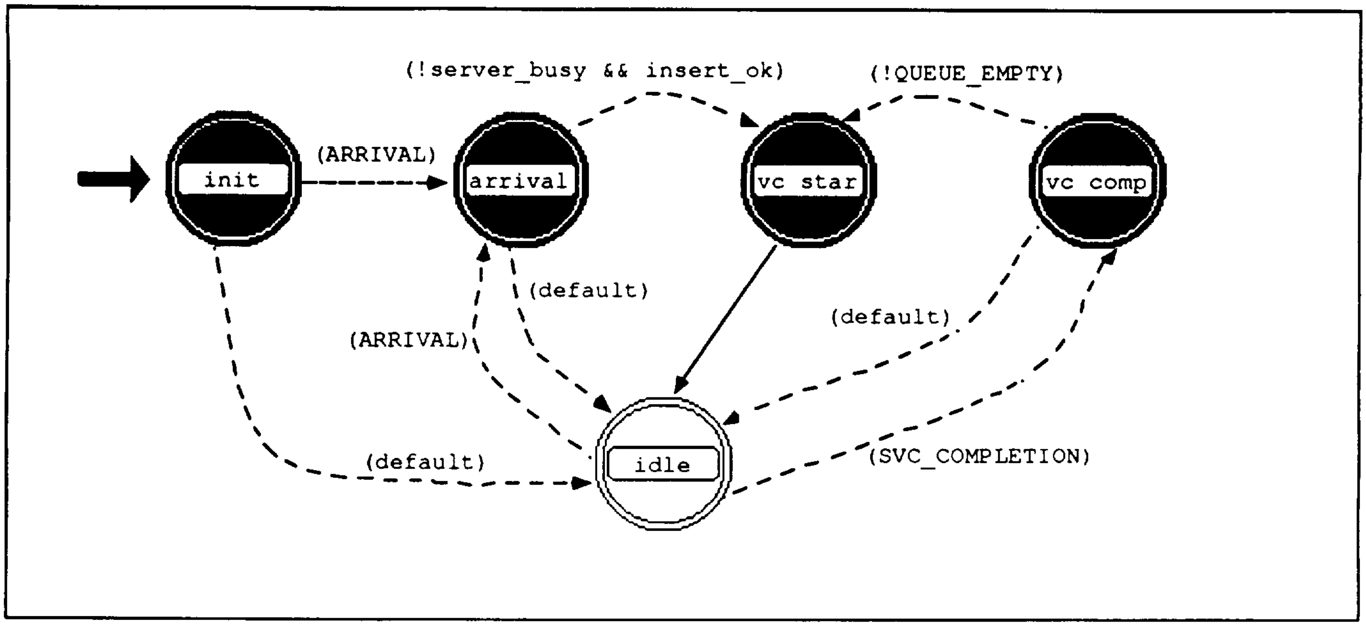
...
...

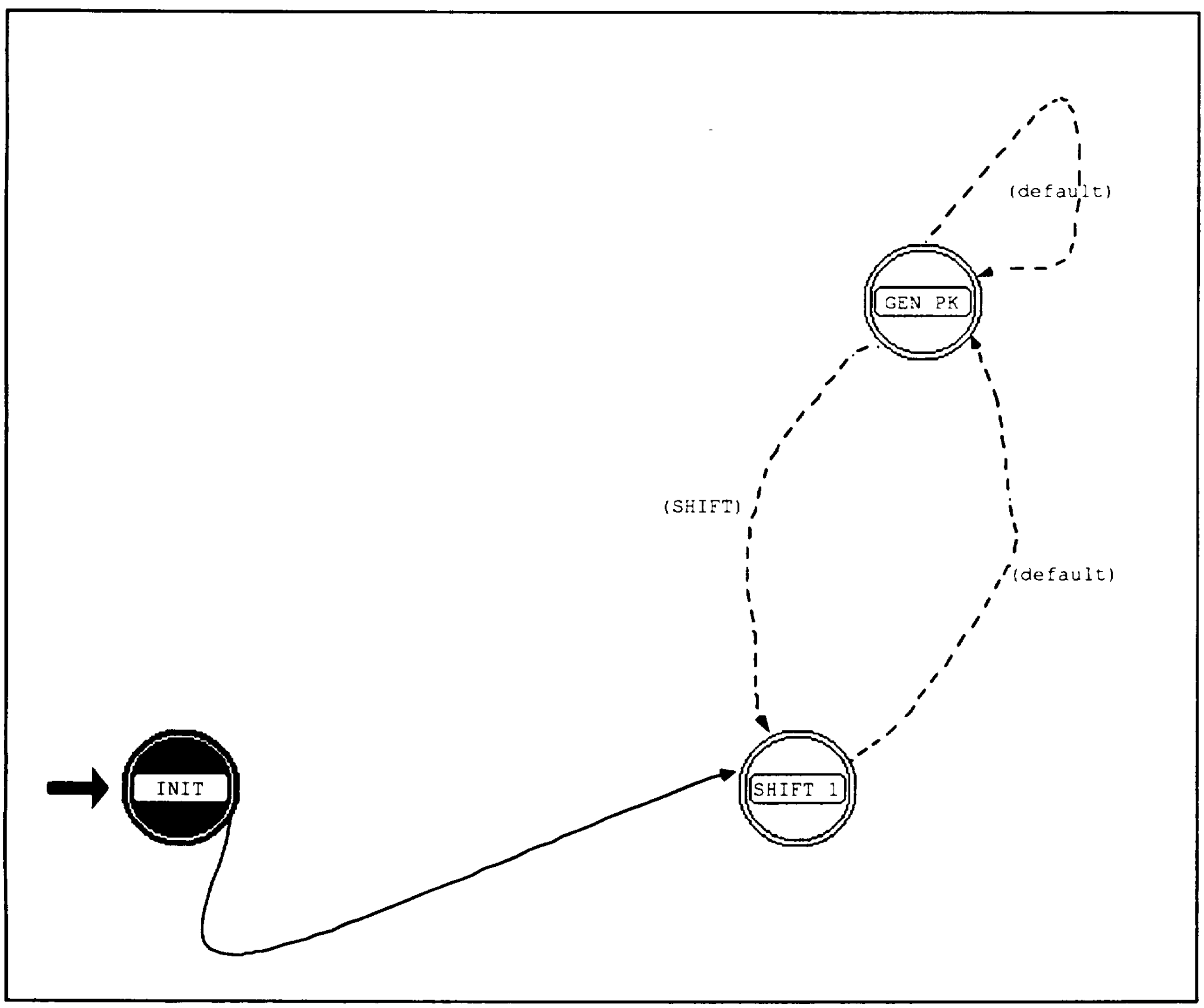
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

transition **svc_compl -> idle**

attribute	*value*	*type*	*default value*
name	tr_9	string	tr
condition	default	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

CCS Process Models





...
...

Process Model Attributes

attribute	*value*	*type*	*default value*
number of messages	promoted	integer	0
message info file 1	promoted	string	
message info file 2	promoted	string	
message info file 3	promoted	string	
station address	promoted	string	

Header Block

```
#define     SHIFT_TYPE      0001
#define     SHIFT_TIME      28800


#define     SHIFT           op_intrpt_code () == 0001

#define     MAX_STRUCT              50
#define     MAX_WORD                20
#define     EOL                     '\n'
#define     TAB                     '\t'
#define     BLANK                   ' '


/* This is the NCC message format */
typedef struct mess_struct {
                        int             type_class;
                        int             message_path;
                        double          arr_mean;
                        double          arr_var;
                        Distribution    *ptr;
                        char            dist_type[20];
                        int             dest_addr;
                        int             ack;
                } message_struct;
```

State Variable Block

```


message_struct      \mess_info[MAX_STRUCT];
int                 \start_count;
Distribution*       \dest_dist_ptr;
int                 \num_of_mess;
char*               \mess_info_file[25];
int                 \station_addr;
Objid               \active_proc_id;

```

Temporary Variable Block

```

Packet          *pk_ptr;
int             pk_count;
Distribution    *dist_ptr;
double          gen_num;
double          next_arr_time;
```

```
int         dest_addr;
char        word[20];
int         not_found;
int         i;
int         mess_type;
int         temp;
int         next_shift;
char        *tmp;
FILE        *fp, *fopen ();
double      next_time;
```

Function Block

```
 /* initalize data structure */
init_model ()
{
 int i;
 start_count = 0;

 for (i=0; i<MAX_STRUCT; i++)
     {
             mess_info[i].type_class = 0;
             mess_info[i].message_path = 0;
             mess_info[i].arr_mean = 0.0;
             mess_info[i].arr_var = 0.0;
             mess_info[i].ack = 0;
             mess_info[i].dest_addr = 0;
     }
} /* end of message initalize function */

/*  Load structure from gdf datafile */

init_msg_struct (fp, val,tmp_info)
FILE *fp;
int val;
message_struct tmp_info[MAX_STRUCT];
{
 int i;
 char s[40];
 int typeclass;
 int messagepath;
 double mean;
 double variance;
 int destaddr;
/*
 printf("\n <<<<<<<< init_struct()>>>>>>");
*/
 for(i=0; i<val; i++)
 {
   fscanf (fp,"%d %d %lf %lf %d %12s ", &typeclass, &messagepath, &mean, &variance, &destaddr, s);
/*
   printf ("\n<<type class =%d>><<message path = %d>><< arr mean= %lf>><<var= %lf>><<dest addr=%d>><<disttype
                 typeclass, messagepath, mean, variance,destaddr, s);
*/
         tmp_info[i].type_class = typeclass;
         tmp_info[i].message_path = messagepath;
         tmp_info[i].arr_mean = mean;
         tmp_info[i].arr_var = variance;
```

```
            strcpy (tmp_info[i].dist_type, s);
            tmp_info[i].dest_addr = destaddr;

    }

} /* end init structure */
```

Diagnostic Block

```
/* printf ("<<%d>>\n", num_of_mess);  */
/* printf ("\n<<%s\n>>", mess_info_file); */
```

unforced state SHIFT_1

attribute	*value*	*type*	*default value*
name	SHIFT_1	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

enter execs SHIFT_1

```
/* get number of messages */
op_ima_obj_attr_get (active_proc_id, "number of messages", &num_of_mess);
/* printf ("\n number of messages <<%d>>\n", num_of_mess);
*/
if (num_of_mess > 0)
{
    /* get file name */
    temp =op_ima_obj_attr_get (active_proc_id,"message info file 1", mess_info_file);
/* debug */
/*  printf("\n temp = <<%d>>",temp);
    printf("\n OPC_COMPCODE_SUCCESS = <<%d>>", OPC_COMPCODE_SUCCESS);
    printf("\n filename = <<%s>>",mess_info_file);
*/
    if (temp == OPC_COMPCODE_SUCCESS)
        {
            fp = fopen (mess_info_file, "r");

            init_msg_struct (fp, num_of_mess, mess_info);


            for (i=0; i< num_of_mess; i++)
                {
                /* schedule message intrpt */
/*
                printf ("type class= %d  path=%d, arrmean= %lf  arrvar= %lf destaddr=%d distype=%s\n",
                        mess_info[i].type_class, mess_info[i].message_path, mess_info[i].arr_mean,
                            mess_info[i].arr_var, mess_info[i].dest_addr, mess_info[i].dist_type);
*/
/* --------------------- variance is modified for testing -------------------------------------*/

/*                      dist_ptr = op_dist_load (mess_info[i].dist_type, mess_info[i].arr_mean, .00021);  */
```

...
...

```
                    dist_ptr = op_dist_load (mess_info[i].dist_type, mess_info[i].arr_mean, mess_info[i].arr_var);
                    next_time = op_dist_outcome (dist_ptr);
               /* check for negative outcome */
                    if (next_time < 0) next_time = fabs(next_time);
/*debug */
/*                  if (next_time > 10000.0) next_time = 100.0;                    */
/*                  printf("\n<< Inter Arrival Time=%lf>>\n",next_time);
*/
                    mess_info[i].ptr = dist_ptr;
                    next_arr_time = op_sim_time () + next_time;
                    op_intrpt_schedule_self (next_arr_time, mess_info[i].type_class);
                    } /* end for */

          } /* end if success */
          else printf("\n [[ERROR - File not found -> %s]]\n", mess_info_file);
} /* end processing message file */

/* schedule second shift */
/* next_shift = op_sim_time () + SHIFT_TIME; */

/*op_intrpt_schedule_self (next_shift, SECOND_SHIFT_TYPE);*/
```

transition **SHIFT 1 -> GEN PK**			
attribute	*value*	*type*	*default value*
name	tr_1	string	tr
condition	default	string	
executive		string	
color	RGB033	color	RGB333
drawing style	spline	toggle	spline

unforced state **GEN PK**			
attribute	*value*	*type*	*default value*
name	GEN_PK	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(See below.)	textlist	(See below.)
status	unforced	toggle	unforced

enter execs **GEN PK**

```
not_found = 0;
/* determine the type of message from the interrupt code */
mess_type = op_intrpt_code ();    /* identify message type, class */

i = 0;

/* locate record for message type, class */
/* perform linear search.  the type class will exist - no chance of infinite loop */

while (not_found == 0)
     {
          if (mess_type == mess_info[i].type_class)
               not_found = 1;
          else
               i++;
```

...
...

```
    }


/* create packet */
        pk_ptr = op_pk_create_fmt ("NCC_message");
        op_pk_nfd_set (pk_ptr,"source_addr", station_addr);
        op_pk_nfd_set (pk_ptr,"service_time", 0.000001);            /* default */
        op_pk_nfd_set (pk_ptr,"message_path", mess_info[i].message_path);
        op_pk_nfd_set (pk_ptr,"destination_addr", mess_info[i].dest_addr);
        op_pk_nfd_set (pk_ptr,"type_class", mess_info[i].type_class);

        /* send packet */
        op_pk_send (pk_ptr, 0);

/* generate next arrival of this message type */
next_time = op_dist_outcome (mess_info[i].ptr);

/* check for negative outcome */
if (next_time < 0) next_time = fabs(next_time);
/*debug */

printf("\n<< Next %d  message in =%lf seconds>>\n",mess_info[i].type_class,next_time);

next_arr_time = op_sim_time () + next_time;
op_intrpt_schedule_self (next_arr_time, mess_info[i].type_class);
```

exit execs GEN_PK

```
/* if (FIRST_SHIFT || SECOND_SHIFT || THIRD_SHIFT)
   initialize_model (); */
```

transition GEN_PK -> GEN_PK

attribute	*value*	*type*	*default value*
name	tr_2	string	tr
condition	default	string	
executive		string	
color	RGB033	color	RGB333
drawing style	spline	toggle	spline

transition GEN_PK -> SHIFT_1

attribute	*value*	*type*	*default value*
name	tr_3	string	tr
condition	SHIFT	string	
executive		string	
color	RGB030	color	RGB333
drawing style	spline	toggle	spline

...
...

