Open Standard based Visualization of Internet Computing Systems

Seung S. Yang and Javed I. Khan Media and Communications and Network Research Laboratory Department of Computer Science Kent State University {syang@cs.kent.edu | javed@kent.edu}

Abstract

The emerging distributed internet computing paradigms envision large conglomeration of dynamic and internet-wide distributed computing resources including general and special purpose computing nodes, mass storage, and complex services built on them. A particularly novel challenge here is the automated seamless management of distributed resource pool. The early computer networking tried to shy away from graphical visualization. However, the complexity and scale of these newly emerging networked systems makes dynamic operational monitoring one of the most critical factor to determine their viability. For some time we have been engaged in experimentation with various internet computing models and services with real system implementations over grid like infrastructure. In the process we have developed novel yet detail formalisms for direct visualization of lifecycle of the application services. In this paper we present the design of this decentralized monitoring and controlling mechanism. This design can be used as a blueprint for emerging internet computing systems.

1. Introduction

The Internet and particularly the Web is increasingly becoming a computation centric network. Emerging initiatives, ranging from scientific grids to application services networks, increasingly view network as an integrated platform for joint computing and communication rather than one only for communication. The Grid initiative is exploring technology so that supercomputers, including distributed idle cycles of massive number of computers in the Internet can be used to perform advanced scientific tasks [13]. The Grid vision contemplates dynamically networked massive computing centers and massive data storage facilities interconnected by high performance data pipes. Due to the dynamic nature of the infrastructure, a computation may not receive the same set of resources on all its runs or it may change even when a computation is underway. The system status fluctuate as the resources changes. In another end of spectrum. Active network paradigm is experimenting with networked router embedded computation. It envisions providing creative solutions to many of currently hard-to-tackle network problems ranging from congestion control for time-sensitive elastic traffic, detection and recovery under distributed denial of service, to Internet telescope [17]. These essentially require placement and management of dynamically distributed active and extensible routers, nodes, and the smart applications deployed onboard over the vast Internet. Also emerging content services networks already have started flourishing over the Web using the ad hoc backend server technology. With emergence of active proxies- spearheaded by IETF technologies such as SOAP [18], OPES [19], ICAP [20], it is very likely that a more efficiency form of network distributed content services will emerge where the content processing (such as adaptation, composition, personalization, filtration) will be performed on network embedded multi-level open architecture based active proxies rather than on inextensible proprietary backend servers. Given the scale and complexity of the Internet, and the sophistication of the emerging netcentric systems, visualization will play an ever increased role in this development. A particular novel challenge here is the automated seamless management of distributed resource pools. Unfortunately, the current techniques for network's visualization are quite inadequate.

In last few years, we have developed and experimented with a number of Ad-hoc Internet Service Systems (AISSs). The developed Ad-hoc Internet Service Systems do not require any specific service node and have been tested on open Internet environment using worldwide nodes. In the process of their development, launch, and management, we were forced to develop generic visualization formalism. Based on this experience, in this paper we suggest an open architecture based process monitoring infrastructure for netcentric Internet Systems. The system works through a distributed messaging formalism that conforms to the multi-party hierarchical development pattern of practical complex systems. At the core the messaging has been built on top of the powerful and formal process description language of Petri Net [16]. The propagation of status information is controlled by hierarchically decentralized agents, and soft filtering. However, the messages can be used to construct numerous perspectives and views detailing the operation of the system for the human operator of the target netcentric system.

In this paper in section two we first present the design considerations of this visualization system including a brief area survey of related works. Then in section 3 we present the architecture of the messaging and visualization framework. In section 4.1 and 4.2 we then explain the use as well as the capabilities of the framework by two real examples of complex netcentric systems. Though in this paper we present the case with the transcoder channel and the made-to-order channel, but this framework is the result of our experience with a number of complex active services (active prefetch-proxy [1], daisychain forwarder [2], harness group communicationware [3], etc).

