Configurable Spacecraft Control Architectures for On-orbit Servicing and Upgrading of Long Life Orbital Platforms

James Ross[1], David Musliner[2], Thomas Kreider[1], Jack Jacobs[1], Michael Fisher[3]

[1]Honeywell DSES-Glendale,19019 N.59th Ave, Glendale,AZ 85308 U.S.A.
{james.a.ross,thom.Kreider,jack.h.jacobs}@honeywell.com
[2]Honeywell Laboratories,3660 Technology Drive, Minneapolis,MN 55418 U.S.A.
david.musliner@honeywell.com
[3]KinetX, Inc., 2141 E. Broadway Rd., Suite 217, Tempe,AZ 85282 U.S.A
mfisher@kinetx.com

Abstract – Large orbital platforms provide unique and essential space-based capabilities for science, intelligence, and defense missions potentially supporting very large aperture imagers, antenna farms, SARs, radiometers and other systems. In order to provide maximum return on the investment required, it is essential to have a significant autonomous on-orbit servicing, upgrade and repair capability such that the platform can operate successfully for decades and have new capabilities added to it. The dependence on human upgrades on-orbit must diminish as we go farther in space to minimize the risk to life. In order for such autonomous operations to be practical, current technologies from robotics, distributed computing, and spacecraft operation need to be integrated and demonstrated. We are exploring the application of collaborative control strategies designed for multi-agent critical systems[1], as well as network and distributed agent computing to demonstrate some of the basic functionality needed to coordinate tasks autonomously in a dynamic environment.

TABLE OF CONTENTS

1 Introduction
2. An Orbital Platform Resupply Scenario
3. Cooperative Real-Time Control Architecture (CIRCA)
4. Autonomous Communication and Coordination Between Independent Modules (Agents)
5. Developing a Taxonomy for Localized Versus Global Planning and Scheduling
6. Conclusions
7. References
8. Biography

1. INTRODUCTION

On-orbit servicing, upgrading, and repair tasks require the largely autonomous interaction of cargo and transfer vehicles, the main platform, its docking interfaces and any free flying local area servicing spacecraft. In an analogy to terrestial applications, it is the equivalent of running an airport with all the inherent issues of cooperative resource planning and scheduling, local area navigation and maneuvering, and resource replenishment and repair. In such a distributed on-orbit system, it is important to have a control architecture that incorporates run-time cooperation between agents which may be transfer vehicles , payload modules, servicesats or orbital managers.

This paper will address some of the fundamental architecture issues in developing autonomous configurable space systems as well as discuss an approach using Cooperative Intelligent Real-Time Control Architecture (CIRCA) based planning control and autonomous orbital navigation and rendezvous. In the proposed architecture, each system is controlled by a CIRCA system running on a virtual machine[2] such that code can be exchanged between agents, even when running on disparate processor systems (which is likely in a long life platform).

2. AN ORBITAL PLATFORM RESUPPLY SCENARIO

In a distributed environment such as an orbital platform, multiple autonomous agents such as the orbital platform controller, various resupply vehicles and local maintenance vehicles are each operating to a local plan incorporating run-time cooperation to achieve team goals in mission critical domains. A critical problem from a systems controls perspective is the autonomous exchange of relevant task and mission information and arranging the necessary resources and manuevers to be assigned and executed. Since each agent is either operating autonomously, in a human guided role, or under direct operator control in a heterogeneous operating environment, there must be a common language to coordinate activities successfully.

Consider an example scenario where a orbital transfer cargo vehicle arrives at a geosynchronous platform with a fuel tank resupply and a new payload module (Fig. 1). To complete the resupply and payload upgrade, several tasks must be completed autonomously for the mission to be successful. In order for this to be accomplished, it is necessary to generate a "work order" which defines the tasks, phases and resources as well as new parameters or software to be uploaded on completion. Once all the resources are assigned, maneuvers must be planned and executed to move the fuel module and payload module into the proper locations while avoiding collisions with each other, removing any old elements as well as avoiding mobile elements in the environment (solar arrays, antennas, etc). Once the new modules move into their docked positions, the platform system must be updated to reflect the new mass properties, the new fuel load as well as the the payload capabilities and any associated processing algorithm updates. This must take place in an environment where processing hardware and operating systems may be different so a virtual runtime environment becomes an important element to consider.