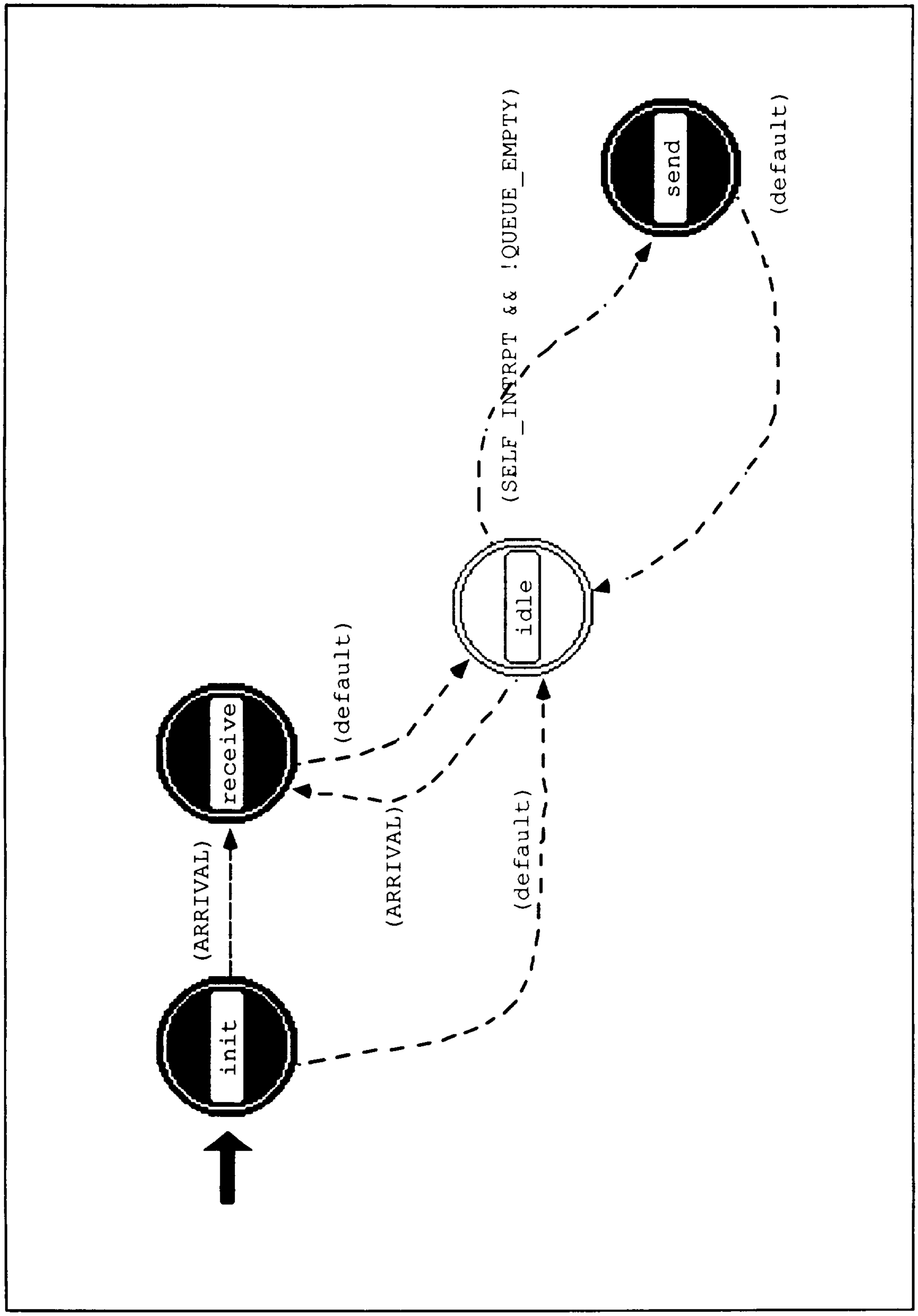
forced state **INIT**			
attribute	*value*	*type*	*default value*
name	INIT	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **INIT**

```
/* obtain ststion address from node model attribute */
active_proc_id=op_id_self();

/* get station address */
op_ima_obj_attr_get(active_proc_id,"station_address",&station_addr);

init_model ();
```

transition **INIT -> SHIFT 1**			
attribute	*value*	*type*	*default value*
name	tr_4	string	tr
condition		string	
executive		string	
color	RGB331	color	RGB333
drawing style	spline	toggle	spline

ISN Process Model



...
...

Header Block

```
#define QUEUE_EMPTY     (op_q_empty ())
#define REGULAR_INTRPT  op_intrpt_type () == OPC_INTRPT_REGULAR
#define SELF_INTRPT  op_intrpt_type () == OPC_INTRPT_SELF
#define ARRIVAL        op_intrpt_type () == OPC_INTRPT_STRM
#define SERVER_ROUTER_OUT_STREAM    1
#define MAC_SOURCE_IN_STREAM     2
#define MAC_ROUTER_IN_STREAM     1
#define ROUTER_SINK_OUT_STREAM  0
#define MAC_SERVER_OUT_STREAM      1

typedef struct
   {
    Objid           node_objid;
    Objid           queue_objid;
    } fixed_node;

/* stream indexes */
#define ISN_IN                  1
#define ISN_OUT                 1
#define ISN_TX                  0
#define ISN_RX                  0

/* queue indexes  */
#define Q_ISN_OUT               1
#define Q_ISN_IN                0

/* CCS ADDRESS to be used for routing packets to segments */
#define CCS_ADDRESS             2
```

State Variable Block

```
int     \server_busy;
double  \service_time;
Objid   \own_id;
Objid      \mac_objid;
Objid      \source_objid;

int        \station_addr;
Ici*       \mac_iciptr;
fixed_node \node_addr[5];
```

Temporary Variable Block

```
Packet*         pkptr;
Packet*         p_pkptr;

int             message_type;

double          pk_svc_time;
int             insert_ok;

int             dest_addr;
int             temp,temp1,temp2,temp3;
int             nodeaddr;
Objid           subnet, node, mac;
int             queue_index;
```

```
int                 stream_index;

```

forced state **init**			
attribute	*value*	*type*	*default value*
name	init	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **init**

```
/* determine id of the mac object */
mac_objid=op_id_self();

/* get the station address from the mac */
op_ima_obj_attr_get(mac_objid,"station_address",&station_addr);

/* get the object ids of the other fixed comm. nodes */

subnet = op_id_from_name(0,OPC_OBJTYPE_SUBNET,"ncc");

node = op_id_from_name(subnet,OPC_OBJTYPE_NODE_FIXED,"CCS");
mac  = op_id_from_name(node,OPC_OBJTYPE_QUEUE,"isn");
op_ima_obj_attr_get(mac,"station_address",&nodeaddr);
node_addr[nodeaddr].node_objid = node;
node_addr[nodeaddr].queue_objid = mac;

/* setup dummy address for external processors that must be routed through the CCS */
node_addr[0].node_objid = node;
node_addr[0].queue_objid = mac;



node = op_id_from_name(subnet,OPC_OBJTYPE_NODE_FIXED,"ITS");
mac  = op_id_from_name(node,OPC_OBJTYPE_QUEUE,"isn");
op_ima_obj_attr_get(mac,"station_address",&nodeaddr);
node_addr[nodeaddr].node_objid = node;
node_addr[nodeaddr].queue_objid = mac;

node = op_id_from_name(subnet,OPC_OBJTYPE_NODE_FIXED,"SPS");
mac  = op_id_from_name(node,OPC_OBJTYPE_QUEUE,"isn");
op_ima_obj_attr_get(mac,"station_address",&nodeaddr);
node_addr[nodeaddr].node_objid = node;
node_addr[nodeaddr].queue_objid = mac;
```

transition **init -> receive**			
attribute	*value*	*type*	*default value*
name	tr_1	string	tr
condition	ARRIVAL	string	
executive		string	
color	RGB300	color	RGB333
drawing style	line	toggle	spline

...
...

transition **init -> idle**

attribute	*value*	*type*	*default value*
name	tr_2	string	tr
condition	default	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

forced state **receive**

attribute	*value*	*type*	*default value*
name	receive	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **receive**

```
/* get stream index from interrupt */
stream_index = op_intrpt_strm ();

/* acquire the arriving packet */
/* multiple arriving streams are supported. */
pkptr = op_pk_get (stream_index);

/* packets on stream 1 -> queue 1  */
/* packets on stream 0 -> queue 0 */
queue_index = stream_index;

/* attempt to enqueue the packet at tail of subqueue 0 */
if (op_subq_pk_insert (queue_index, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
   {
   /* the inserton failed (due to to a full queue) */
   /* deallocate the packet */
   op_pk_destroy (pkptr);

      /* set flag indicating insertion fail */
      /* this flag is used to determine transition */
      /* out of this state */
   insert_ok = 0;
   }
else{
      /* insertion was successful */
   insert_ok = 1;
   /* schedule packet removal operation */
   op_intrpt_schedule_self(op_sim_time(),queue_index);
   }
```

transition **receive -> idle**

attribute	*value*	*type*	*default value*
name	tr_3	string	tr

...
...

condition	default	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

unforced state **idle**

attribute	*value*	*type*	*default value*
name	idle	string	st
enter execs	(empty)	textlist	(empty)
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

transition **idle -> receive**

attribute	*value*	*type*	*default value*
name	tr_4	string	tr
condition	ARRIVAL	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

transition **idle -> send**

attribute	*value*	*type*	*default value*
name	tr_5	string	tr
condition	SELF_INTRPT && !QUE...	string	
executive		string	
color	RGB033	color	RGB333
drawing style	spline	toggle	spline

forced state **send**

attribute	*value*	*type*	*default value*
name	send	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