2. Design Considerations

2.1 Related researches

Though traditional networking research has ignored visualization, but monitoring and management of complex distributed system are becoming critical for highperformance distributed computing. However, monitoring an active distributed system such as Grid, active application or the Internet content services has several serious obstacles to overcome. The first set of complexity evolves from the scale, dynamism and versatility requirements. Additional challenge arises from autonomous ownership of the Internet systems. It is further complicated by the hierarchical and multi-party nature of net centric systems development pathway. During run time, a sound message management principle becomes very important otherwise potentially huge number of status messaging can end-up being a serious performance drag.

Recently, there have been few pioneering works in the area of Grid visualization. Tierney et. al. [4][5] suggested an agent based monitoring system to automate the execution of monitoring sensors and the collection of event data in Grid Environment. They use a direct connection between a producer and a consumer to reduce communication traffic.

Waheed et. al [8] developed an monitoring infrastructure to share monitored data using common APIs. The infrastructure is built on three basic modules, sensors, actuators, and a grid event service, and at the top of those basic modules, they built a layered monitoring system. Another layer based visualization system was suggested by Bonnassieux et. al. [6]. They offer a flexible presentation layer in huge and heterogeneous environment. It provides a simple, autonomous and extensible model that enables the visualization of any level of abstraction using a hierarchical view model of resources status, with propagation of monitoring status up to the top of the tree view. The gathered information for monitoring can also used for system management. Reed, et. al. [7] suggests using system monitoring results for adaptive control to improve system reliability. The system uses diskless check-pointing, which enables more frequent checkpoints by redundantly saving check-pointed data in memory, and low-cost mechanisms to capture data for failure prediction, which enables creation of dynamic schemes for improved application resilience.

2.2 Proposed System

2.2.1 Autonomous Service Hierarchy

One of the major challenges that differentiate netcentric systems from the traditional modular distributed software is the fact that the concept of internet autonomous systems (that separates network for the Internet) also extends to the software systems. This hierarchically dependent multiparty involvement extends to both to the development process of compose-able services as well as to the runtime service ownership. Clearly, these systems are not built with a simple one big program rather with several independent system components running on multiple computing systems. Each system component is also composed with several sub components and distributed among multiple computing systems. Also uniquely quite often these are developed under multiple autonomous service authorships, and deployed and managed under multiple service ownerships. Because of such nature, system monitoring and controlling get considerably more difficult and complex. In addition, the trend that current network based sub-systems and components have to go through frequent modification for the newly included or upgraded components makes the overall task further unmanageable. As a result, the system management and monitoring

software encounters difficulties to visualize whole system across the participating computing systems and services, and some times the software is faced with disparity between system status reports and control messages, and their representation in the system. However, the same complexity makes monitoring and visualization of the process nevertheless more critical. Therefore, for the Internet centric system's visualization the support for autonomous modular visualization becomes very critical. In this paper we present a simple yet powerful framework towards this goal.

2.2.2 Other Features

Besides autonomous modular visualization we also offer the following architectural features to overcome the challenges of scale, dynamism and versatility requirements.

- Controllability of message flow.
- Adaptation of system viewer's perspectives.
- Use of meta-information for interpretation of current system information.

As the system gets bigger, the generated system status messages also grow. Without controllability of a system message flow, a system is easily overwhelmed by the generated status messages and couldn't deliver important information.

The system status representation should be useful enough to produce multiple points of views. Users require different perspective views depending on user's interests at a given moment. Representing same system information in various ways will increase a user's focus on his/her interests.

The separation of system information data and its structures by using meta information makes it easy on upgrading system components while system is running and verifying message information as well. Using the meta information also leverage automation of system information representation. A system can generate various target representations by dynamically interpreting message data with its meta information.