In our scenario, the cargo vessel provides the platform with a Bill-Of Materials (BOM) which provides necessary information on the cargo that is being delivered (Fig. 2). The platform master planner task reads the BOM and proceeds to evaluate what to do with it:

- Check available docking connections
  - no available connections
- Check current fuel tank inventory and status
  - locate empty tank which can be replaced
  - reserve connection for new tank
- Assign new fuel tank to dock connection
- Generate task to remove empty tank
  - requires servicebot
  - requires destination orbit parameters
- Generate task to install new fuel tank
  - requires servicebot
  - must be completed in 24 hour.
- Publish task requests to "work board" for bidding
- Recheck task status to guarantee transaction can be completed

At this point, the sequence of tasks is broken down so that they can be accepted by potentially more than one servicebot. The tasks are linked though and must be completed in order for the overall task to be successful. Agents such as the servicebots will bid on the tasks and specify their capability to complete the tasks. Since both the removal and install tasks must be completed sequentially and within a time constraint, they can be grouped into a transaction. Tasks that are grouped into transactions must be guaranteed completion as a group or the tasks rebid. If the transaction cannot be guaranteed, resources may have to be preempted to allow the tasks to complete.

The install task that is generated contains sufficient information for the servicebot to evaluate its capability to perform the task within the constrains

```
Task Install
• Requires carrier_connect
• Requires transport:fuelcell
• Start task
# get tank from OTV and connect to OP socket 3
-     moveto OTV:socket 1
-     attach fuelcell:carrier_connect
-     disconnect fuelcell from OTV:socket 1
-     wait until clear OP:socket 3
-     moveto OP:socket 3
-     connect fuelcell:fuel_connect to OP:socket 3
-     update OP:status
      unattach fuelcell:carrier_connect
• Release task
```

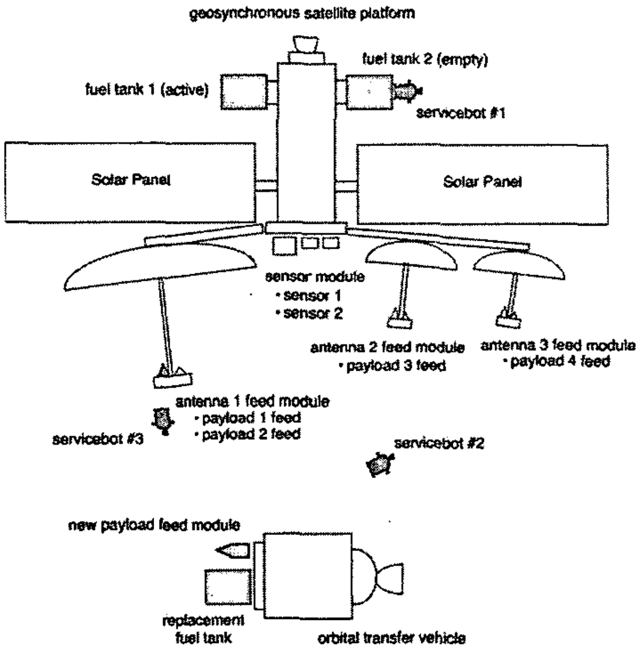
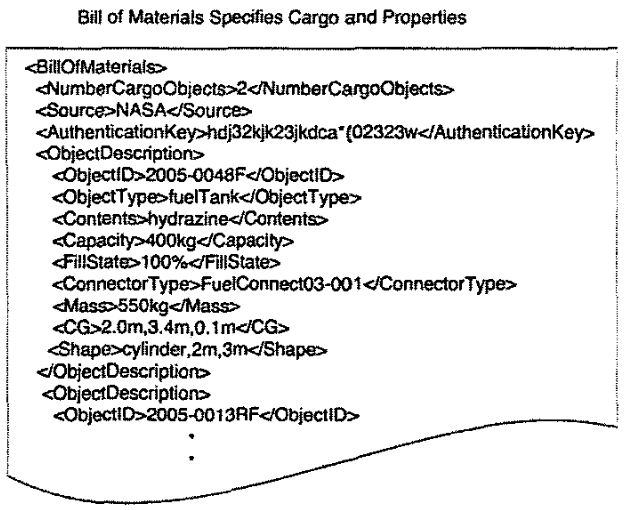


Figure 1. Example of a Orbital Platform Resupply Scenario

The servicebot must evaluate the task request to determine if it has the required resources (i.e. can it handle the fuel cell and does it have the proper connectors). It then must determine the time and resources to complete the manuevers required to get to the OTV, retrieve the fueltank and move to the platform socket and return to its dock. This requires determining relative orbital vectors for the maneuver start and end locations as well as the potential flight path maneuvers required to reach the destination location(Fig. 3).