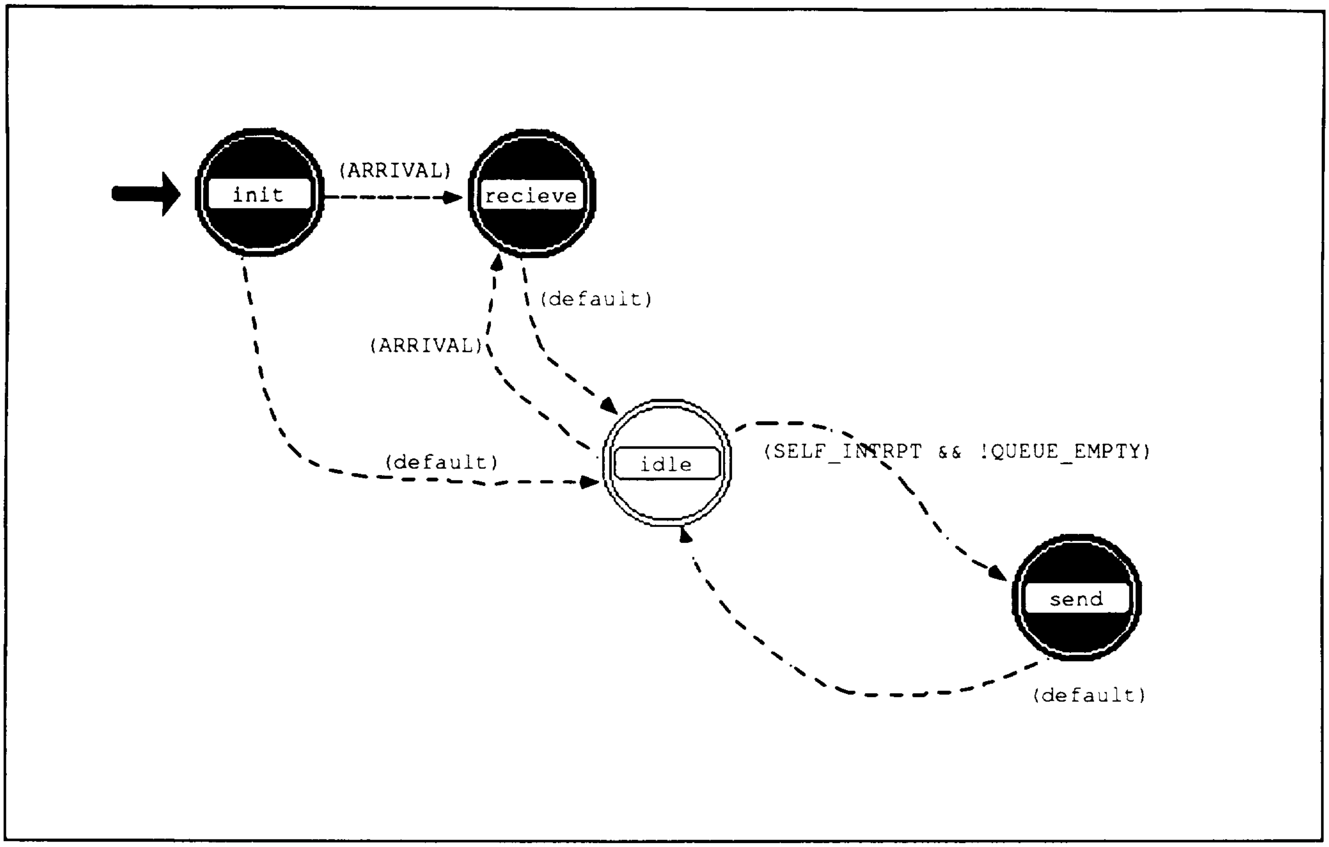
enter execs **send**

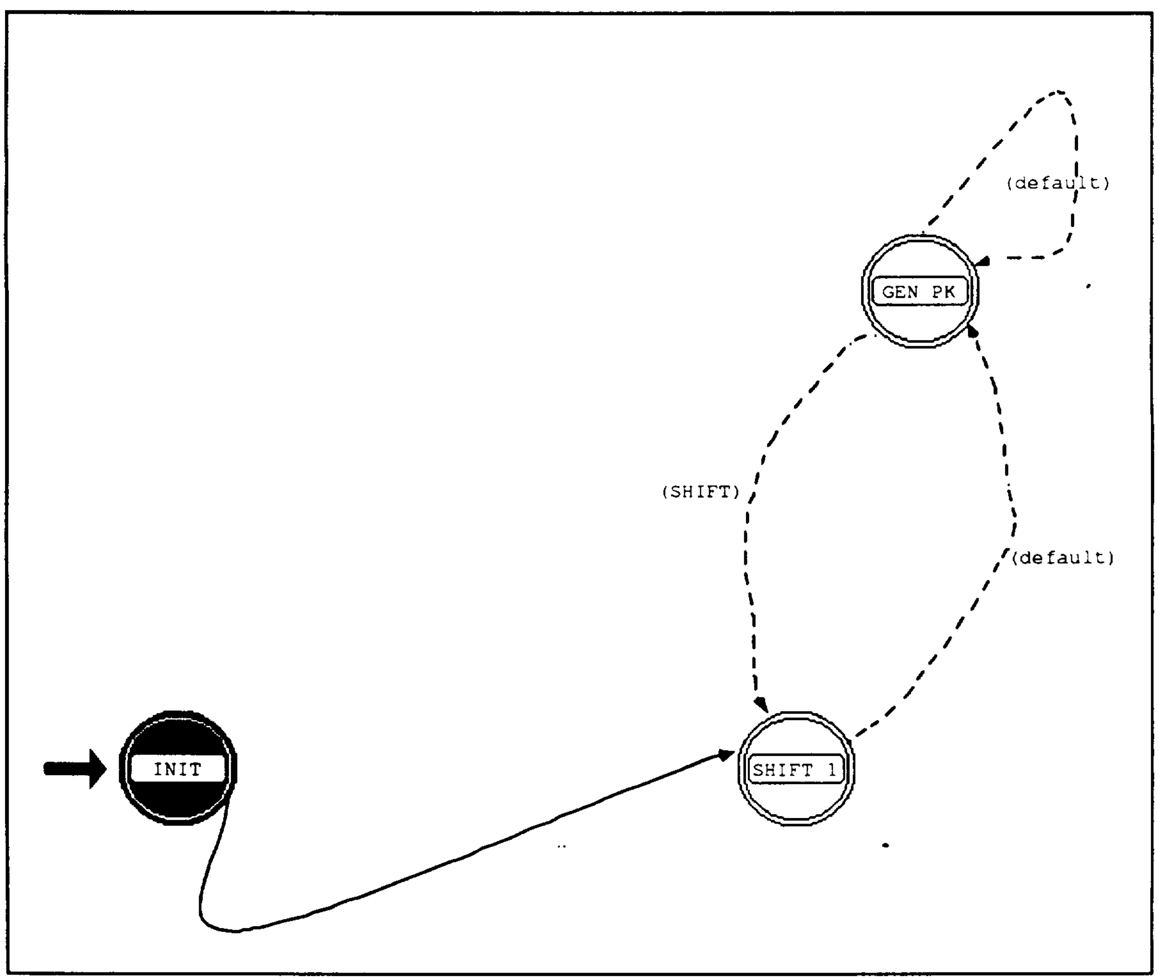
```
/* get interrupt code */
queue_index = op_intrpt_code();

/* extract packet at head of queue */
pkptr = op_subq_pk_remove (queue_index, OPC_QPOS_HEAD);

/* send queue ISN_IN packets to the server */
if (queue_index == Q_ISN_IN) op_pk_send(pkptr, ISN_OUT);
	else
		{
		/* acquire the packet parameters for processing */
		op_pk_nfd_get(pkptr,"destination_addr",&dest_addr);

		if (dest_addr > 10  && dest_addr < 18)  dest_addr = CCS_ADDRESS;
```

```
    /* send packet to server or transmit to fixed comm node */
    if (dest_addr > 1 && dest_addr < 5)
        {
        /* send to remote comm. node */
        op_pk_deliver(pkptr,node_addr[dest_addr].queue_objid,ISN_RX);
        }
        else {
            printf("\n---------------ERROR INVALID PATH------------------\n");
            op_pk_print(pkptr);
            op_pk_destroy(pkptr);
            }

    } /* queue index is Q_ISN_OUT */
```

transition **send -> idle**			
attribute	*value*	*type*	*default value*
name	tr_6	string	tr
condition	default	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

External Server and Message Generator





...
...

Header Block

```
#define QUEUE_EMPTY     (op_q_empty ())
#define REGULAR_INTRPT  op_intrpt_type () == OPC_INTRPT_REGULAR
#define SELF_INTRPT  op_intrpt_type () == OPC_INTRPT_SELF
#define ARRIVAL       op_intrpt_type () == OPC_INTRPT_STRM

/* stream constants */
#define FDF          0
#define JSC          1
#define NASCOM       2
#define NGT          3
#define POCC         4
#define SDPF         5
#define WSGT         6
#define NCC_RX       7
#define NCC_TX       7

/* SERVER QUEUES  */

#define Q_INCOMING 1
#define Q_OUTGOING 0

#define ADDRESS_OFFSET       11
```

Temporary Variable Block

```
Packet*         pkptr;
int             insert_ok;

int             dest_addr;
int             stream_index;
int             queue_index;
```

forced state **init**

attribute	*value*	*type*	*default value*
name	init	string	st
enter execs	(empty)	textlist	(empty)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

transition **init -> recieve**

attribute	*value*	*type*	*default value*
name	tr_1	string	tr
condition	ARRIVAL	string	
executive		string	
color	RGB300	color	RGB333
drawing style	line	toggle	spline

transition **init -> idle**

attribute	*value*	*type*	*default value*
name	tr_2	string	tr
condition	default	string	

executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

forced state **recieve**

attribute	*value*	*type*	*default value*
name	recieve	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **recieve**

```
/* acquire the arriving packet */
/* multiple arriving streams are supported. */
stream_index = op_intrpt_strm();
pkptr = op_pk_get (stream_index);

if (stream_index == NCC_RX) queue_index = Q_INCOMING;
      else queue_index = Q_OUTGOING;

/* attempt to enqueue the packet at tail of subqueue Q */
if (op_subq_pk_insert (queue_index, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
   {
   /* the inserton failed (due to to a full queue) */
   /* deallocate the packet */
   op_pk_destroy (pkptr);

      /* set flag indicating insertion fail */
      /* this flag is used to determine transition */
      /* out of this state */
   insert_ok = 0;
   }
else{
      /* insertion was successful */
   insert_ok = 1;
   /* schedule packet removal operation */
   op_intrpt_schedule_self(op_sim_time(),queue_index);
   }
```

transition **recieve -> idle**

attribute	*value*	*type*	*default value*
name	tr_3	string	tr
condition	default	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

...
...

unforced state **idle**

attribute	*value*	*type*	*default value*
name	idle	string	st
enter execs	(empty)	textlist	(empty)
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

transition **idle -> recieve**

attribute	*value*	*type*	*default value*
name	tr_4	string	tr
condition	ARRIVAL	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

transition **idle -> send**

attribute	*value*	*type*	*default value*
name	tr_5	string	tr
condition	SELF_INTRPT && !QUE...	string	
executive		string	
color	RGB033	color	RGB333
drawing style	spline	toggle	spline

forced state **send**

attribute	*value*	*type*	*default value*
name	send	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **send**

```
/* get queue information */
queue_index = op_intrpt_code();


/* extract packet at head of queue */
pkptr = op_subq_pk_remove (queue_index, OPC_QPOS_HEAD);

/* determine output stream */
if(queue_index == Q_OUTGOING) stream_index = NCC_TX;
    else
        {
        /* acquire the packet parameters for processing */
        op_pk_nfd_get(pkptr,"destination_addr",&dest_addr);
        stream_index = dest_addr - ADDRESS_OFFSET;
        }
/* insert packet on appropiate stream */

op_pk_send(pkptr,stream_index);
```

...
...

transition **send -> idle**			
attribute	*value*	*type*	*default value*
name	tr_6	string	tr
condition	default	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

9. NCC Messages Manual

NCC INTERNAL AND EXTERNAL MESSAGE MANUAL

JANUARY 1992

Submitted to:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

GODDARD SPACE FLIGHT CENTER
Greenbelt, Maryland 20770

Submitted by:

CENTER FOR SYSTEMS ENGINEERING AND COMPUTING

SCHOOL OF ENGINEERING
HOWARD UNIVERSITY
2300 Sixth Street, N.W.
Washington, D.C. 20059

Prepared by:

Mary Charles, Undergraduate Student
Norman Benjamin, Research Assistant

Table of Contents

INTRODUCTION

National Aeronautics and Space Administration (NASA) STDN consists of space and ground segments. The space segment is the primary source of communication and tracking for low earth orbiting satellites. The STDN comprises Space Network, Ground Network, and the Deep Space Network.

The NCC is an element of the National Aeronautics and Space Administration's (NASA) Spaceflight Tracking and Data Network (STDN). The STDN is a network that uses the Tracking and Data Relay Satellite System (TDRSS) as the primary source of support for orbiting spacecraft. The current STDN consists of the relay satellite system and several ground segments. All of the STDN ground segments are linked to the NCC at Goddard Space Flight Center (GSFC) which serves as the central control facility of the STDN. The NCC is responsible for network scheduling, acquisition and tracking support, data quality assurance, performance monitoring, overall coordination of STDN.

This document will serve to list and describe the external messages passed between the NCC and seven ground segments, which are:

1) Flight Dynamics Facility (FDF)
2) Johnson Space Center (JSC)
3) NASA Communication Network (NASCOM)
4) NASA Ground Terminal (NGT)
5) Payload Operations Control Center (POCC)
6) Sensor Data Processing Facility (SDPF)
7) White Sands Ground Terminal (WSGT)

The document will also list the internal messages of the NCC. These are messages passed between the following NCC segments:

1) Communication Control Segment (CCS)
2) Intelligent Terminal Segment (ITS)
3) Service Planning Segment (SPS)

EXTERNAL MESSAGES

This section groups messages entering and leaving the NCC by segments. To perform the functions of Service Planning, Control, Service Assurance and Accounting for the SN, the NCC requires the capability to communicate by high speed messages, secure facsimile, voice, and teletype. The primary mode of communication will be high speed messages and the others serve as supplementary and/or backup communication capability.

HIGH SPEED MESSAGES

The NCCDS has the capability of receiving and transmitting (both automatically and in response to operator request) formatted high speed data messages via secured and nonsecured communications circuits. All incoming and outgoing messages are in the standard NASCOM 4800-bit block format, as defined in NASCOM Interface Standard for Digital Data Transmission (NISDDT).

Message Handling Requirements

NCC requirements for handling electronic messages are as follows:

1. Acknowledgement

   As specified in the applicable interface control documentation, the NCCDS shall determine whether incoming messages have been received correctly or in error. For correctly received messages that indicate that an acknowledgement is requested , the NCCDS shall transmit an acknowledgement with in 2 seconds of receipt. Messages received in error shall not be acknowledged. The NCCDS shall check each correctly received message to determine if is a retransmitted message. If so the NCCDS shall determine if a previous transmission of the same message has been correctly received. If so the retransmitted message will be acknowledged but shall not be otherwise processed.

2. Validation Checking

   As specified in the preceding sections, the NCCDS will have the capability to detect invalid messages, alert the operator, and selectively log the message.

3. Message Routing. The NCCDS shall:

   Automatically route correctly received incoming messages to the appropriate functions/positions. When a destination function or position is temporarily unavailable, the NCCDS shall retain correctly received incoming messages for routing to that function or position at a later time. The NCCDS shall be capable of retaining each such message for at least two hours. Within 5 seconds of the System Supervisor's (SS) request, the NCCDS shall present a summary of such retained messages. Retained messages shall be summarized by source, type, and class. The NCCDS shall provide the SS with capability to selectively purge such retained messages by specifying one or more of source, type, class, and appropriate time related parameters (e.g requested event start time in a specific schedule add request). Send and receive all high-speed messages to and from unsecured facilities through the RAP subsystem that is currently prime. For each secured facility having a high-speed message interface with the NCC, send and receive all high-speed messages to and from that facility using the protected circuit dedicated to that interface.

4. Message Metering

The NCCDS shall be capable of metering the transmission of high speed message blocks so that the transmission rate to any destination does not exceed the maximum rate specified for that destination. The maximum transmission rate for each destination will be specified in terms of a minimum time interval between the initiation of the transmission of two successive high speed message blocks to that destination. For all messages except stand-alone acknowledgements, the NCCDS's message block transmission algorithm shall use these specified minimum time intervals to control the initiation of message transmissions to each destination. Stand-alone acknowledgement messages may be transmitted as soon as generated.

5. Message Logging.

The NCCDS shall be capable of controlling the logging and delogging of all incoming and outgoing messages from a central point under operator control. Specific logging requirements are contained in section 8 of STDN 203.13.

6. Retransmission

As specified by the applicable interface control documentation, the NCCDS shall be capable of formatting outgoing messages to indicate that acknowledgement is requested. For messages for which acknowledgement is not received within 5 seconds of transmission the NCCDS shall retransmit the message. The message shall indicate that it is a retransmission. If acknowledgement of the first retransmission is not received within 5 seconds of retransmission the NCCDS shall retransmit the message a second time. If acknowledgement of the second retransmission of a message is not received within 5 seconds of retransmission, the NCCDS shall send action alerts to the NCC console operator responsible for the acknowledged message and to the SS.

7. Acknowledgement Reporting

In all instances where the transmission of an individual high-speed message is initiated by, or requested by an NCC operator, the NCCDS shall, within 5 seconds of receipt of the acknowledgement of the transmission, present an information alert to the originating console operator. In those instances where the transmission of a stream, sequence, or batch of high-speed messages is originated by, or requested by a NCC console operator, the NCCDS will report the receipt of acknowledgements as specified elsewhere in this document.

MESSAGES COMMON TO ALL SEGMENTS

The following message are used by all the network elements.