We have built the visualization schema on the powerful process description language of Petri Net. A Petri Net is a graphical and mathematical modeling tool which consists of places, transitions, and arcs that connect them. [16] It is powerful tool for modeling systems that are concurrent, asynchronous, distributed, parallel, nondeterministic, and stochastic. It is well suitable to describe a system's status and its transition. Recent proposal of Petri Net Markup Language (PNML) is pushing Petri Net language to more interchangeable format for system modeling. [15].

The framework we propose is particularly suitable for monitoring the lifecycle of loosely coupled and scalable complex multiparty active systems. The developed formalism allows sub-system to maintain its own status and control messages within it. A sub-system, when used as a part of a high level composed system, can however further report some of its status and control messages to its upper level system. Furthermore each level supports several reporting and message propagation modes to allow performance tuning. The time, type, and content of messages are decided initially by the service designer. However this default behavior can be overridden by the system operator at run time. A privileged user can freely control and monitor the system status using a flexibly configurable multi-view visualization system.

3. Ad-hoc Internet Service System Model

We demonstrate the use of this visualizer system on a set of custom transport systems built on ABONE [14]. ABONE [14] is an operational network and provides an Internet wide network of routing as well as processing capable nodes. The software structure of ABONE node involves a native Node Operating System (NOS) and Execution Environments (EEs), which acts like a remote shell and provides a programming framework to ABONE applications. A special EE called ANETD allows users to obtain secured and controlled access to the ABONE resources and support EE module management, such as starting, stopping, monitoring modules and EE. We have proposed the Ad-hoc Internet Service System (AISS) framework to launch and manage general purpose internet services. For ABONE we have also developed a distributed system shell called Virtual Switching Machine (VSM), which turns an ABONE computer into a node capable of working within AISS service [12]

The AISS is shown in Figure **1**. It accepts deployed and lunched system components to build and run ad hoc deployable distributed Internet Service. A system component is deployed and lunched after authenticated by the VSM with the system policies. [12]. The Ad-hoc Internet Service System has following features.

- A system consists of a group of sub-systems. Each sub-system may have its own sub-systems/services.
- Each sub-system can be independently executed and monitored.
- Each sub-system has uniform interfaces to access its status and information.

A sub-system has its own services/components to the system. and it is connected to the other subsystems. Each sub-system further can be divided into more small subsystems/components and has its own statuses and controls. The control and interfaces monitor user run independently from the monitoring system and controlling system. With the separation of the user interfaces from the monitoring and controlling system, a user freely controls and monitors the system from any authorized terminal.

A status code represents a status of a system. The status code only has meaning within the 'system'. The meaning of the status value is represented in a well formed status description language and is gathered and interpreted by the monitoring system. When a new system component is developed, the description of the status values are supplied together with the component. The monitoring and controlling system dynamically binds and interprets the meaning of status message with the given description.

3.1 Visualization Architecture

Figure 3 shows visualization The monitoring architecture. components of AISS are run on Kent VSMs. They are deployed and executed on Kent VSM as a part of a construction. А service Status Monitor (SM) processes status message of a sub-system. A SM stores status message structure descriptions and delivers or saves status messages of the sub-system. A Control Monitor (CM) handles control messages. A CM is added in a sub-system when the sub-system supports a control mechanism from outside of the system. Initiation and execution of a monitor is coordinated by sub-system management software.

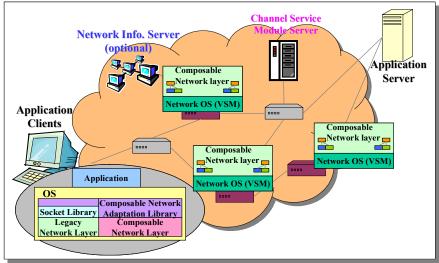
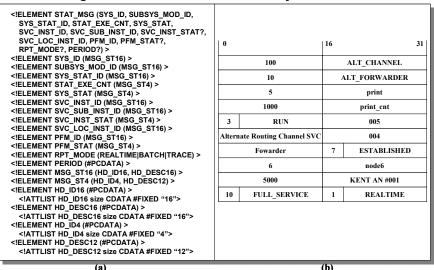