The result of this evaluation is returned to the planner as part of the task "bid". The task planner then evaluates all "bids" to find the combination which can successfully complete the transaction within the specified constraints.


Bill of Materials Specifies Cargo and Properties

```
<BillOfMaterials>
  <NumberCargoObjects>2</NumberCargoObjects>
  <Source>NASA</Source>
  <AuthenticationKey>hdj32kjk23jkdca*(02323w</AuthenticationKey>
  <ObjectDescription>
    <ObjectID>2005-0048F</ObjectID>
    <ObjectType>fuelTank</ObjectType>
    <Contents>hydrazine</Contents>
    <Capacity>400kg</Capacity>
    <FillState>100%</FillState>
    <ConnectorType>FuelConnect03-001</ConnectorType>
    <Mass>550kg</Mass>
    <CG>2.0m,3.4m,0.1m</CG>
   <Shape>cylinder,2m,3m</Shape>
  </ObjectDescription>
   <ObjectDescription>
    <ObjectID>2005-0013RF</ObjectID>
                    .
                    .
```

Figure 2 - Example of Cargo Bill of Materials (BOM)

3. COOPERATIVE REAL-TIME CONTROL ARCHITECTURE (CIRCA)

A key element in our approach to coordinating the realtime planning by each of the agents is to use the Multi-Agent Self-Adaptive Cooperative Intelligent Real-Time Control Architecture (MASA-CIRCA) which has been developed for missions such as coordinated Unmanned Aerial Vehicle (UAV) operation[3]. MASA-CIRCA is a domain-independent architecture for intelligent, self-adaptive autonomous control systems that can be applied to hard real-time, mission-critical applications. MASA-CIRCA includes a Controller Synthesis Module (CSM) that can automatically synthesize reactive controllers for environments that include timed discrete dynamics. In order to best tailor its behavior to the current context, MASA-CIRCA will sequence through a number of different controllers, one for each phase of the mission (see Figure 1).

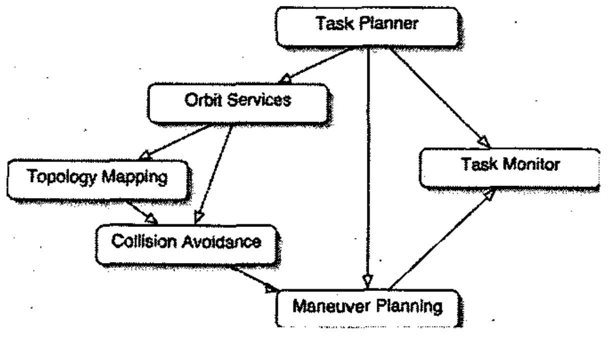


Figure 3. Computing Real-time Maneuver Plans

This controller synthesis process can occur both offline, before the system begins operating in the environment, and online, during execution of phase controllers(Fig.4) . Online controller synthesis is used to adapt to changing circumstances and to continually improve the quality of controllers for current and future mission phases. The controller synthesis process operates under the control of MASA-CIRCA's Adaptive Mission Planner (AMP) (Fig.5).

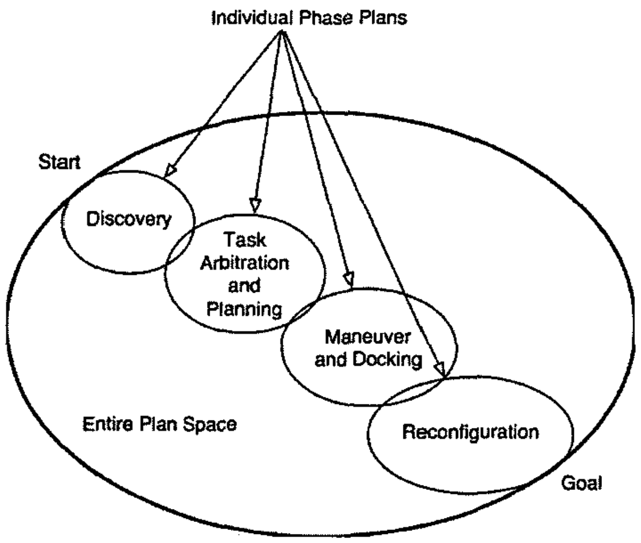


Figure 4 - MASA-CIRCA Agents Sequence Through Multiple Timed Controllers Over the course of Multiple Phase Plans.