MESSAGE NAME : ACKNOWLEDGEMENT

ORIGINATION : NCC DESTINATION : SN ELEMENT

TYPE/CLASS : 03/14

DESCRIPTION : Sent upon reception of a complete message from a user message requiring an acknowledgement.

MESSAGE NAME : ACKNOWLEDGEMENT

ORIGINATION : SN ELEMENT DESTINATION : NCC

TYPE/CLASS : 03/60

DESCRIPTION : Sent upon the reception of a complete message from the NCC requiring an acknowledgement.

MESSAGE NAME : COMMUNICATION TEST

ORIGINATION : DESTINATION :

TYPE/CLASS : 91/03

DESCRIPTION : Used to ascertain the existence of an operational communications link between two communication link. The originator must always request an acknowledgement for this message.

EXTERNAL MESSAGES BETWEEN NCC AND FDF

The FDF provides orbit-related data for unclassified spaceflight missions from early planning through to end of the operational phase. The FDF is responsible for receiving, validating (in real time), calibrating, and archiving STDN tracking data. Based on tracking data received, the FDF will provide the spacecraft/payload NASA transponder frequency history to each user. The FDF provides orbit data used in developing trajectory information, acquisition data, and scheduling aids. For each spacecraft, the FDF generates a predicted Sight Acquisition Table for a station, where a station may be a TDRS or GN site. The FDF also acts as the operations control center for the Bilateral Transponder System (BTRS). The NCC request additional data when needed.

MESSAGE NAME : USER ORBIT PREDICTION FORCE MODEL

ORIGINATION : FDF DESTINATION : NCC

TYPE/CLASS : 03/09

DESCRIPTION : This message provides the capability to define a subset of the user orbit prediction force model by specifying which components of the force model are to be used.

MESSAGE NAME : IMPROVED INTERRANGE VECTORS (IIRV) - NOMINAL

ORIGINATION : FDF DESTINATION : NCC

TYPE/CLASS : 03/10

DESCRIPTION : Provides the nominal spacecraft position and velocity vectors for the given epoch time.

MESSAGE NAME : IMPROVED INTERRANGE VECTORS (IIRV) - INFLIGHT

ORIGINATION : FDF DESTINATION : NCC

TYPE/CLASS : 03/15

DESCRIPTION : Provides the real-time spacecraft position and velocity vectors for the given epoch time.

MESSAGE NAME	:	USER SCHEDULE MESSAGE - NORMAL			
ORIGINATION	:	NCC	DESTINATION	:	FDF
TYPE/CLASS	:	94/01			
DESCRIPTION	:	Normal schedule for use of the SN.			

MESSAGE NAME	:	USER SCHEDULE MESSAGE - EMERGENCY			
ORIGINATION	:	NCC	DESTINATION	:	FDF
TYPE/CLASS	:	94/02			
DESCRIPTION	:	Emergency schedule for use of the SN.			

MESSAGE NAME	:	USER SCHEDULE MESSAGE - SIMULATION			
ORIGINATION	:	NCC	DESTINATION	:	FDF
TYPE/CLASS	:	94/03			
DESCRIPTION	:	Simulation schedule for use of the SN.			

MESSAGE NAME	:	SCHEDULE DELETION NOTIFICATION			
ORIGINATION	:	NCC	DESTINATION	:	FDF
TYPE/CLASS	:	99/01			
DESCRIPTION	:	The Schedule Deletion Notification Message is used to notify a SN user of final deletion of an event previously for that user.			

MESSAGE NAME	:	SCHEDULE RESULT MESSAGE		
ORIGINATION	:	NCC	DESTINATION :	FDF
TYPE/CLASS	:	99/02		
DESCRIPTION	:	The Schedule Result Message is sent from the NCC to the FDF in response to a Schedule Request. The message describes the results of the NCC processing of an add or delete. The NCC will transmit a message with, the appropriate code. Once the NCC has processed a valid schedule request, either send a Schedule Result Message or the appropriate schedule (for successful adds).		

MESSAGE NAME	:	SCHEDULE ADD REQUEST		
ORIGINATION	:	FDF	DESTINATION :	NCC
TYPE/CLASS	:	99/10		
DESCRIPTION	:	Adds a single schedule event.		

MESSAGE NAME	:	SCHEDULE DELETE REQUEST		
ORIGINATION	:	FDF	DESTINATION :	NCC
TYPE/CLASS	:	99/11		
DESCRIPTION	:	Deletes a single event.		

EXTERNAL MESSAGES BETWEEN NCC AND JSC

The JSC provides command, control, and systems monitoring capabilities for the Space Transportation System (STS). To support the STS, the MCC is required to interface with NCC to schedule the STDN and NASCOM resources. The NCC will receive and display performance data and transmit ground control messages requests (GCMR's) that results in certain reconfiguration of the space network.

The NCC receives performance data from the WSGT and provide this information to the MCC once every 5 seconds in the format described in this document. At the MCC, the performance data will be routed to the network communications interface common (NCIC), which performs certain validation checks on the network header, and routes this data to the mission operations computer (MOC). The MOC interprets, formats, and provides performance data for use by the flight control team.

The JSC generates GCMR's which results in configuration changes in the SN. These GCMR's are generated within the MOC as a result of operator action and are routed to the NCC by NASCOM, where certain validation is performed prior to transmission of the corresponding ground control message to the WSGT. The NCC processing of the GCMR's is a real-time function. Message protocol will be invoked, and the GCM status and dispositions will be provided by the NCC.

MESSAGE NAME : ACQUISITION FAILURE NOTIFICATION

ORIGINATION : NCC DESTINATION : JSC

TYPE/CLASS : 92/63

DESCRIPTION : When TDRS fails to acquire the signal from a user spacecraft, an acquisition failure is sent to the JSC.

MESSAGE NAME : USER SCHEDULE MESSAGES - NORMAL

ORIGINATION : NCC DESTINATION : JSC

TYPE/CLASS : 94/01

DESCRIPTION : User normal schedule for SN.

MESSAGE NAME : USER SCHEDULE MESSAGES - EMERGENCY

ORIGINATION : NCC DESTINATION : JSC

TYPE/CLASS : 94/02

DESCRIPTION : User emergency schedule for SN.

MESSAGE NAME : GCM STATUS

ORIGINATION : NCC DESTINATION : JSC

TYPE/CLASS : 98/01

DESCRIPTION : These messages indicates acceptance or reason for rejection of user transmitted GCMR.

MESSAGE NAME : GCM DISPOSITION MESSAGE

ORIGINATION : NCC DESTINATION : JSC

TYPE/CLASS : 98/02

DESCRIPTION : Indication to the JSC of whether or not an acknowledgement was received from the SN.

MESSAGE NAME : REACQUISITION REQUEST

ORIGINATION : JSC DESTINATION : NCC

TYPE/CLASS : 98/03

DESCRIPTION : Provides the JSC with the capability to request reacquisition of service.

MESSAGE NAME : RECONFIGURATION REQUEST

ORIGINATION : JSC DESTINATION : NCC

TYPE/CLASS : 98/04

DESCRIPTION : These are four messages providing the JSC with the capability to request a reconfiguration to the specified services.

MESSAGE NAME : FORWARD LINK SWEEP REQUEST

ORIGINATION : JSC DESTINATION : NCC

TYPE/CLASS : 98/05

DESCRIPTION : Provides the JSC with the capability to request a Forward Link Sweep.

MESSAGE NAME : FORWARD LINK EIRP RECONFIGURATION

ORIGINATION : JSC DESTINATION : NCC

TYPE/CLASS : 98/06

DESCRIPTION : Provides the JSC with the capability to reconfigure the SSA and KSA Forward EIRP between normal and high power mode on the TDRS.

MESSAGE NAME : EXPANDED USER FREQUENCY UNCERTAINTY REQUEST

ORIGINATION : JSC DESTINATION : NCC

TYPE/CLASS : 98/07

DESCRIPTION : Provides the JSC with the capability of expanding the frequency uncertainty of the referenced schedule return event.

MESSAGE NAME : DOPPLER COMPENSATION INHIBIT REQUEST

ORIGINATION : JSC DESTINATION : NCC

TYPE/CLASS : 98/08

DESCRIPTION : Provides the JSC with the capability to inhibit forward link doppler compensation on a specific link.

MESSAGE NAME	:	SCHEDULE RESULT MESSAGE			
ORIGINATION	:	NCC	DESTINATION	:	JSC
TYPE/CLASS	:	99/02			
DESCRIPTION	:	Sent to user in response to SAR.			

MESSAGE NAME	:	SPECIFIC SCHEDULE REQUEST MESSAGE - ADD			
ORIGINATION	:	JSC	DESTINATION	:	NCC
TYPE/CLASS	:	99/10			
DESCRIPTION	:	These messages are used to add shuttle events for network resources.			

MESSAGE NAME	:	SPECIFIC SCHEDULE REQUEST MESSAGE - DELETE			
ORIGINATION	:	JSC	DESTINATION	:	NCC
TYPE/CLASS	:	99/11			
DESCRIPTION	:	These messages are used to delete shuttle events for network resources.			

EXTERNAL MESSAGES BETWEEN NCC AND NASCOM

NASCOM provides common carrier communication services among the TDRSS ground segment (including NGT), Johnson Space Center (JSC), and GSFC using a wideband data system interfaced through a Multiplexer/Demultiplexer (MDM) system and a Statistical Multiplexer (SM) system. As part of NASCOM, MDM and SM units are located at the TDRSS ground segment, JSC, and GSFC. The MDM baseline composite transmission service will be 6 mb/sec from NGT and 2.5 mb/sec to the TDRSS ground segment. Spacecraft data with rates up to 2 mb/sec will normally be transmitted from the TDRSS ground segment by MDM. Spacecraft telemetry data with higher rates will be transmitted by the SM which is capable of transmitting up to four channels of data simultaneously with a maximum composite data rate of 48 mb/sec. Data from TDRSS ground segment is transmitted to JSC and GSFC simultaneously. Data to the TDRSS ground segment from GSFC and JSC will be transmitted via the MDM only. In addition, NASCOM provides TV, voice, TTY, and systems control circuits.

NASCOM operates within the STDN in accordance with a schedule provided by the NCC and reconfigure equipment in response to direction from the NCC. NASCOM provides the NCC with the status of services and also provides a postevent performance summary.

NASCOM will also provide multiple 56-kb/sec circuits or a 224-kb/sec circuit among the GN, GSFC, JSC, and the NCC. Additional circuits will be provided to support the communication interfaces among the NCC, GSFC, DOD control centers, and other NASA control centers.

MESSAGE NAME : NASCOM EVENT SCHEDULE

ORIGINATION : NCC DESTINATION : NASCOM

TYPE/CLASS : 90/01

DESCRIPTION : Notifies Nascom of all scheduled services which involve data flow by sending a NES for each event.

MESSAGE NAME	:	NASCOM EVENT CANCEL		
ORIGINATION	:	NCC	DESTINATION :	NASCOM
TYPE/CLASS	:	90/02		
DESCRIPTION	:	Notifies Nascom that a pending or active event is canceled.		

MESSAGE NAME	:	NASCOM EVENT SCHEDULE UPDATE		
ORIGINATION	:	NCC	DESTINATION :	NASCOM
TYPE/CLASS	:	90/04		
DESCRIPTION	:	Used add events to an already established Nascom CSS Schedule.		

MESSAGE NAME	:	NASCOM EVENT SCHEDULE EMERGENCY		
ORIGINATION	:	NCC	DESTINATION :	NASCOM
TYPE/CLASS	:	90/05		
DESCRIPTION	:	Used to add an event with very short lead time to the Nascom CSS Schedule. A NESE is used when the start of an event is less than 45 minutes, but a least 5 minutes away from the time that the message is transmitted to the Nascom CSS.		

MESSAGE NAME	:	NASCOM RECONFIGURATION REQUEST		
ORIGINATION	:	NCC	DESTINATION :	NASCOM
TYPE/CLASS	:	90/06		
DESCRIPTION	:	Includes up to five data streams within a single active service of an ongoing event, up to three of which may be active. A NRR is normally executed within 15 seconds.		

EXTERNAL MESSAGES BETWEEN NCC AND NGT

The NGT provides the interface between the NASCOM/common carrier and the TDRSS services. The NGT receives schedule messages based upon user requests. The NGT Scheduling System (NSS) schedules and allocates selected NGT resources based on these messages and provides status back to the NCC. The NGT Control and Status System (NCSS) will control and configure the NGT equipment. The NGT also sends data monitoring results and status reports to the NCC. The NCC uses data monitoring results for fault isolation and TDRSS data accountability.

MESSAGE NAME	:	NGT SCHEDULING SYSTEM EVENT ADD - NORMAL
ORIGINATION	:	NCC DESTINATION : NGT
TYPE/CLASS	:	86/01
DESCRIPTION	:	NSS Event Add are used to transmit normal event from NCC to NGT when the event start time is more 45 minutes in the future from the time the event was added to the NCC data base.

MESSAGE NAME	:	NGT SCHEDULING SYSTEM EVENT ADD - EMERGENCY
ORIGINATION	:	NCC DESTINATION : NGT
TYPE/CLASS	:	86/02
DESCRIPTION	:	NSS Event Add message is used to transmit emergency schedule events from NCC to the NGT. A schedule event will be transmitted as an emergency event when the start time is less than 45 minutes but more than 5 minutes in the future from the time the event was added to the NCC data base.

MESSAGE NAME : NGT SCHEDULING SYSTEM - EVENT DELETE

ORIGINATION : NCC DESTINATION : NGT

TYPE/CLASS : 86/03

DESCRIPTION : NSS Event Delete messages are used by the NCC to delete events from the NGT schedule. Event may be deleted up to and including the time that they are active.

MESSAGE NAME : NGT SCHEDULING SYSTEM - SERVICE RECONFIGURATION

ORIGINATION : NCC DESTINATION : NGT

TYPE/CLASS : 86/04

DESCRIPTION : These messages are sent to the NGT as directives to change one or more data streams within an ongoing service of a user event. The reconfiguration are specified on a service level and are limited to changes to the TDRSS interface channel, data rate and data stream ID for each data with in the service.