Figure 1 Ad-hoc Internet Service System Model





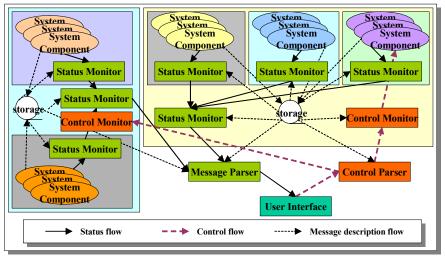


Figure 3 Visualization System Architecture

3.2 Dynamic Message Binding

The visualization system supports 1) dynamic interpretation of the system status messages, 2) seamless navigation through layer abstraction and visualization of the given layer of a system, 3) uniform method of visualization at all levels. When a new system component is developed, the descriptions of its status message structures and the descriptions of its state diagram are supplied by the developer together with the component in a *state description file* (SDF). As per this template, a status monitoring and visualization system dynamically binds and interprets the meaning of a status message with the given description.

The Visualizer is an independent set of software which can interprets state messages with their state description file, visualization device description, and user's display method selection.

Each system is composed of code modules (if it a base system) and/or isomorphic sub-systems (for more complex ones). Each time a system is installed (i.e. all of its sub-systems are launched) a status monitor, which is one of system code modules, is also installed. A set of messages are generated towards this state monitor in subsystem's leader module. A visualization system can use a subset of the messages to present various perspectives on the system. The key challenge here is that these messages should carry enough information to identify it-self with respect to the major perspective frameworks within which an active service operates. Figure 2 shows an example of status message structure and a status message. Below we provide the proposed try-partite identifier system. This message system encodes the fields in its messages (i) system identifier, (ii) subsystem module identifier, (iii) system state identifier, (iv) state execution count (v) system status, (vi) service instance identifier, (vii) service subsystem instance identifier, (viii) service instance status, (ix) service location instance identifier, (x) platform identifier (xi) platform status. The primary state identifier set i-iii is assigned by the module programmer who has coded the active modules. This identifier set has to be hierarchically unique within a specific version of specific software. The identifier set vi-vii are to be assigned by the active service administration system (such as EEs/ ANETDs) at the time of installing and initializing instances of the service at each instantiating of loaded modules. Again, these identifier set has to be hierarchically unique within the service administration domain. The last identifier x is to be supplied by the active node's owners. These are assigned when a node joins an active network domain. This typically can include autonomous system number, IP address, location etc. The status information iv and v is computed by the code

modules at run time. Its values are determined by the programmer. The service instance status information viii, if any, is passed on to the monitor messaging agents by the service administration local agent (such as node EE). The status information xi, if any, is set by the local node administrator during the period the service is running. The monitor messaging system collects and composes the messages prior to generating the network messages. Messages can contain control flags to control the mode of reporting and even to filter the content to tune performance. The system allows three reporting modes (i) REAL-TIME, (ii) BATCH, (iii) TRACE-ONLY. In realtime mode the monitor messages are generated and sent when the code executes through the state points. In BATCH-ONLY mode the messages are generated at realtime but forwarded periodically in batch. The period is decided by a PERIOD field. The mode feature only modifies the time of sending the monitor messages but do not affect their content. Three flags are further used to negotiate filtering the three status fields in the messages. In every message sent by the monitor messaging agent the flags are set according to the current value of these flags. A set of control messages can be potentially sent in reverse direction to request change in these flags (and the PERIOD field). The transition among the modes is shown in Figure 5.

4. Case Analysis

In this section we illustrate the features of the visualization schema with two real examples. The first example illustrates a simple netcentric system which offers a virtualized bandwidth adapting high level transport. It used network embedded video rate transcoders. The second example uses multi-level virtualization.