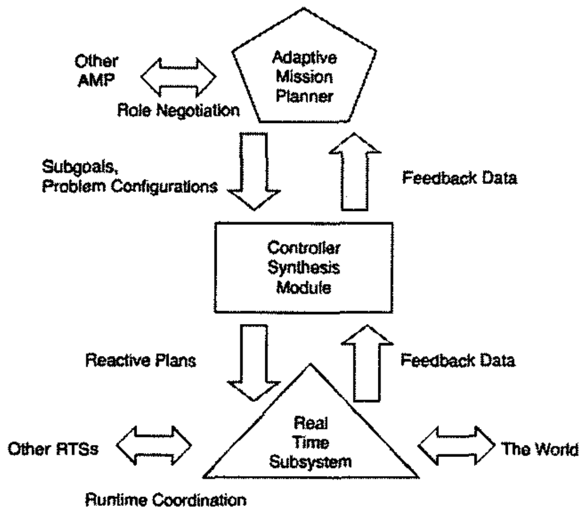


Figure 5. CIRCA Architecture

Individual CIRCA agents make guarantees of system safety by automatically building reactive control plans that guarantee to *preempt* all forms of system failure. By *preempt*, we mean that an action is planned to disable the preconditions of a potential failure, and that the action is time-constrained to definitely occur before the failure could possible occur. In our example, it is required the the new fuel cell is installed within a certain time interval after the rendezvous with the OTV (Fig. 6).

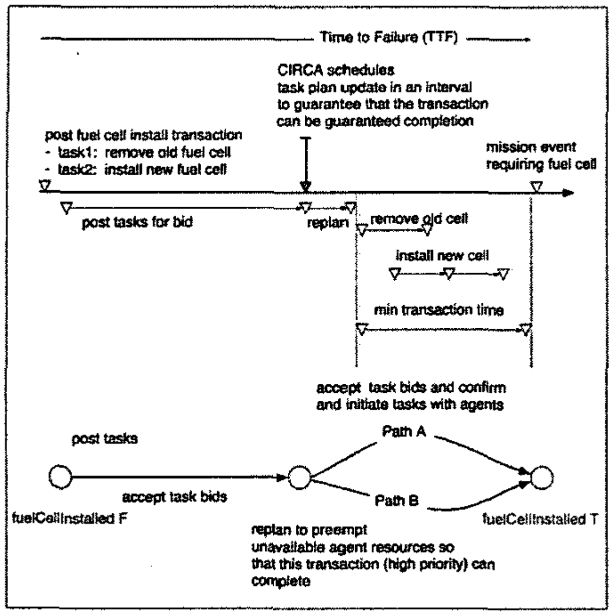

Time to Failure (TTF)
CIRCA schedules task plan update in an interval to guarantee that the transaction can be guaranteed completion
post fuel cell install transaction
- task1: remove old fuel cell
- task2: install new fuel cell
mission event requiring fuel cell
post tasks for bid
replan
remove old cell
install new cell
min transaction time
accept task bids and confirm and initiate tasks with agents
post tasks
Path A
accept task bids
Path B
fuelCellInstalled F
fuelCellInstalled T
replan to preempt unavailable agent resources so that this transaction (high priority) can complete

Figure 6. Simple Failure Preemption Example

CIRCA builds its reactive plans in the form of Test-Action Pairs (TAPs) that test for a particular situation and invoke a planned response (Fig. 7).