MESSAGE NAME : NGT SCHEDULE SYSTEM-EVENT STATUS

ORIGINATION : NGT DESTINATION : NCC

TYPE/CLASS : 86/51

DESCRIPTION : NSS Schedule Status messages are transmitted by the NGT to the NCC in response to NSS Add and Delete messages.

MESSAGE NAME	:	NGT SCHEDULING SYSTEM- RECONFIGURATION ACCEPT/REJECT			
ORIGINATION	:	NGT	DESTINATION	:	NCC
TYPE/CLASS	:	86/54			
DESCRIPTION	:	NSS Reconfiguration Accept/Reject messages are transmitted from NGT to NCC in response to a NGT Service Reconfiguration Request. These messages indicate that the NGT has accepted or rejected the referenced reconfiguration request and if rejected, the reason for rejection.			

MESSAGE NAME	:	ADMINISTRATIVE			
ORIGINATION	:	NCC	DESTINATION	:	NGT
TYPE/CLASS	:	88/01			
DESCRIPTION	:	Administrative messages are used to exchange free text format text from NCC to NGT.			

MESSAGE NAME	:	FAULT ISOLATION & MONITORING SYSTEM REPORTS			
ORIGINATION	:	NGT	DESTINATION	:	NCC
TYPE/CLASS	:	88/03			
DESCRIPTION	:	FIMS messages are used to transmit FIMS Data quality information, collected from the channels monitored, to the NCC.			

MESSAGE NAME	:	ADMINISTRATIVE MESSAGE			
ORIGINATION	:	NGT	DESTINATION	:	NCC
TYPE/CLASS	:	88/54			
DESCRIPTION	:	Administrative message are used to exchange free format alphanumeric text between the NGT and NCC.			

EXTERNAL MESSAGES BETWEEN NCC AND POCC

MESSAGE NAME : USER PERFORMANCE DATA MESSAGE

ORIGINATION : NCC DESTINATION : POCC

TYPE/CLASS : 91/01

DESCRIPTION : TDRS Performance Data requested by user POCC for Schedule Event. No acknowledgement required.

MESSAGE NAME : USER PERFORMANCE DATA REQUEST

ORIGINATION : POCC DESTINATION : NCC

TYPE/CLASS : 92/04

DESCRIPTION : Allows user to select or deactivate operation data messages.

MESSAGE NAME : RETURN CHANNEL TIME DELAY DATA

ORIGINATION : NCC DESTINATION : POCC

TYPE/CLASS : 92/52

DESCRIPTION : Used to transmit return channel time delay measurement data from NCC to user.

MESSAGE NAME : ACQUISITION FAILURE NOTIFICATION

ORIGINATION : NCC DESTINATION : POCC

TYPE/CLASS : 92/63

DESCRIPTION : Notifies the user that return services did not occur due to the in ability of TDRSS to acquire user spacecraft.

MESSAGE NAME	:	CONFIRM NORMAL SCHEDULE		
ORIGINATION	:	NCC	DESTINATION :	POCC
TYPE/CLASS	:	94/01		
DESCRIPTION	:	Generated for Forecast Week transmission or when nonemergency add executed during active time frame.		

MESSAGE NAME	:	CONFIRM PREMIUM SUPPORT SCHEDULE		
ORIGINATION	:	NCC	DESTINATION :	POCC
TYPE/CLASS	:	94/02		
DESCRIPTION	:	Generated when a schedule add is executed within 45 minutes of event start time.		

MESSAGE NAME	:	CONFIRM SIMULATION SCHEDULE		
ORIGINATION	:	NCC	DESTINATION :	POCC
TYPE/CLASS	:	94/03		
DESCRIPTION	:	Generated when simulation event is added an active time frame.		

MESSAGE NAME	:	GCM STATUS MESSAGE		
ORIGINATION	:	NCC	DESTINATION :	POCC
TYPE/CLASS	:	98/01		
DESCRIPTION	:	Generated when GCMR receipt acknowledgement received or when Operation Message (OPM) status acceptance/rejection message SN site.		

MESSAGE NAME : GCM DISPOSITION

ORIGINATION : NCC DESTINATION : POCC

TYPE/CLASS : 98/02

DESCRIPTION : Transmitted to the user to indicate whether or not an acknowledgement was received from WSGT.

MESSAGE NAME : REACQUISITION REQUEST

ORIGINATION : POCC DESTINATION : NCC

TYPE/CLASS : 98/03

DESCRIPTION : Provides the user the capability to request a service compatible link reacquisition procedure.

MESSAGE NAME : USER RECONFIGURATION REQUEST

ORIGINATION : POCC DESTINATION : NCC

TYPE/CLASS : 98/04

DESCRIPTION : Provides the user the capability to request a reconfiguration of a specified service.

MESSAGE NAME : FORWARD LINK SWEEP REQUEST

ORIGINATION : POCC DESTINATION : NCC

TYPE/CLASS : 98/05

DESCRIPTION : Provides the user the capability to request a forward link sweep on the designated service.

MESSAGE NAME	:	FORWARD LINK EIRP RECONFIGURATION REQUEST			
ORIGINATION	:	POCC	DESTINATION	:	NCC
TYPE/CLASS	:	98/06			
DESCRIPTION	:	Provides the user the capability to request a reconfiguration of the SSA or KSA forward Link EIRP between normal and high power at TDRSS.			

MESSAGE NAME	:	EXPANDER USER FREQUENCY UNCERTAINTY REQUEST			
ORIGINATION	:	POCC	DESTINATION	:	NCC
TYPE/CLASS	:	98/07			
DESCRIPTION	:	Provides the user the capability to expand the frequency uncertainty of the referenced ongoing return service.			

MESSAGE NAME	:	DOPPLER COMPENSATION INHIBIT REQUEST			
ORIGINATION	:	POCC	DESTINATION	:	NCC
TYPE/CLASS	:	98/08			
DESCRIPTION	:	Provides the user with the capability to request that Forward Link Doppler Compensation on specified link be inhibited.			

MESSAGE NAME	:	SCHEDULE DELETION NOTIFICATION			
ORIGINATION	:	NCC	DESTINATION	:	POCC
TYPE/CLASS	:	99/01			
DESCRIPTION	:	Used to notify user of pending or final deletion of an event.			

MESSAGE NAME	:	SCHEDULE ACCEPT/REJECT NOTIFICATION			
ORIGINATION	:	NCC	DESTINATION	:	POCC
TYPE/CLASS	:	99/02			
DESCRIPTION	:	Sent to user in response to a schedule request.			

MESSAGE NAME	:	SCHEDULE ADD REQUEST			
ORIGINATION	:	POCC	DESTINATION	:	NCC
TYPE/CLASS	:	99/10			
DESCRIPTION	:	Used to request addition of an event to the schedule.			

MESSAGE NAME	:	SCHEDULE DELETE REQUEST			
ORIGINATION	:	POCC	DESTINATION	:	NCC
TYPE/CLASS	:	99/11			
DESCRIPTION	:	Used by POCC to request deletion of an event from the schedule.			

EXTERNAL MESSAGES BETWEEN NCC AND SDPF

The SDPF is a user support facility that processes telemetry data for earth-orbiting free-flyer payloads. The SDPF provides for data input capture, accounting, decommutation, and storing and forwarding of standard products. The SDPF also processes image data and provides rectification, calibration, and user/experimenter products such as computer-compatible tapes, film, prints, and plots. Project-unique requirements and unique data products can be provided to a user under formalized agreements.

In response to requests from users with the SDPF specified as a destination for return service data, the NCC schedules the flow of data to SDPF and provides the SDPF with schedules. The SDPF prepares to receive the process and telemetry data based on the schedule. In response to request from users to reconfigure on going services, the NCC notifies the SN elements and the SDPF will adjust to any reconfiguration affecting the flow of return data to the SDPF.

MESSAGE NAME : NASCOM EVENT SCHEDULE (NES)

ORIGINATION : NCC DESTINATION : SDPF

TYPE/CLASS : 90/01

DESCRIPTION : NES contains information of all scheduled services which involve data flow. This NES message is also use NCCDS to schedule Nascom resources needed to support an SN event. Each NES will add an event to the SDPF.

MESSAGE NAME : NASCOM EVENT CANCEL (NEC)

ORIGINATION : NCC DESTINATION : SDPF

TYPE/CLASS : 90/02

DESCRIPTION : The NES is used to cancel resource allocations previously scheduled by an NES or to cancel an active event. May be transmitted at any time prior to or during an event.

MESSAGE NAME : NASCOM EVENT SCHEDULE UPDATE (NESU)

ORIGINATION : NCC DESTINATION : SDPF

TYPE/CLASS : 90/04

DESCRIPTION : NESU sent greater than 45 minutes prior to event start time.

MESSAGE NAME : NASCOM EVENT SCHEDULE EMERGENCY (NESE)

ORIGINATION : NCC DESTINATION : SDPF

TYPE/CLASS : 90/05

DESCRIPTION : NESE is functionally identical to a NES message except that the NESE is used when the start of the event being scheduled is less than 45 minutes but a least 5 minutes away from the time that the message is transmitted to Nascom CSS.

MESSAGE NAME : NASCOM RECONFIGURATION REQUEST (NRR)

ORIGINATION : NCC DESTINATION : SDPF

TYPE/CLASS : 90/06

DESCRIPTION : NRR is a ground control message use to reconfigure data streams in an active service of an ongoing event. Each service within an event requires a separate NRR message.

EXTERNAL MESSAGES BETWEEN NCC AND WSGT

WSGT operates in accordance with a schedule provided by the NCC and changes ongoing service parameters in response to NCC instructions. TDRS antenna pointing angles and Doppler compensation information are determined from detailed spacecraft orbit data. WSGT compute this information by propagating a state vector using a predefined force model. Both the state vector and force model are provided by the NCC. WSGT inform the NCC of the status and quality of ongoing services and also of equipment status. Based on WSGT requests, the NCC schedules WSGT Preventive Maintenance (PM) on a service basis.

MESSAGE NAME : NORMAL SHO

ORIGINATION : NCC DESTINATION : WSGT

TYPE/CLASS : 02/01

DESCRIPTION : Describes services contained in a normal event.

MESSAGE NAME : SIMULATION SHO

ORIGINATION : NCC DESTINATION : WSGT

TYPE/CLASS : 02/03

DESCRIPTION : Describes the services contained in a routine verification event.

MESSAGE NAME : ROUTINE VERIFICATION SHO

ORIGINATION : NCC DESTINATION : WSGT

TYPE/CLASS : 02/04

DESCRIPTION : Describes the service contained in a routine verification event.

MESSAGE NAME : EMERGENCY ROUTINE VERIFICATION (ERVS)

ORIGINATION : NCC DESTINATION : WSGT

TYPE/CLASS : 02/05

DESCRIPTION : Describes the services contained in a emergency routine verification event.

MESSAGE NAME : SPECIAL REQUEST

ORIGINATION : NCC DESTINATION : WSGT

TYPE/CLASS : 03/01

DESCRIPTION : Used to send free-form alpha-numeric text messages.

MESSAGE NAME : REACQUISITION REQUEST

ORIGINATION : NCC DESTINATION : WSGT

TYPE/CLASS : 03/02

DESCRIPTION : Used to initiate a reacquisition.

MESSAGE NAME : RECONFIGURATION REQUEST

ORIGINATION : NCC DESTINATION : WSGT

TYPE/CLASS : 03/03

DESCRIPTION : Used to reconfigure equipment supporting a user spacecraft.

MESSAGE NAME : FORWARD LINK SWEEP REQUEST

ORIGINATION : NCC DESTINATION : WSGT

TYPE/CLASS : 03/04

DESCRIPTION : Used to initiate a sweep of forward link carrier frequency.

MESSAGE NAME : FORWARD LINK EIRP RECONFIGURATION REQUEST

ORIGINATION : NCC DESTINATION : WSGT

TYPE/CLASS : 03/06

DESCRIPTION : Used to set the SSA or KSA EIRP to normal or high power.

MESSAGE NAME : EXPANDER USER FREQUENCY UNCERTAINTY REQUEST

ORIGINATION : NCC DESTINATION : WSGT

TYPE/CLASS : 03/07

DESCRIPTION : Used to increase receiver bandwidth for DG1, mode 2 and DG2.

MESSAGE NAME : USER ORBIT PREDICTION FORCE MODEL

ORIGINATION : NCC DESTINATION : WSGT

TYPE/CLASS : 03/09

DESCRIPTION : Provides information that the TDRSS uses to propagate a stable vector.

MESSAGE NAME : IMPROVED INTERRANGE VECTOR (IIRV) - NOMINAL

ORIGINATION : NCC DESTINATION : WSGT

TYPE/CLASS : 03/10

DESCRIPTION : Provides spacecraft position and velocity vectors to be used in scheduling. This message is generated at FDF.

MESSAGE NAME : DOPPLER COMPENSATION INHIBIT REQUEST

ORIGINATION : NCC DESTINATION : WSGT

TYPE/CLASS : 03/11

DESCRIPTION :

MESSAGE NAME : CANCEL SHO REQUEST

ORIGINATION : NCC DESTINATION : WSGT

TYPE/CLASS : 03/12

DESCRIPTION :

MESSAGE NAME : TDRS MANEUVER APPROVAL

ORIGINATION : NCC DESTINATION : WSGT

TYPE/CLASS : 03/13

DESCRIPTION :

MESSAGE NAME : DELTA-T-ADJUSTMENT

ORIGINATION : NCC DESTINATION : WSGT

TYPE/CLASS : 03/18

DESCRIPTION : Used to adjust the epoch time parameter within state vectors.

MESSAGE NAME : SHO STATUS

ORIGINATION : WSGT DESTINATION : NCC

TYPE/CLASS : 03/51

DESCRIPTION :

MESSAGE NAME : RETURN CHANNEL TIME DELAY

ORIGINATION : WSGT DESTINATION : NCC

TYPE/CLASS : 03/52

DESCRIPTION :

MESSAGE NAME	:	PREVENTATIVE MAINTENANCE REQUEST			
ORIGINATION	:	WSGT	DESTINATION	:	NCC
TYPE/CLASS	:	03/53			
DESCRIPTION	:	Used to request TDRSS preventive maintenance.			

