IMPLEMENTING REAL-TIME PROVISIONING FOR SPACE LINK EXTENSION (SLE) SERVICE INSTANCES

Kirill Lokshin, Amit Puri, Dana Irvin, Frank Ross, Rebecca Rush
Ingenicomm, Inc.

ABSTRACT

Space Link Extension (SLE) is a set of recommended standards for mission cross support developed by the Consultative Committee for Space Data Systems (CCSDS). The SLE recommendations define protocols for extending the space link from ground terminals to other facilities deeper within a ground network, allowing distributed access to space link telecommand and telemetry services. The SLE protocols are widely used to provide cross support between sites, programs, and agencies.

In traditional SLE deployments, individual service instances have been manually provisioned well in advance of the commencement of cross support for a particular mission, and hardware and software resources have been allocated to those service instances at the time of provisioning. While valid, this approach requires that dedicated resources be provided for each mission and service instance, and limits an SLE provider's ability to reallocate resources in real time based on system availability or other factors.

This paper discusses an alternative approach to SLE service provisioning, in which individual service instances are assigned resources from a common resource pool at the time that each service instance is initialized. The paper addresses the key design elements and technical tradeoffs involved in this approach, and discusses the potential benefits with regard to load balancing, equipment reuse, and resiliency against system failure.

KEY WORDS

Space Link Extension, SLE, User Services, Service Provisioning, Resource Allocation

INTRODUCTION

Space Link Extension (SLE) is a set of recommended standards for mission cross support developed by the Consultative Committee for Space Data Systems (CCSDS). The SLE recommendations define protocols for extending the space link from ground terminals to other facilities deeper within a ground network, allowing distributed access to space link telecommand and telemetry services. The SLE protocols are widely used to provide cross support between sites, programs, and agencies.

The traditional approach to SLE service provisioning involves the identification and allocation of specific resources to a service well in advance of the time at which the service becomes active. In a majority of cases, SLE services continue to rely on the availability of dedicated SLE gateway nodes – whether as stand-alone gateways or as integrated components of a mission's front-end processing infrastructure –

which are assigned to the mission at the time that the service agreement for the mission is negotiated between the entity providing SLE services and the entity using them.

While assigning a dedicated SLE gateway to every mission satisfies each mission's individual requirements, the mechanism incurs compounded costs when services are provided for multiple missions simultaneously. In particular, two weaknesses may be identified:

(1) The SLE service instances required by a typical mission do not utilize the full computing resources of a dedicated SLE gateway, particularly if those services consist primarily of low-rate command and spacecraft housekeeping telemetry channels rather than high-rate mission data channels. The remaining resources of the gateway are typically unavailable to other missions, since each gateway is dedicated to a particular mission. The problem compounds itself as the number of missions increases, as shown in Figure 1; with multiple dedicated gateways provisioning a small per-mission number of low-rate services, the majority of available aggregate computing resources remains unavailable to the user.

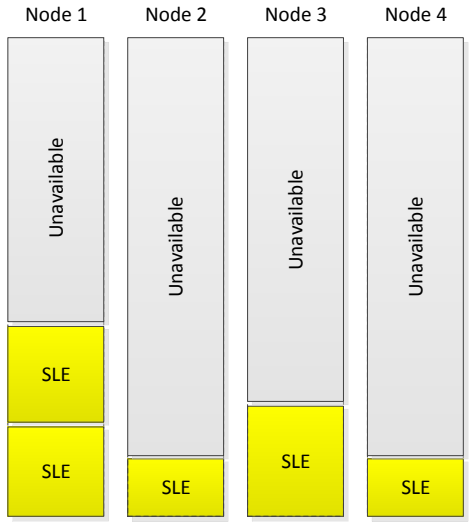

Figure 1: Resource usage across four gateway nodes by traditionally provisioned SLE services.
A significant portion of the total processing capacity of each node is effectively unavailable to the user.

(2) SLE gateway systems are typically subject to stringent redundancy requirements, particularly in the case of systems transporting mission-critical command and telemetry data. Because each gateway has a dedicated, mission-specific configuration, system redundancy is typically also implemented on a per-mission basis; this means that the number of redundant systems is equal to the number of operational systems, despite the fact that a sufficient level of fault tolerance can be achieved with a significantly lower level of redundancy.

The net result of these two weaknesses is a highly inefficient resource allocation methodology; in any given scenario, the greater portion of the computing resources available to any set of SLE services is both unused and unusable.