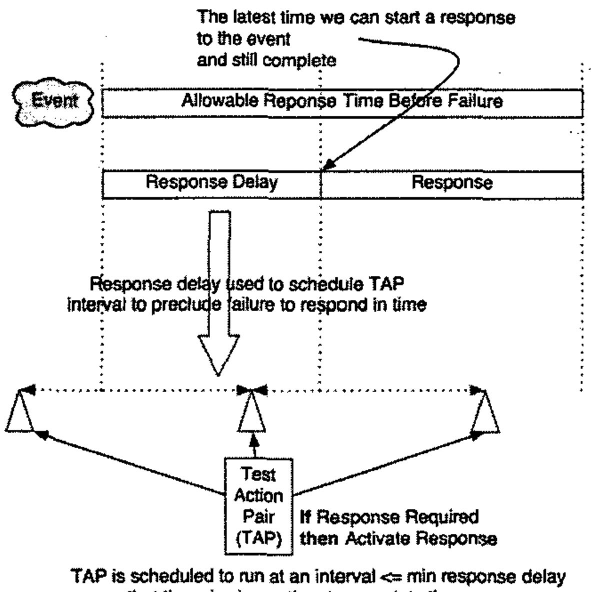

The latest time we can start a response to the event and still complete
Event
Allowable Reponse Time Before Failure
Response Delay
Response
Response delay used to schedule TAP interval to preclude failure to respond in time
Test Action Pair (TAP)
If Response Required then Activate Response
TAP is scheduled to run at an interval <= min response delay so that there is always time to complete the response

Figure 7. CIRCA Schedules TAP Intervals to Preclude Failure

In the case of the fuel cell replacement task the time interval has to account for the time required to collect and evaluate agent "bids" as well as the worst case time to execute the remove and install subtasks.

$$\Delta T\text{failure} = \Delta T\text{bid} + \Delta T\text{remove} + \Delta T\text{task interval} + T\text{install} + \Delta T\text{delay} \quad (1)$$

The associated TAP which listens for agent bids is scheduled sufficiently often that the associated tasks can be completed under worst case conditions. Since the remove and install tasks have been identified as a transaction, bids will not be accepted unless both tasks can be completed in the proper order and within the time constraints. Selecting a worst case time for scheduling is an issue as the maneuver times are determined dynamically by the servicebots based on their current location and the flight path they require to complete the tasks.

4. AUTONOMOUS COMMUNICATION AND COORDINATION BETWEEN INDEPENDENT MODULES (AGENTS)

A common issue in the scenarios described is the desired capability for autonomous discovery of all of the modules in the local space and their associated resource capabilities as well as a method for passing messages to implement the CIRCA collaborative control scheme. Based on foundations of Autonomous Agent communication, each Module can be considered an independent agent and exchange messages with other agents based on an agent communication language(ACL). In our example, various modules may be configured differently during various mission phases depending on the function and capabilities of each (Fig. 8)

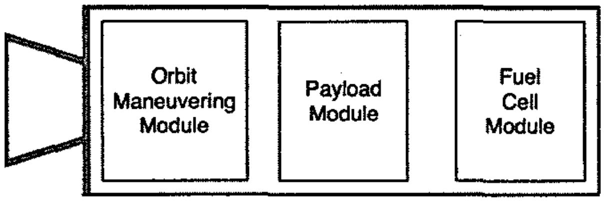


Figure. 8 Multiple Modules Acting as Independent Agents

Messages need to be exchanged on different channels depending on the service invoked[4]. At the top level, each mission is controlled by a top level work order (task plan) which specifies the phases, goals and milestones for each specific mission. Such a work order would specify the tasks to be accomplished in sufficient detail to allow the Task Planner to request the necessary resources. Our current approach is to construct a work order in a similar fashion to software install scripts, typically in a structured language like XML.

5. DEVELOPING A TAXONOMY FOR LOCALIZED VERSUS GLOBAL PLANNING AND SCHEDULING

One of the most difficult decisions in any distributed system is that of architecting the level of independent, distributed scheduling and resource management versus the level of centralized control. This difficulty is inherent in a multi-agent system whether it is autonomous or not[5]. Autonomy of a system can be distributed or centralized to varying degrees. It is the attainable independence of the agents in a system that governs how distributed the scheduling and resource management function can be, and how centralized the scheduling and resource management functions should be. Although CIRCA will provide guarantees of task completion based in the failure preemption strategies, it is still necessary to evaluate the balance between fully autonomous agent capabilities where each agent acts in its own self interest, and centralized planning of all resources. Our approach takes the middle ground with a central coordinator (the Orbital Platform planner in our example) which supervises the task generation, bidding and execution to guarantee overall best completion according to a global set of constraints.