MESSAGE NAME	:	SPECIAL REQUEST OR INFORMATION			
ORIGINATION	:	WSGT	DESTINATION	:	NCC
TYPE/CLASS	:	03/54			
DESCRIPTION	:	Used to send free-form alphanumeric text.			

MESSAGE NAME	:	RESULTS OF ROUTINE VERIFICATION			
ORIGINATION	:	WSGT	DESTINATION	:	NCC
TYPE/CLASS	:	03/55			
DESCRIPTION	:				

MESSAGE NAME	:	TDRS MANEUVER REQUEST			
ORIGINATION	:	WSGT	DESTINATION	:	NCC
TYPE/CLASS	:	03/59			
DESCRIPTION	:	Used to request approval for a TDRS spacecraft maneuver.			

MESSAGE NAME	:	STATE VECTOR REJECTION			
ORIGINATION	:	WSGT	DESTINATION	:	NCC
TYPE/CLASS	:	03/61			
DESCRIPTION	:				

MESSAGE NAME	:	OPM STATUS			
ORIGINATION	:	WSGT	DESTINATION	:	NCC
TYPE/CLASS	:	03/62			
DESCRIPTION	:	Used to accept or reject OPM.			

MESSAGE NAME	:	ACQUISITION FAILURE NOTIFICATION			
ORIGINATION	:	WSGT	DESTINATION	:	NCC
TYPE/CLASS	:	03/63			
DESCRIPTION	:	Provides notification that TDRS cannot acquire a user spacecraft.			

MESSAGE NAME	:	STATE VECTOR PROPAGATION COMPLETE			
ORIGINATION	:	WSGT	DESTINATION	:	NCC
TYPE/CLASS	:	03/64			
DESCRIPTION	:				

MESSAGE NAME	:	DELTA-T ADJUSTMENT REJECTION			
ORIGINATION	:	WSGT	DESTINATION	:	NCC
TYPE/CLASS	:	03/65			
DESCRIPTION	:				

MESSAGE NAME	:	SERVICE LEVEL STATUS REPORT			
ORIGINATION	:	WSGT	DESTINATION	:	NCC
TYPE/CLASS	:	04			
DESCRIPTION	:				

MESSAGE NAME : SA OPERATIONS DATA MESSAGE

ORIGINATION : WSGT DESTINATION : NCC

TYPE/CLASS : 05

DESCRIPTION :

MESSAGE NAME : MA OPERATIONS DATA MESSAGE

ORIGINATION : WSGT DESTINATION : NCC

TYPE/CLASS : 06

DESCRIPTION :

MESSAGE NAME : SIMULATION OPERATIONS DATA MESSAGE

ORIGINATION : WSGT DESTINATION : NCC

TYPE/CLASS : 07

DESTINATION :

MESSAGE NAME : PERIODIC SHO - NORMAL

ORIGINATION : NCC DESTINATION : WSGT

TYPE/CLASS : 08/01

DESCRIPTION :

MESSAGE NAME : PERIODIC SHO - SIMULATION

ORIGINATION : NCC DESTINATION : WSGT

TYPE/CLASS : 08/03

DESCRIPTION :

MESSAGE NAME	:	PERIODIC SHO - ROUTINE VERIFICATION			
ORIGINATION	:	NCC	DESTINATION	:	WSGT
TYPE/CLASS	:	08/04			
DESCRIPTION	:				

INTERNAL MESSAGES

This section describes the message interface of the segments of the NCC. The segments of the NCC are:

1) Communications and Control Segment (CCS)
   Major functions include :
   a) Service Control
   b) Service Assurance
   c) Service Accounting
   d) System Operation

2) Intelligent Terminal Segment (ITS)
   Major functions include :
   a) Service Assurance
   b) System Operation

3) Service Planning Segment (SPS)
   Major functions include :
   a) Service Planning
   b) Service Control
   c) Service Assurance
   d) Service Accounting
   e) System Operation

The interfaces between CCS/ITS, SPS/ITS, and CCS/SPS are through the dual-rail Intersegment Local Area Network (LAN). Each of these interfaces can be on either rail of the LAN and are not , necessarily, all using the same rail at one particular time.

The NCCDS Intersegment message is used for all messages exchanged between segments. Each message begins with and some messages consists solely of a three-word LAN header followed by an eight-word NCC header. The remaining messages may contain a variable length NCC subheader following the NCC header, and a variable length data area.

The LAN header is prepared by the segment software that routes the message across the LAN and is used by the segment software that receives the message from the LAN to put the message pages together to form the complete message. The NCC header is prepared by the software that builds the intersegment message and is used to route the message to the receiving software which uses it to determine the message characteristics. The NCC subheader is used in cases where additional header information is needed.

INTERNAL MESSAGES BETWEEN THE CCS AND ITS

This interface is through the dual-rail Intersegment Local Area Network (LAN) and can be on either rail of the LAN at any particular time. Each message passed between the ITS and CCS is uniquely identified by a combination of the NCC Function Type, the NCC Command Code / Function Code, and the NCC Command Subcode. The function type is used to identify the segments involved. The ITS / CCS interface is identical to the ITS / SPS interface with the exception of the function type. The ITS can identify the sending segment (CCS or SPS) of a message by its function type or the LAN connection on which the message was received because LAN connections are unique rather than shared.

MESSAGE NAME : Alert Additional Data Display

ORIGINATION : CCS DESTINATION : ITS

FUNCTION TYPE : 5 FUNCTION CODE : 5

MESSAGE LENGTH : var COMMAND SUBCODE :

DESCRIPTION : This message is sent to the ITS to request that a display be sent to the screen of the ITS.

MESSAGE NAME : Alert Message (Action Alert)

ORIGINATION : CCS DESTINATION : ITS

FUNCTION TYPE : 5 FUNCTION CODE : 1

MESSAGE LENGTH : COMMAND SUBCODE : 2

DESCRIPTION : One of three Alert Messages. These messages are sent to the IT for display in the Alert Areas of the screen. Action alert messages are queued on the SPS and they are replaced by operator acknowledgement.

MESSAGE NAME	:	Alert Message (Information Alert)			
ORIGINATION	:	CCS	DESTINATION	:	ITS
FUNCTION TYPE	:	5	FUNCTION CODE	:	1
MESSAGE LENGTH	:	72 B	COMMAND SUBCODE	:	1
DESCRIPTION	:	Alert messages are of three types [Information Alert, Action Alert, and Action-Alert-with-Associated Data]. Alert Messages are sent to the ITS for display in the Alert Areas of the screen. The primary screen has display areas for two Information-Alerts. Information Alerts are queued on the ITS. Information Alert is replaced upon an Information Alert Timeout.			

MESSAGE NAME	:	Alert Messages (Action Alert With Associated Data)			
ORIGINATION	:	CCS	DESTINATION	:	ITS
FUNCTION TYPE	:	5	FUNCTION CODE	:	1
MESSAGE LENGTH	:		COMMAND SUBCODE	:	3
DESCRIPTION	:	Same for other alert messages.			

MESSAGE NAME	:	Background Display Request			
ORIGINATION	:	CCS	DESTINATION	:	ITS
FUNCTION TYPE	:	5	FUNCTION CODE	:	25
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	This message is sent to the ITS to request that a display be sent to the screen of the ITS.			

MESSAGE NAME	:	Coordinated Universal Time :- Maintenance			
ORIGINATION	:	CCS	DESTINATION	:	ITS
FUNCTION TYPE	:	5	FUNCTION CODE	:	40
MESSAGE LENGTH	:	16	COMMAND SUBCODE	:	2
DESCRIPTION	:	This message is sent periodically at whatever time interval is needed to keep the ITS UTC in sync with the real UTC. It is also sent at startup/restart protocol in order to initialize UTC on the ITS. Unlike the transition UTC, the maintenance UTC has no associated message for the IT operator, hence the field H (message flag) is null.			

MESSAGE NAME	:	Coordinated Universal Time :- Transition			
ORIGINATION	:	CCS	DESTINATION	:	ITS
FUNCTION TYPE	:	5	FUNCTION CODE	:	40
MESSAGE LENGTH	:	16 B	COMMAND SUBCODE	:	1
DESCRIPTION	:	This message is sent by the CCS to all ITS nodes on the LAN. This will be done periodically at whatever time interval is needed to keep the ITS UTC in sync with the real UTC. It is also sent as part of the start up/restart protocol in order to initialize UTC on the ITS. This message expects the IT to display to the operator the message indicated by field H (message flag where 1 = Display 'Ready For Logon' message and 2 = Display 'CCs Available' message).			

MESSAGE NAME	:	Display Allowable Console Position :- Bit Map			
ORIGINATION	:	CCS	DESTINATION	:	ITS
FUNCTION TYPE	:	5	FUNCTION CODE	:	55
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	This message specifies which logged on positions are allowed access to each display.			

MESSAGE NAME	:	Display Data Return ASCII			
ORIGINATION	:	ITS	DESTINATION	:	CCS
FUNCTION TYPE	:	5	FUNCTION CODE	:	27
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	This message is sent to the CCS in response to data entries made in a display by the operator.			

MESSAGE NAME	:	Display Data Return In Error ASCII			
ORIGINATION	:	CCS	DESTINATION	:	ITS
FUNCTION TYPE	:	5	FUNCTION CODE	:	12
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	This message is sent from the CCS to the ITS in response to an erroneous Display Data Return ASCII message. A bit in the error bit map is for the position of the prompt text on the screen as defined by the Display Template.			

MESSAGE NAME	:	Display Data Send ASCII For Consecutive Dynamic Update.			
ORIGINATION	:	CCS	DESTINATION	:	ITS
FUNCTION TYPE	:	5	FUNCTION CODE	:	29
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	This message is sent to the ITS to request that a display be sent to the screen of the ITS.			

MESSAGE NAME	:	Display Data Send ASCII For Initial Dynamic Display.			
ORIGINATION	:	CCS	DESTINATION	:	ITS
FUNCTION TYPE	:	5	FUNCTION CODE	:	28
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	This message is sent to the ITS to request that a display be sent to the screen of the ITS.			

MESSAGE NAME	:	Display Data Send ASCII Message For New Display			
ORIGINATION	:	CCS	DESTINATION	:	ITS
FUNCTION TYPE	:	5	FUNCTION CODE	:	3
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	This message is sent to the ITS to request that a display be sent to the screen of the ITS. For displays that need foreground data, the application will send that data as part of this message. The format of the data area of the message is dependent on the display number.			

MESSAGE NAME	:	Freeze Dynamic Updates Command			
ORIGINATION	:	CCS	DESTINATION	:	ITS
FUNCTION TYPE	:	5	FUNCTION CODE	:	31
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	

DESCRIPTION : This message is sent to the ITS to request that a display be sent to the screen of the ITS.

MESSAGE NAME	:	Host IT Transition Control Message Down			
ORIGINATION	:	CCS	DESTINATION	:	ITS
FUNCTION TYPE	:	5	FUNCTION CODE	:	60
MESSAGE LENGTH	:	12 B	COMMAND SUBCODE	:	2

DESCRIPTION : This message is sent as a result of the CCS receiving an "Open Success" indication for an attempted LAN connection. The LAN Configuration Message specifies the function (prime or backup) of the LAN pathways from the CCS computer to/from the ITS computer.

MESSAGE NAME	:	Host IT Transition Control :- LAN Configuration Message.			
ORIGINATION	:	CCS	DESTINATION	:	ITS
FUNCTION TYPE	:	5	FUNCTION CODE	:	60
MESSAGE LENGTH	:	12 B	COMMAND SUBCODE	:	1

DESCRIPTION : This message is sent as a result of the CCS receiving an "Open Success" indication for an attempted LAN connection. The LAN Configuration Message specifies the function (prime or backup) of the LAN pathways from the CCS computer to/from the ITS computer.

MESSAGE NAME	:	Host IT Transition Control :- Template Error			
ORIGINATION	:	CCS	DESTINATION	:	ITS
FUNCTION TYPE	:	5	FUNCTION CODE	:	60
MESSAGE LENGTH	:	12 B	COMMAND SUBCODE	:	3

DESCRIPTION : This message is sent as a result of the CCS receiving an "Open Success" indication for an attempted LAN connection. The LAN Configuration Message specifies the function (prime or backup) of the LAN pathways from the CCS computer to/from the ITS computer.

MESSAGE NAME	:	Logoff Accepted			
ORIGINATION	:	CCS	DESTINATION	:	ITS
FUNCTION TYPE	:	5	FUNCTION CODE	:	41
MESSAGE LENGTH	:	4 B	COMMAND SUBCODE	:	51

DESCRIPTION : This message is sent to the ITS as a result of a valid logoff by the console operator.

MESSAGE NAME	:	Logon Accepted			
ORIGINATION	:	CCS	DESTINATION	:	ITS
FUNCTION TYPE	:	5	FUNCTION CODE	:	41
MESSAGE LENGTH	:	1120 B	COMMAND SUBCODE	:	50

DESCRIPTION : This message is sent to the ITS to signal a valid logon by the console operator. This message gives to the ITS the list of default rapid access displays for the positions. This message also sends to the ITS the password sequence number and a figure.

MESSAGE NAME	:	Pending Alert Display			
ORIGINATION	:	CCS	DESTINATION	:	ITS
FUNCTION TYPE	:	5	FUNCTION CODE	:	4
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	This message is sent to the ITS to request that a display be sent to the screen of the ITS.			

MESSAGE NAME	:	Service Message To It			
ORIGINATION	:	CCS	DESTINATION	:	ITS
FUNCTION TYPE	:	5	FUNCTION CODE	:	2
MESSAGE LENGTH	:	22	COMMAND SUBCODE	:	0
DESCRIPTION	:	Service Messages are sent to the ITS for display in the Service Message Area of the screen. Numbered service messages are possible, but the sender may optionally send self-generated service text.			

MESSAGE NAME	:	Template Compare Command			
ORIGINATION	:	CCS	DESTINATION	:	ITS
FUNCTION TYPE	:	5	FUNCTION CODE	:	52
MESSAGE LENGTH	:		COMMAND SUBCODE	:	1
DESCRIPTION	:	This message contains the date of the last time the Template TIP files for the current configuration level were modified. The ITS is expected to compare this date with the date saved from the last time the Template Compare Request was received. If the dates do not match, the Display Directory Message will contain compilation time information associated with each displays template object currently stored on the ITS.			