4.1 Test Case 1: SONET Channel system

"SONET Channel" is an internet service packaged as custom channels which offers video rate transcoding for video applications. Internally it is made of network deployable distributed video transcoding modules. However, end-applications do not have to worry about them. From end-applications' point of view, they simply request and use this special transport by socket like interface and rest is handled automatically. See [9][11] for detail of this already demonstrated concept system. In reality, the VSM redirects the request to SONET manager module and which in turn handles the installation of the components in the pathway between the requesting endpoints with help from the VSMs. The channel's basic components are X-DEC, X-ENC, and X-MUX. Depending on the components needs, the SONET channel deploys as many X-ENCs as possible to fulfill the user required transcoding rates. When a video serverapplication sends a video stream, X-DEC decodes the input video stream and schedule to send a video unit to one of X-ENCs based on the available computing and network resources on the network. An X-ENC encodes given video unit and forwards the transcoded video unit to the X-MUX. An X-MUX forward received video units to a client in sequence.

We have augmented the proposed visualization mechanism with this novel SONET channel system. Figure 6 shows the underlying Petri-net model of the sonnet channel system. This is provided by the SONET developer and resides at the Channel Service Module Server (Figure 1). This is retrieved by visualization manager (in SONET's control center program) as the channel system is installed and then the visualization manager begins receiving the status messages as the channel system starts running.

Figure 7 shows an example display schema. When the visual system receives a system status message, the visualization manager interprets the value with given system status message description. With the programmer supplied component and status descriptions, the scheme can be quite intuitive and telling as shown in Figure 7. The manager of the service can receive real-time information about the status of the entire service based on this. Since, the actual "symbols" and interpretation method of the symbols are sent to the visual system - thus very rich real-time status information can be passed on to this schema by this method.

4.2 Test Case 2: MTO Channel System

"MTO Channel" is another interesting example of

channels custom that features concurrent communication to meet the QoS requirements specified by the user applications [12]. It too uses a socket like interface. However, internally, it's manager analyzes component network graph and individual links' OoS metrics and then casts hierarchically configured pathways. The splicing is performed by embedding specialized forwarding modules appropriate in

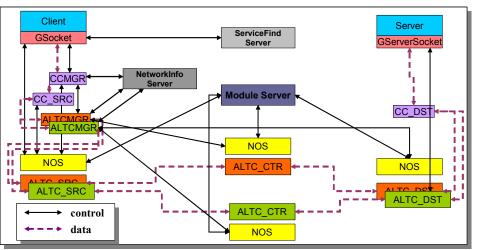


Figure 4 Concurrent Channel builds with using two Altered Routing Channels

network points. For infinite hierarchical construction these modules are packaged further into two special channel constructs called "Concurrent Channel (CC)" and "Altered-Routing Channel (ARC)". An MTO Channel is constructed by using the above two types as its sub components in various combinations. The Figure 4 shows an example CC channel construction scenario using two ARC channels as its sub channels. [10]. As before for each sub-system the messages are default routed to their sub-system's status monitor (in this case, all the status message is delivered to the Channel Control Visualizer).

The messages can be used to create quite a versatile set of views and perspectives about the overall service. In this example we show two views. The first shows (Figure 8) color coded system status on a "Component Location Matrix" view. Here all the modules (including those of sub-systems) have a column, and each Internet site participating in this service (i.e. running a module) has a row. The color in a cell shows the status of the module running on a site. In this channel, the same set of messages has also been used to build another scheme "Hierarchical Service Tree" view. It shows as a tree how that which sub-systems are operating or not and what is their status- thus can identify issues specific to ownership of sub-services. The power of the proposed visualization schema is derived from that fact that a wide number of views can be constructed to meet quite diverse range of requirements conforming to the multi-party and hierarchical ownership and development pattern of Internet services and systems.