6. CONCLUSIONS

There are several key issues to be addressed to develop a highly automated remote servicing capability for orbital platforms. The common issues of dynamic resource allocation and tasking are complicated by a highly dynamic environment where everything is in motion. In addition, the system must be developed using standard interface layers to accommodate the technology and mission changes likely to occur during the life of the platform. We are investigating the application of the CIRCA collaborative control technology currently being developed for autonomous UAV control to the space environment incorporating our experience with large scale resource planning and scheduling gained from operations with large commercial satellite networks.

REFERENCES

[1] David J Musliner, Michael J.S. Pelican, Kurt D. Krebsach, "Building Coordinated Real-Time Control Plans", 3rd International NASA Workshop on Planning and Scheduling for Space, October 2002

[2] Thomas Kreider, James Ross, "Re-Configurable Spacecraft Software: Demands and Solution", 2004 IEEE Aerospace Conference, March 2004

[3] David J Musliner, Michael J.S. Pelican, Kurt D. Krebsbach, "Multiple Agent-Based Autonomy for Satellite Constellations" , 3rd Int. NASA Workshop on Planning and Scheduling for Space, Oct 2002

[4] Reid Simmons, Trey Smith, M. Bernadine Dias, Dani Goldberg, David Hershberger, Anthony Stentz, Robert Zlot", "A Layered Architecture for Coordination of Mobile Robots", Robotics Institute, Carnegie Mellon University

[5] Daniel E. Neiman, David W. Hildum, Victor R. Lesser, Tuomas W. Sandholm, "Exploiting Meta-Level Information in a Distributed Scheduling System", Dept. of Computer Science, University of Massachusetts, May 1994

BIOGRAPHY

Jamie Ross is a Senior Principal Engineer in the Advanced Technology Department at Honeywell DSES-Glendale. He holds a BSc. in Astronautical Engineering from MIT, BSc in Computer Science for Univ. of Maryland and a MSc in Mechanical Aerospace Engineering from ASU. His previous work involved development of Honeywell's ERADS autonomous ultraviolet attitude determination and navigation sensor as well as systems engineering work on the IRIDIUM™ satellite constellation system. His current research interests include engineering information systems, distributed computing, meta-control systems design and dark beer.

Thom Kreider is a Senior Project Engineer in the Advanced Technology Department at Honeywell DSES-Glendale. He specializes in hard real time and operating systems software architecture for electronic, mechanical, power, and explosive/high-energy applications. The last three years he has focused on aerospace applications with emphasis on advanced technology for software systems and total systems engineering; several related patents are pending. He holds an B.S. in Computer Science from Park University

Dr. David Musliner is a Senior Principal Research Scientist with Honeywell Laboratories. He holds a Ph.D. in Computer Science from the University of Michigan and a B.S.E. in Electrical Engineering and Computer Science from Princeton University. Dr. Musliner designed and implemented the Cooperative Intelligent Real-Time Control Architecture (CIRCA), one of the first AI control architectures capable of reasoning about and interacting with dynamic, hard real-time domains. His current research projects include developing extensions to the CIRCA architecture for multi-agent planning and control in real-time domains including controlling teams of unmanned aerial vehicles and computer network security.

Dr. Jack Jacobs is currently the Manager of Advanced Technology for Honeywell Space Systems and is responsible for the insertion of new technology into spacecraft sub-systems. Prior to coming to Honeywell in 1997, he was a technical director for the McDonnell Douglas phantom works. Dr. Jacobs has over 25 publications and 9 patents and serves on 3 professional society committees. His specialties include attitude control systems, structural dynamics and smart structures.

Dr. Michael Fisher is currently Chief Technical Officer for KinetX, Inc. Dr. Fisher received his Ph.D. degree from Massachusetts Institute of Technology's Department of Aeronautics and Astronautics and began his engineering career at Federal Express. While there, he created software systems for aircraft and crew scheduling, aircraft fleet purchase planning, and express cargo system design. In 1990 Dr. Fisher moved to GTE Laboratories to design and implement systems to facilitate telecommunications physical plant provisioning. Dr. Fisher took a position with Motorola in 1994 as the Lead Engineer on the IRIDIUM™ project. He designed all resource allocation functionality in the network, including channel allocation, call routing, beam scheduling, and PSTN connectivity.