REAL-TIME SERVICE PROVISIONING

One alternative to the traditional service provisioning approach is the concept of real-time service provisioning. In this model, a common pool of resources, such as a set of several SLE gateways, is shared among all service instances; when a particular service instance is to become active, it is assigned the necessary resources from the unallocated portion of the pool.

While multiple implementation mechanisms for a real-time provisioning system exist, it is easiest to consider its behavior when it is represented as a pool of processing nodes, each of which has resources to provision one or more simultaneous service instances. The processing nodes may be implemented as individual hardware platforms, virtual machines running on shared hardware, or subsystem-level elements within a common hardware pool; while the approach is most applicable to a scenario where each node represents a dedicated hardware resource, it is not inherently limited to such a configuration.

Effective implementation of a real-time service provisioning mechanism requires the existence of some central point of control, which is responsible for the allocation of resources across the entire collection of processing nodes. This central point is represented by the control node shown in Figure 2 below; the control node is responsible for selecting a specific processing node to which a new service instance provisioning request is assigned, and subsequently for initializing the service instance as requested by the service management element.

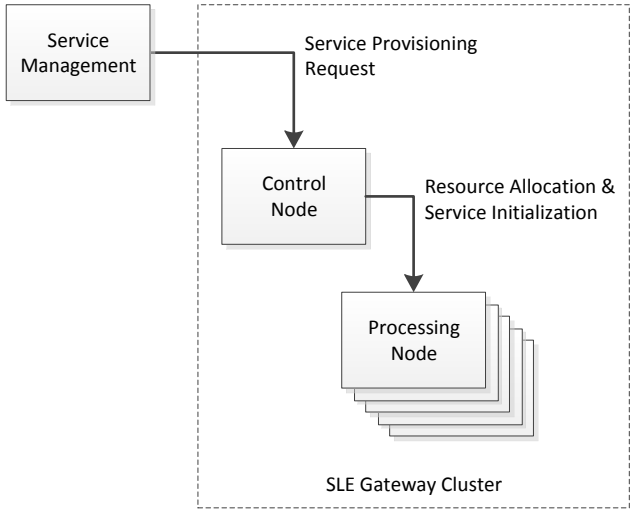

Figure 2: Real-time service provisioning in a cluster configuration. A provisioning request from the site service management element is received by a control node, which then allocates resources within a pool of processing nodes and initializes the service.

In this configuration, the provisioning of a new service instance takes place as a series of distinct operations:

(1) An external service management element sends a service provisioning request to the control node.
(2) The control node selects a processing node on which the new service instance is to reside.

(3) The control node sends system-specific commands to the selected processing node in order to instantiate and configure the new service.

In a real-time service provisioning environment of this type, the allocation of specific resources to individual service instances no longer needs to take place at the time that a service agreement is negotiated, since resources are not dedicated to a service instance for the entire lifetime of the agreement; rather, resources from a shared pool are assigned to a service instance upon request, as shown in Figure 3.