MESSAGE NAME	:	Template Objects			
ORIGINATION	:	CCS	DESTINATION	:	ITS
FUNCTION TYPE	:	5	FUNCTION CODE	:	54
MESSAGE LENGTH	:		COMMAND SUBCODE	:	
DESCRIPTION	:	A Template Object Message is sent to the ITS by the CCS for each display in the 11 display directory that is not consistent with the CCS Display Directory. This message contains the templates that are used by the ITS in generating displays and managing data entries.			

MESSAGE NAME	:	Terminate Dynamic Display Command			
ORIGINATION	:	CCS	DESTINATION	:	ITS
FUNCTION TYPE	:	5	FUNCTION CODE	:	30
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	This message is sent to the ITS to request that a display be sent to the screen of the ITS.			

MESSAGE NAME	:	Unfreeze Dynamic Updates Command			
ORIGINATION	:	CCS	DESTINATION	:	ITS
FUNCTION TYPE	:	5	FUNCTION CODE	:	32
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	This message is sent to the ITS to request that a display be sent to the screen of the ITS.			

MESSAGE NAME	:	Action Alert Acknowledgement from IT Operator		
ORIGINATION	:	ITS	DESTINATION :	CCS
FUNCTION TYPE	:	5	FUNCTION CODE :	6
MESSAGE LENGTH	:	var	COMMAND SUBCODE :	
DESCRIPTION	:	This message is sent by the ITS to the CCS to acknowledge a Display Data Send ASCII Message.		

MESSAGE NAME	:	Display Directory		
ORIGINATION	:	ITS	DESTINATION :	CCS
FUNCTION TYPE	:	5	FUNCTION CODE :	53
MESSAGE LENGTH	:	var	COMMAND SUBCODE :	
DESCRIPTION	:	This message is sent by the ITS to the CCS in response to the ITS receiving a Template Compare Request message from the SPS. It contains a return status flag indicating if the Template Configuration Date as contained in the Template Compare Request matched on the ITS and CCS. If no match the display Directory Message contains a list of displays in use on the ITS and compilation dates for each display. The CCS uses these dates to determine if new Template Objects should be sent.		

MESSAGE NAME	:	IT Host Transition Control :- Deactivate		
ORIGINATION	:	ITS	DESTINATION :	CCS
FUNCTION TYPE	:	5	FUNCTION CODE :	61
MESSAGE LENGTH	:		COMMAND SUBCODE :	2
DESCRIPTION	:	This message is sent by the ITS to the CCS in response to a request in the LAN Configuration Message. The Profile Message contains the logon state relative to each screen.		

MESSAGE NAME	:	IT Host Transition Control - Profile			
ORIGINATION	:	ITS	DESTINATION	:	CCS
FUNCTION TYPE	:	5	FUNCTION CODE	:	61
MESSAGE LENGTH	:		COMMAND SUBCODE	:	1
DESCRIPTION	:	This message is sent by the ITS to the CCS in response to a request in the LAN Configuration Message. The Profile Message contains the logon state relative to each screen.			

MESSAGE NAME	:	Loop Test :- Life Test			
ORIGINATION	:	ITS	DESTINATION	:	CCS
FUNCTION TYPE	:	5	FUNCTION CODE	:	62
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	1
DESCRIPTION	:	This message is sent in response to a request in the LAN Configuration Message.			

MESSAGE NAME	:	Loop Test :- Life Test Response			
ORIGINATION	:	ITS	DESTINATION	:	CCS
FUNCTION TYPE	:	5	FUNCTION CODE	:	62
MESSAGE LENGTH	:		COMMAND SUBCODE	:	2
DESCRIPTION	:				

MESSAGE NAME	:	Pending Action Alerts Display Request			
ORIGINATION	:	ITS	DESTINATION	:	CCS
FUNCTION TYPE	:	5	FUNCTION CODE	:	22
MESSAGE LENGTH	:		COMMAND SUBCODE	:	
DESCRIPTION	:	This message is sent by the ITS to the CCS.			

INTERNAL MESSAGES BETWEEN THE SPS AND CCS

MESSAGE NAME	:	Authorized User IDs/Passwords		
ORIGINATION	:	SPS	DESTINATION :	CCS
FUNCTION TYPE	:	7	FUNCTION CODE :	20
MESSAGE LENGTH	:	var	COMMAND SUBCODE :	
DESCRIPTION	:	This message is sent to the CCS to transfer all authorized user IDs and Passwords.		

MESSAGE NAME	:	Current Site Status Response		
ORIGINATION	:	SPS	DESTINATION :	CCS
FUNCTION TYPE	:	7	FUNCTION CODE :	1
MESSAGE LENGTH	:	0 B	COMMAND SUBCODE :	3
DESCRIPTION	:	This message is sent from the SPS to the CCS during communication synchronization. It contains an acknowledgement of the previous CCS Site Status Message.		

MESSAGE NAME	:	Display Directory Message		
ORIGINATION	:	SPS	DESTINATION :	CCS
FUNCTION TYPE	:	7	FUNCTION CODE :	46
MESSAGE LENGTH	:	var	COMMAND SUBCODE :	na
DESCRIPTION	:	This message is sent to the CCS as a result of the SPS receiving a Display Directory Request Message from the CCS. It contains a list of displays in use on the SPS and compilation dates for each display. The CCS will use the compilation dates to determine which new template objects should be requested.		

MESSAGE NAME	:	Event And Service Information Message			
ORIGINATION	:	SPS	DESTINATION	:	CCS
FUNCTION TYPE	:	7	FUNCTION CODE	:	30
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	The detailed event portion of this message requires modifications to compensate for the change from the 36 bit U1100 to the 32 bit VAX.			

MESSAGE NAME	:	Event Termination Message From SSQ4 to EMQ8			
ORIGINATION	:	SPS	DESTINATION	:	CCS
FUNCTION TYPE	:	7	FUNCTION CODE	:	31
MESSAGE LENGTH	:	44 B	COMMAND SUBCODE	:	0
DESCRIPTION	:	This message from SSQ4 to EMQ8 signals the termination of an event.			

MESSAGE NAME	:	Service Parameter Message			
ORIGINATION	:	SPS	DESTINATION	:	CCS
FUNCTION TYPE	:	7	FUNCTION CODE	:	24
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	This message transfers service parameter data from the SPS to the CCS. If the parameter values for a given service type and parameter type do not change for subsequent spacecraft the verification method for the spacecraft will be set as "same".			

MESSAGE NAME	:	SPS Application Routing Information			
ORIGINATION	:	SPS	DESTINATION	:	CCS
FUNCTION TYPE	:	7	FUNCTION CODE	:	1
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	4

DESCRIPTION : This message is sent from the SPS to the CCS during communication synchronization.

MESSAGE NAME	:	SPS System Configuration			
ORIGINATION	:	SPS	DESTINATION	:	CCS
FUNCTION TYPE	:	7	FUNCTION CODE	:	1
MESSAGE LENGTH	:	12 B	COMMAND SUBCODE	:	1

DESCRIPTION : This message is sent to the CCS during Communication synchronization. This message contains the SPS System Level (Operational, Test, Development), SPS Role Configuration (Prime, Backup) and SPS Software Execution Level, which specifies the data base to use.

MESSAGE NAME	:	SPS System Parameter Transfer Message			
ORIGINATION	:	SPS	DESTINATION	:	CCS
FUNCTION TYPE	:	7	FUNCTION CODE	:	1
MESSAGE LENGTH	:	0 B	COMMAND SUBCODE	:	2

DESCRIPTION : This message is sent from the SPS to the CCS during communication synchronization. It contains an acknowledgement to the previous CCS System Parameter Transfer Message.

MESSAGE NAME	:	SPS-IT Logon/Logoff Status			
ORIGINATION	:	SPS	DESTINATION	:	CCS
FUNCTION TYPE	:	7	FUNCTION CODE	:	1
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	5
DESCRIPTION	:	This message is sent to the CCS during communication synchronization. It contains all the It Logon/Logoff Status as SPS views it.			

MESSAGE NAME	:	Static Data Transfer Message			
ORIGINATION	:	SPS	DESTINATION	:	CCS
FUNCTION TYPE	:	7	FUNCTION CODE	:	23
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	This message is sent by the SPS to the CCS to request transfer of static data to CCS.			

MESSAGE NAME	:	Template Compare Message			
ORIGINATION	:	SPS	DESTINATION	:	CCS
FUNCTION TYPE	:	7	FUNCTION CODE	:	45
MESSAGE LENGTH	:	12 int	COMMAND SUBCODE	:	1
DESCRIPTION	:	This message contains the data of the last time the template TIP files for the current configuration level were modified. If this date does not match with other data at the CCS, the CCS sends a Display Directory Request to continue the synchronization process.			

MESSAGE NAME	:	Template Object Message			
ORIGINATION	:	SPS	DESTINATION	:	CCS
FUNCTION TYPE	:	7	FUNCTION CODE	:	47
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	NA
DESCRIPTION	:	This message is sent by the SPS to the CCS for each display in the CCS display that is not current with the SPS Display Directory. It contains any one of the templates that are used by the CCS supporting ITS displays and display data entries.			

MESSAGE NAME	:	Valid SICs and Spacecraft Names			
ORIGINATION	:	SPS	DESTINATION	:	CCS
FUNCTION TYPE	:	7	FUNCTION CODE	:	21
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	These messages transfer valid SICs and Spacecraft names from the SPS to the CCS.			

MESSAGE NAME	:	Valid TDRS - ID			
ORIGINATION	:	SPS	DESTINATION	:	CCS
FUNCTION TYPE	:	7	FUNCTION CODE	:	22
MESSAGE LENGTH	:	40 B	COMMAND SUBCODE	:	
DESCRIPTION	:	This message transfers valid SICs and Spacecraft names from the SPS to the CCS.			

MESSAGE NAME	:	CCS Application Routing Information		
ORIGINATION	:	CCS	DESTINATION :	SPS
FUNCTION TYPE	:	6	FUNCTION CODE :	1
MESSAGE LENGTH	:	var	COMMAND SUBCODE :	4
DESCRIPTION	:	This message is sent from the CCS to the SPS during communication synchronization. It contains the CCS Application Routing Information.		

MESSAGE NAME	:	CCS System Configuration		
ORIGINATION	:	CCS	DESTINATION :	SPS
FUNCTION TYPE	:	6	FUNCTION CODE :	1
MESSAGE LENGTH	:	12	COMMAND SUBCODE :	1
DESCRIPTION	:	This is the first message sent to the SPS during communication synchronization. This message contains the CCS System Level (Operational, Test, Development) and CCS Role Configuration (Prime, Backup).		

MESSAGE NAME	:	CCS System Parameters Transfer		
ORIGINATION	:	CCS	DESTINATION :	SPS
FUNCTION TYPE	:	6	FUNCTION CODE :	1
MESSAGE LENGTH	:	var	COMMAND SUBCODE :	2
DESCRIPTION	:	This message is sent from the CCS to the SPS during communication synchronization. It contains the CCS to SPS I am alive interval and the ITS connection addresses.		

MESSAGE NAME	:	CCS-IT Logon/Logoff Status			
ORIGINATION	:	CCS	DESTINATION	:	SPS
FUNCTION TYPE	:	6	FUNCTION CODE	:	1
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	5

DESCRIPTION : This message is sent from the CCS to the SPS during communication synchronization. It contains the ITS Logon/Logoff Status as the CCS views it.

MESSAGE NAME	:	Current Site Table Transfer			
ORIGINATION	:	CCS	DESTINATION	:	SPS
FUNCTION TYPE	:	6	FUNCTION CODE	:	1
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	3

DESCRIPTION : This message is sent from CCS to the SPS during communication synchronization. It contains all the current site status.

MESSAGE NAME	:	Display Directory Request Message			
ORIGINATION	:	CCS	DESTINATION	:	SPS
FUNCTION TYPE	:	6	FUNCTION CODE	:	46
MESSAGE LENGTH	:	0 B	COMMAND SUBCODE	:	NA

DESCRIPTION : This message is sent by the CCS to the SPS if the compilation dates for the templates in the two segments do not match. It signals SPS to return the Display Directory Message.

MESSAGE NAME	:	Template Compare Request Message			
ORIGINATION	:	CCS	DESTINATION	:	SPS
FUNCTION TYPE	:	6	FUNCTION CODE	:	45
MESSAGE LENGTH	:	0 B	COMMAND SUBCODE	:	1

DESCRIPTION : This message is sent from the CCS to the SPS at the beginning of the template synchronization process. It signals SPS to return the Template Compare Message containing the compilation dates for the templates on SPS.

MESSAGE NAME	:	Template Object Request			
ORIGINATION	:	CCS	DESTINATION	:	SPS
FUNCTION TYPE	:	6	FUNCTION CODE	:	47
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	NA

DESCRIPTION : This message is sent by the CCS to the SPS to request template objects for which the compilation dates do not match. The request contains the templates needed by the template number.

INTERNAL MESSAGES BETWEEN THE SPS AND ITS

This interface is through a Local Area Network (LAN) to which the SPS and each Intelligent terminal is connected. Each message passed between the ITS and the SPS are uniquely defined by a combination of the NCC Function Type, the NCC Command / Function Code, and the NCC Command Subcode.

MESSAGE NAME	:	Alert Additional Data Display			
ORIGINATION	:	SPS	DESTINATION	:	ITS
FUNCTION TYPE	:	1	FUNCTION CODE	:	5
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	This message is sent to the ITS to request that a display be sent to the screen of the ITS.			

MESSAGE NAME	:	Alert Message -Action Alert With Data			
ORIGINATION	:	SPS	DESTINATION	:	ITS
FUNCTION TYPE	:	1	FUNCTION CODE	:	1
MESSAGE LENGTH	:	72 B	COMMAND SUBCODE	:	3
DESCRIPTION	:	Same as Information Alert.			

MESSAGE NAME	:	Alert Message To IT - Action Alert			
ORIGINATION	:	SPS	DESTINATION	:	ITS
FUNCTION TYPE	:	1	FUNCTION CODE	:	1
MESSAGE LENGTH	:	72 B	COMMAND SUBCODE	:	2
DESCRIPTION	:	Same as Information Alert.			