5. Conclusion

Internet computing increases the availability and the resource utilization. However, it also increases

complexity of a system. As a system becomes complex, the control of the system and the representation of the system status increase their complexity too. It becomes almost impossible to manage a system without proper status monitoring and controlling system. In this paper, we have presented a streamlined yet simple status representation methods for net centric systems using decentralized component based status reports and dynamic binding of message descriptions. The suggested visualization system for Ad-hoc Internet Service System gives intuitive status report and simple control using decentralized status report and controlling mechanism. Layered approach along with dynamic bindings of message descriptions supports powerful abstraction for status and control representation. This work can be used as a blueprint for an open standard based visualization framework for the emerging internet computing systems.

This work has been funded by the DARPA Research Grant F30602-99-1-0515 under its Active Network initiative.

6. References

- [1] Javed I. Khan and Qingping Tao, Partial Prefetch for Faster Surfing in Composite Hypermedia, 3rd USENIX Symposium on Internet Technologies and Systems, USITS 2001, San Francisco, March 2001, pp13-24.
- [2] Javed I. Khan, & S. S. Yang, Made-To-Order Custom Channels for Netcentric Applications over Active Network, Proc. Of the Conf. On Internet and Multimedia Systems and Applications, IMSA 2000, Nov 2000, Las Vegas, pp22-26.
- [3] Javed I. Khan and Asrar Haque, An Active Programmable Communication Harness for Measurement of Composite Network States, Submitted to IEEE International Conference on Networking, ICN' 2001, pp628-638.
- [4] B. Tierney, B. Crowley, D. Gunter, M. Holding, J. Lee, M. Thompson, A Monitoring Sensor Management System for Grid Environments, Proceeding of the IEEE High Performance Distributed Computing Conference (HPDC-9), August 2000
- [5] B. Tierney, R. Aydt, D. Gunter, W. Smith, V. Taylor, R. Wolski, M. Swany, A Grid Monitoring Service Architecture, Global Grid Forum Performance Working Group, 2001
- [6] Bonnassieux F., Harakaly R., Primet P.: MapCenter: an Open GRID Status Visualization Tool, proceedings of ISCA 15th International Conference on parallel and distributed computing systems, Louisville, Kentucky, USA, September 2002.
- [7] Daniel A. Reed, Charng-da Lu, Celso L. Mendes, Big Systems and Big Reliability Challenges, Proceedings of Parallel Computing 2003, Dresden,

Germany, September 2003.

- [8] A. Waheed, W. Smith, J. George, J. Yan, An Infrastructure for Monitoring and Management in Computational Grids.
- [9] Seung S. Yang and Javed I. Khan, Delay and Jitter Minimization in Active Diffusion Computing, International Symposium on Applications and the Internet, Jan. 27-31 2003, Orlando, Florida
- [10] Seung S. Yang and Javed I. Khan, A Frame work for Recursive Channel Construction, Technical Report: 2004-01-05, Kent State University, [URL: <u>http://www.medianet.kent.edu/technicalreports</u> .html]
- [11] Javed I. Khan, Seung S. Yang, Qiong Gu, at el. Resource Adaptive Netcentric System: A case Study with SONET– a Self-Organizing Network Embedded Transcoder. In Proceedings of the ACM Multimedia 2001, pp617-620, Ottawa, Canada, October 2001.
- [12] Javed I. Khan and Seung S. Yang, A Framework for Building Complex Netcentric Systems on Active Network, Proceedings of the DARPA Active Networks Conference and Exposition, DANCE 2002, May 21-24, 2002, IEEE Computer Society Press, San Jose, CA.
- [13] Daniel Reed, Grids, the Teragrid, and Beyond, IEEE Computers, January 2003, pp.62-68.
- [14] Steve Berson, Bob Braden, Steve Dawson. Evolution of an Active Networks Testbed, Proceedings of the DARPA Active Networks Conference and Exposition 2002, pp. 446-465, San Francisco, CA, 29-30 May 2002
- [15] Jonathan Billington, Søren Christensen, Kees van Hee, Ekkart Kindler, Olaf Kummer, Laure Petrucci, Reinier Post, Christian Stehno, Michael Weber, The Petri Net Markup Language: Concepts, Technology, and Tools, ICATPN 2003, Eindhoven, Netherlands, June 2003
- [16] Gianfranco Balbo, Jörg Desel, Kurt Jensen, Wolfgang Reisig, Grzegorz Rozenberg, and Manuel Silva, Introductory Tutorial on Petri Nets, 21st International Conference on Application and Theory of Petri Nets, Aarhus, Denmark, June 26-30, 2000
- [17] M.Soga, T.tanaka, M.Okyudo, et.al., "A Remote Control System of a Telescope by WWW and Real Time Communication for Astronomical Education", ED-MEDIA/ ED-TELECOM'98, Vol.2, P.1305-1310
- [18] W3C. Simple Object Access Protocol, Technical Reports, URL: http://www.w3.org/TR/soap
- [19] Internet Engineering Task Force Open Pluggable Edge Services Working Group, URL: http://standards.nortelnetworks.com/opes/index.htm
- [20] The ICAP Forum, Internet Content Adaptation Protocol, URL http://www.i-cap.org