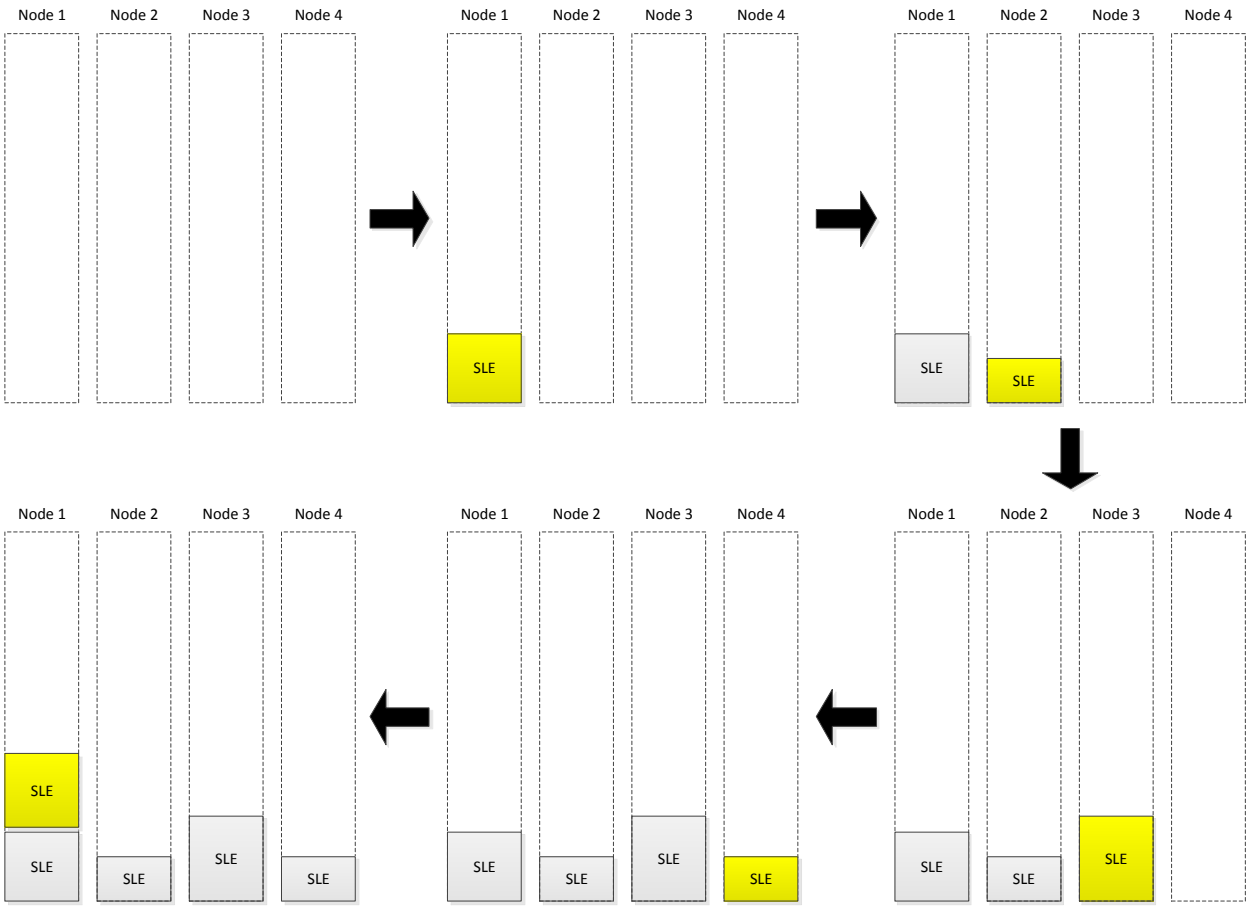

Figure 3: Real-time service provisioning in a pool of four processing nodes. Each service is assigned resources on one of the nodes via a round-robin algorithm.

The real-time provisioning mechanism provides two key advantages over the traditional provisioning mechanism, addressing the inefficiencies in the latter's resource allocation methodology:

(1) Because individual SLE gateway nodes are not dedicated to particular missions – the assignment of a particular service instance to a particular node is governed by the control node's allocation algorithms, not by a manual decision-making process – the entire computing power of each node is available for use. This allows a significantly higher total service capacity across both each individual node as well as the entire pool of nodes.

(2) Redundancy is provided to all service instances on a shared basis; in the event that any individual processing node fails, the services provisioned on that node can easily be transferred to another, arbitrary node within the pool. Because dedicated redundant gateways are no longer required, the number of redundant nodes can be sized in proportion to the required level of fault tolerance across the entire gateway cluster, reducing the aggregate volume of resources which are unavailable for use due to being located within redundant nodes.

RESOURCE ALLOCATION

The key technical element of real-time service provisioning is the resource allocation algorithm used by the control node to assign service instances to individual processing nodes. The choice of allocation algorithm has significant consequences in terms of resource allocation efficiency and aggregate resilience against node failures.

The simplest approach to allocation is a round-robin or similarly scheduled approach, as shown in the example in Figure 3. This approach provides no significant benefits to the efficiency or robustness of the cluster as a whole; its chief strengths are its simplicity of implementation and its ability to ensure that all nodes are used within a reasonable period of time. The latter offers some advantage in detecting latent node faults, since it eliminates the possibility that a node will remain in an undetectably failed condition for a significant period of time due to the lack of any service assignments to it.

The most typical approach for resource allocation in general, particularly in the case of larger clusters and increased aggregate service numbers, is a load-balancing one, as shown in Figure 4. Many specific algorithms for generalized load-balancing exist, all sharing the characteristic that they attempt to spread processing load evenly across the available nodes.