MESSAGE NAME	:	Alert Message to IT - Information Alert		
ORIGINATION	:	SPS	DESTINATION :	ITS
FUNCTION TYPE	:	1	FUNCTION CODE :	1
MESSAGE LENGTH	:	72 B	COMMAND SUBCODE :	1
DESCRIPTION	:	Alert messages are of three types [Information Alert, Action Alert and Action Alert with Associated Data]. Alert Messages are sent to the ITS for display in the Alert Areas of the screen. The primary screen has display areas for one Action Alert and two Information Alerts. Action Alerts are queued on the SPS and Information Alerts are queued on the ITS. An Action Alert is replaced upon operator acknowledgement, and an Information Alert is replaced upon an Information Alert Timeout.		

MESSAGE NAME	:	Background Display Request		
ORIGINATION	:	SPS	DESTINATION :	ITS
FUNCTION TYPE	:	1	FUNCTION CODE :	25
MESSAGE LENGTH	:	var	COMMAND SUBCODE :	
DESCRIPTION	:	This message is sent to the ITS to request that a display be sent to the screen of the ITS.		

MESSAGE NAME	:	Coordinated Universal Time : Maintenance		
ORIGINATION	:	SPS	DESTINATION :	ITS
FUNCTION TYPE	:	1	FUNCTION CODE :	40
MESSAGE LENGTH	:	16 B	COMMAND SUBCODE :	2
DESCRIPTION	:	This message is sent periodically at whatever time interval is needed to keep the ITS UTC in sync with the real UTC. It is also sent at startup/restart protocol in order to initialize UTC on the ITS.		

MESSAGE NAME	:	Coordinated Universal Time : Transition			
ORIGINATION	:	SPS	DESTINATION	:	ITS
FUNCTION TYPE	:	1	FUNCTION CODE	:	40
MESSAGE LENGTH	:	16 B	COMMAND SUBCODE	:	2
DESCRIPTION	:	This message is sent periodically to all ITS nodes on the LAN at whatever time interval is needed to keep the ITS UTC in sync with the real UTC. It is also sent at startup/restart protocol in order to initialize UTC on the ITS.			

MESSAGE NAME	:	Display Allowable Console Posit Bit Map			
ORIGINATION	:	SPS	DESTINATION	:	ITS
FUNCTION TYPE	:	1	FUNCTION CODE	:	55
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	This message specifies which logged-on positions are allowed access to each display.			

MESSAGE NAME	:	Display Data Return In Error ASCII			
ORIGINATION	:	SPS	DESTINATION	:	ITS
FUNCTION TYPE	:	1	FUNCTION CODE	:	12
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	This message is sent from the SPS to the ITS in response to an erroneous Display Data Return ASCII Message. A bit in the Error Bit Map is set for the position of the prompt text on the screen as defined by the display template.			

MESSAGE NAME	:	Display Data Send ASCII for Consecutive Dynamic Update			
ORIGINATION	:	SPS	DESTINATION	:	ITS
FUNCTION TYPE	:	1	FUNCTION CODE	:	29
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	This message is sent to the ITS to request that a display be sent to the screen of the ITS.			

MESSAGE NAME	:	Display Data Send ASCII-Initial Dynamic Display			
ORIGINATION	:	SPS	DESTINATION	:	ITS
FUNCTION TYPE	:	1	FUNCTION CODE	:	28
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	This message is sent to the ITS to request that a display be sent to the screen of the ITS. For displays that need foreground data, the application will send that data as part of the message. Format of data for message is dependent on display number.			

MESSAGE NAME	:	Display Data Send-ASCII for new display			
ORIGINATION	:	SPS	DESTINATION	:	ITS
FUNCTION TYPE	:	1	FUNCTION CODE	:	3
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	This message is sent to the ITS to request that a display be sent to the screen of the ITS. For displays that need foreground data, the application will send that data as part of this message. The format of the data area of the message is dependent on the display number.			

MESSAGE NAME	:	Freeze Dynamic Updates Command			
ORIGINATION	:	SPS	DESTINATION	:	IT
FUNCTION TYPE	:	1	FUNCTION CODE	:	31
MESSAGE LENGTH	:	Var	COMMAND SUBCODE	:	

DESCRIPTION : This message is sent to the ITS to request that a display be sent to the screen of the ITS.

MESSAGE NAME	:	Host-IT Transition Control : Down			
ORIGINATION	:	SPS	DESTINATION	:	ITS
FUNCTION TYPE	:	1	FUNCTION CODE	:	60
MESSAGE LENGTH	:	12 B	COMMAND SUBCODE	:	2

DESCRIPTION :

MESSAGE NAME	:	Host-IT-Transition-Control LAN Configuration.			
ORIGINATION	:	SPS	DESTINATION	:	ITS
FUNCTION TYPE	:	1	FUNCTION CODE	:	60
MESSAGE LENGTH	:	12 B	COMMAND SUBCODE	:	1

DESCRIPTION : This message is sent as a result of the SPS receiving an "Open Success" indication for an attempted LAN connection. The LAN Configuration Message specifies the function (prime or backup) of the LAN pathways from the SPS computer to/from the ITS computer.

MESSAGE NAME	:	Host-IT-Transition-Control :Template Error			
ORIGINATION	:	SPS	DESTINATION	:	ITS
FUNCTION TYPE	:	1	FUNCTION CODE	:	60
MESSAGE LENGTH	:	12 B	COMMAND SUBCODE	:	3
DESCRIPTION	:				

MESSAGE NAME	:	Logoff Accepted			
ORIGINATION	:	SPS	DESTINATION	:	ITS
FUNCTION TYPE	:	1	FUNCTION CODE	:	41
MESSAGE LENGTH	:	4 B	COMMAND SUBCODE	:	51
DESCRIPTION	:	This message is sent to the ITS as a result of a valid logoff by the console operator.			

MESSAGE NAME	:	Logon Accept			
ORIGINATION	:	SPS	DESTINATION	:	IT
FUNCTION TYPE	:	1	FUNCTION CODE	:	41
MESSAGE LENGTH	:	1120 B	COMMAND SUBCODE	:	50
DESCRIPTION	:	This message is sent to the ITS to signal a valid logon by the console operator. This message gives to the ITS the list of default rapid access displays for the position. This message also sends to the ITS the password sequence number and the logon position that must be included in any profile message.			

MESSAGE NAME	:	Loop Test - Life Test Response			
ORIGINATION	:	SPS	DESTINATION	:	IT
FUNCTION TYPE	:	1	FUNCTION CODE	:	62
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	2
DESCRIPTION	:	This message is sent in response to a request in the LAN Configuration Message.			

MESSAGE NAME	:	Loop Test : Life Test			
ORIGINATION	:	SPS	DESTINATION	:	ITS
FUNCTION TYPE	:	1	FUNCTION CODE	:	62
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	1
DESCRIPTION	:	This message is sent in response to a request in the LAN Configuration Message.			

MESSAGE NAME	:	Pending Alerts Display			
ORIGINATION	:	SPS	DESTINATION	:	ITS
FUNCTION TYPE	:	1	FUNCTION CODE	:	4
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	This message is sent to the ITS to request that a display be sent to the screen of the ITS.			

MESSAGE NAME	:	Service Message To IT-Text Included			
ORIGINATION	:	SPS	DESTINATION	:	ITS
FUNCTION TYPE	:	1	FUNCTION CODE	:	2
MESSAGE LENGTH	:	72 B	COMMAND SUBCODE	:	0

DESCRIPTION : Service Messages are sent to the ITS for display in the Service Message Area of the screen. Numbered service messages are possible, but the sender may optionally send self-generated service text.

MESSAGE NAME	:	Template Compare Command			
ORIGINATION	:	SPS	DESTINATION	:	IT
FUNCTION TYPE	:	1	FUNCTION CODE	:	52
MESSAGE LENGTH	:		COMMAND SUBCODE	:	1

DESCRIPTION : This message contains the date of the last time the Template TIP files for the current configuration level were modified. The ITS is expected to compare this date with the date saved from the last time the Template Compare Message was received. If the dates do not match, the Display Directory Message will contain compilation time information associated with each displays template currently stored on the ITS.

MESSAGE NAME	:	Template Object			
ORIGINATION	:	SPS	DESTINATION	:	ITS
FUNCTION TYPE	:	1	FUNCTION CODE	:	54
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	

DESCRIPTION : A Template Object Message is sent to the ITS by the SPS for each display in the 11 display directory that is not consistent worth the SPS Display Directory. This message contains the templates that are used by the IT in generating displays and managing data entries.

MESSAGE NAME	:	Terminate Dynamic Display Commands			
ORIGINATION	:	SPS	DESTINATION	:	ITS
FUNCTION TYPE	:	1	FUNCTION CODE	:	30
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	This message is sent to the ITS to request that a display be sent to the screen of the ITS.			

MESSAGE NAME	:	Unfreeze Dynamic Updates Command			
ORIGINATION	:	SPS	DESTINATION	:	ITS
FUNCTION TYPE	:	1	FUNCTION CODE	:	32
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	This message is sent to the ITS to request that a display be sent to the screen of the ITS.			

MESSAGE NAME	:	Action Alert Acknowledgement From IT Operator			
ORIGINATION	:	ITS	DESTINATION	:	SPS
FUNCTION TYPE	:	1	FUNCTION CODE	:	6
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	
DESCRIPTION	:	This message is sent by the ITS to the SPS to acknowledge a Display Data Send ASCII Message.			

MESSAGE NAME	:	Display Data Return ASCII (ie operator data entries.)			
ORIGINATION	:	ITS	DESTINATION	:	SPS
FUNCTION TYPE	:	1	FUNCTION CODE	:	27
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	

DESCRIPTION : This message is sent to the SPS in response to data entries made in a display by the operator. It contains those entries plus information associated with particular entries for any non-mandatory data entry fields. In addition to being sent as a result of the operator indicating an end of data display this message may be saved by the ITS and sent again in response to the Prior Display and Rapid Access Retrieve Command.

MESSAGE NAME	:	Display Directory			
ORIGINATION	:	ITS	DESTINATION	:	SPS
FUNCTION TYPE	:	1	FUNCTION CODE	:	53
MESSAGE LENGTH	:	var	COMMAND SUBCODE	:	

DESCRIPTION : This message is sent as a result of the ITS receiving a Template Compare Request Message. This message contains return status flag indicating if the date in the request matched on the SPS and ITS. If no match occurs this message contains a list of displays in use on the ITS and the compilation date for each display. The SPS will use the compilation dates to determine if new template objects should be sent to the ITS.

MESSAGE NAME	:	IT-Host-Transition Control : Deactivate		
ORIGINATION	:	ITS	DESTINATION :	SPS
FUNCTION TYPE	:	1	FUNCTION CODE :	61
MESSAGE LENGTH	:	var	COMMAND SUBCODE :	2
DESCRIPTION	:	Sent in response to a request in the LAN Configuration.		

MESSAGE NAME	:	IT-Host-Transition Control : Profile		
ORIGINATION	:	ITS	DESTINATION :	SPS
FUNCTION TYPE	:	1	FUNCTION CODE :	61
MESSAGE LENGTH	:	var	COMMAND SUBCODE :	1
DESCRIPTION	:	This message is sent by the ITS to the SPS in response to a request in the LAN Configuration Message. The Profile Message contains the logon state relative to each screen.		

MESSAGE NAME	:	Pending Action Alerts Display Request		
ORIGINATION	:	ITS	DESTINATION :	SPS
FUNCTION TYPE	:	1	FUNCTION CODE :	22
MESSAGE LENGTH	:	var	COMMAND SUBCODE :	
DESCRIPTION	:	This message is sent by the ITS to the SPS.		

INDEX

10. References

[Law & Kelton] Law.A.M. and Kelton.D.W, Simulation Modeling & Analysis, McGraw Hill Inc. New York, 1991.

[CSC/SD-85/6709] NCC System Specification, Vol4, Sept. 1989

[CSEC] Center for Systems Engineering and Computing, Simulating the Operations of the Network Control Center, 1992

[Jain] Jain Raj, The Art of Computer Systems Performance Analysis, John Wiley & Sons, Inc, 1991

[Khoshnevis] Khoshnevis.B. Discrete Systems Simulation, McGraw Hill Inc. New York, 1994.

[NASA 3124] Space Network Control Conference on Resource Allocation Concepts and Approaches, 1990

[ND, 6/28/90] ND, 6/28/90

[MacDouglas] MacDouglas M.H, Simulating Computer Systems Techniques and Tools, MIT Press, 1987

[Welker and Horne 1960] Welker.E.L and Horne.R.C., Concepts Associated with System Effectiveness, ARINC Research Monograph No. 9 (July 1960).

[STDN 101.2] Network Users Guide, 9/1988

[STDN 101.8] STDN 101.8

[STDN 106] STDN Operations Concept

[STDN 203.13] Functional and Performance Requirements for TDRSS

[STDN 203.6] NCC Functional and Performance Requirements for NCC

[STDN 220.5] NCC/POCC Interface Control Document

[STDN 907] NCC Data System BaselineProject Plan, April 1988

[STDN 1151.2] TDRSS Orientation and System Data Flow Coursse 820

[STDN 1153.4] TDRSS Constraints Scheduling Procedures Course 882

[Stallings1] Stallings, W Handbook of Computer Communications Standards, Vol 1 Macmillan Publishing, New York (1988)

[Stallings2] Stallings, W Handbook of Computer Communications Standards, Vol 2 Macmillan Publishing, New York (1988)

[Stallings3] Stallings, W Handbook of Computer Communications Standards, Vol 3 Macmillan Publishing, New York (1988)

[Elperin] Elperin ,T Gertsbakh, I, Lomonosov,M Estimation of Network Reliability Using Graph Evolution Models, IEEE Transactions on Reliability, Vol 40 No. 5 Dec 1991, pp 572-581

[Wilson] Wilson, R Introduction to Graph Theory, Longman Inc, New York, 1985

[Engelhardt] Engelhardt, B ,Introduction to Probability and Statistics, Duxbury Press, Belmont, California, 1992

[Schwartz] Schwartz, M. Computer Communication Network Design and Analysis, Prentice-Hall, INC, Englewood Cliffs, New Jersey, 1977