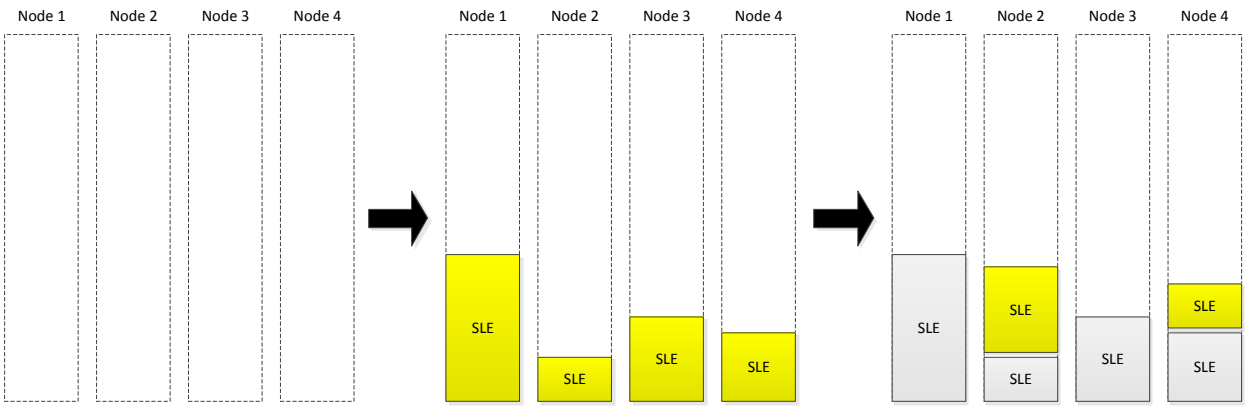

Figure 4: Resource allocation in a pool of four nodes using a load-balancing algorithm. Each new service instance is allocated to the node with the highest aggregate resource availability.

The load-balancing approach is well-adapted to ensuring efficient resource allocation, particularly in scenarios where all services consume similar amounts of resources, without unduly loading any individual node. It is also optimal from an aggregate resiliency standpoint, in that it minimizes the number of

services affected by a single node failure; this is especially true if the approach is modified to balance the number of services rather than the processing resources used by each service.

However, conventional load-balancing algorithms are vulnerable to failure scenarios in cases where the resource requirements of different service instances are widely divergent; if a service provisioning request with extremely high requirements is received when the cluster is heavily loaded, it is possible for it to fail because no single node has the required resources, despite the fact that they are available within the cluster as a whole.

An alternative to a load-balancing approach is a load-concentrating one; in this scenario, the control node attempts to allocate services onto the fewest possible number of nodes, and continues allocating new requests to those nodes until available resources are exhausted. An example of this approach is shown in Figure 5.

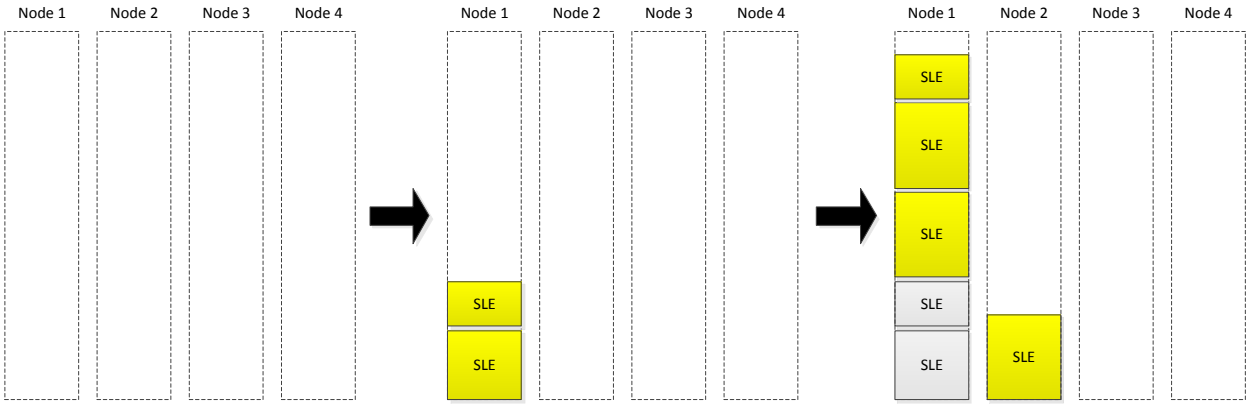

Figure 5: Resource allocation in a pool of four nodes using a load-concentrating algorithm. Each new service instance is allocated to the node with the lowest aggregate resource availability, provided that the node has sufficient resources to provision the new service instance.

The primary benefit of load-concentrating algorithms is that they eliminate the failure cases of load-balancing algorithms in scenarios with widely unbalanced resource utilizations by individual service instances; because a significant number of nodes is permitted to remain idle, requests for new services with high resource footprints can be serviced more readily than in a load-balancing scenario. However, the approach is weak in terms of resiliency; because service instances are concentrated on a smaller number of nodes, each node failure has a higher impact on overall availability.

EXTENSION TO A GENERAL CASE

While the real-time service provisioning approach lends itself to use with SLE services, it is not fundamentally limited to dealing with one type of service; so long as the resource footprint of a particular service instance can be quantified – and this is the case with virtually all user services – it is feasible to extend the real-time provisioning mechanism to manage a common resource pool shared among multiple service types.

The simplest approach to generalizing the mechanism is to superficially combine multiple, independent resource pools, with each constituent pool consisting of systems dedicated to a particular service type. In this scenario, for example, the overall cluster might consist of a pool of SLE gateways, a pool of CFDP gateways, a pool of DTN gateways, and so forth. Individual requests for each type of service could be routed through a central control node; the resource allocation, however, would occur from a dedicated pool for each service type.

A more general approach is to amalgamate the individual service-type resource pool into a shared pool, and allocate resources based purely on service footprint, regardless of the type of service involved. Such a mechanism is not feasible in all cases – certain service types may impose unique requirements on their underlying platforms – but is a reasonable option for most user services, which require nothing more than processing capacity. An example of a fully shared resource pool, with no restrictions on service allocation, is shown in Figure 6.

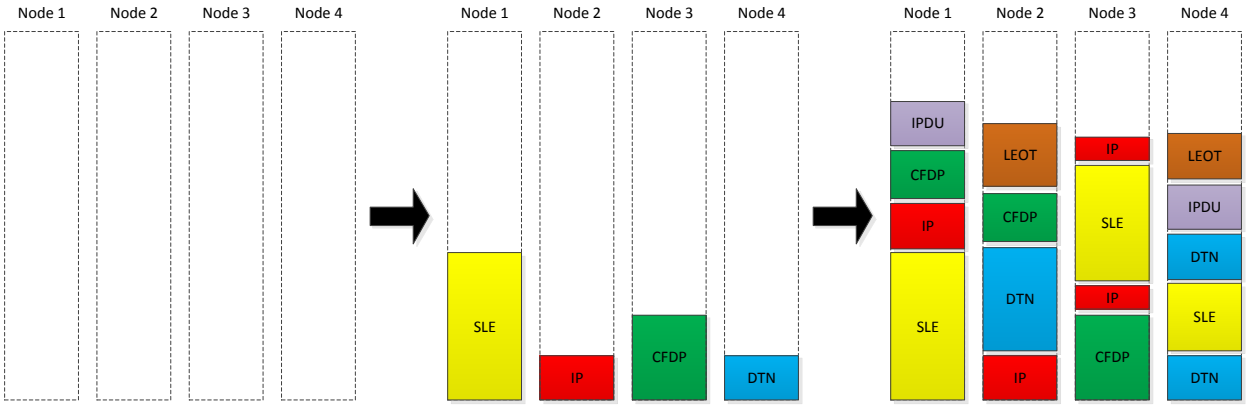

Figure 6: Unrestricted allocation of six distinct types of user services within a pool of four nodes. Each node supports multiple types of user services simultaneously.

The primary advantage of a fully unrestricted approach is increased allocation efficiency; the unused processing capacity that would otherwise be spread among multiple independent resource pools is combined into a single pool, allowing greater flexibility with regard to its subsequent allocation. The approach also lends itself to use in virtualized and cloud environments, where resources might be assigned at an abstraction layer that is fundamentally agnostic with regard to the identities of the underlying hardware platforms.

CONCLUSIONS

The real-time service provisioning approach presented here is an advantageous alternative to traditional resource management for SLE services in cases where a service provider must provision multiple logically independent service instances. It is particularly valuable as the number of instances increases; while manual, directly-managed allocation remains feasible when tens of service instances must be provisioned, it is impractical in a scenario involving hundreds or thousands of independent services.

At the same time, considerable work remains to be done before dedicated resource allocation can be entirely obsoleted. In particular, while determining the impact of failures and the necessary level of redundancy in the case of dedicated systems is relatively intuitive, doing so for a shared resource pool is considerably less so. This is especially the case in a fully generalized scenario, where multiple types of services are allocated from the same pool; in such cases, ostensibly independent service instances may have data-driven dependencies – such as the case where one user service receives data from another – that may propagate the effects of single-node failures to other nodes. The possibility of such dependencies must be taken into account when designing a resource allocation algorithm; the optimal approach may require analysis beyond that provided by simple load-balancing or similar algorithms.

REFERENCES

[1] *Cross Support Reference Model—Part 1: Space Link Extension Services*, October 2005, CCSDS 910.4-B-2.