Published in the Proceedings of the Int. Conf. on Computer Graphics, Imaging and Visualization Penang, Malaysia, 26-29 July 2004, Elsevier, ISBN 983-861-289-8, p319-327

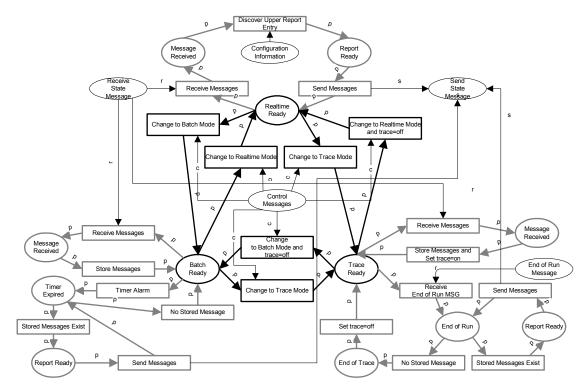


Figure 5 Monitoring Point Status Transition Diagram

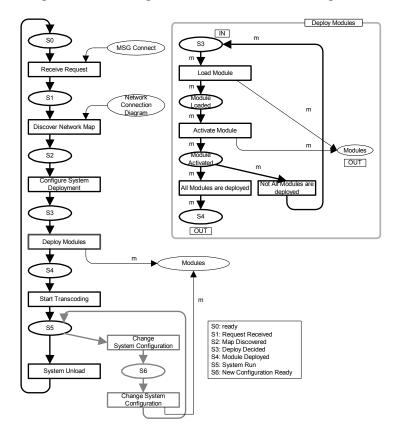


Figure 6 SONET Service Status Transition

Published in the Proceedings of the Int. Conf. on Computer Graphics, Imaging and Visualization Penang, Malaysia, 26-29 July 2004, Elsevier, ISBN 983-861-289-8, p319-327

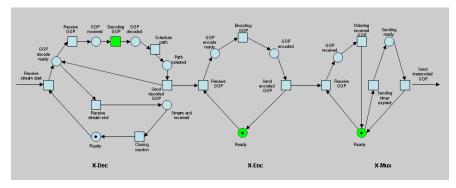


Figure 7 State Transition Diagram View of SONET System

	cchannel	cosrc	ccdst	allchann	altcsrc_0	altcctr_0	altcdst_0	altchann	allcsrc_1	altcctr_1	altcdst_1	Status Legend	
mk00	ESTABLI	ESTABLL	N/A	ESTABLI	ESTABLI	NA	N/A	MITCHLU.	WITHALL	N/A	NA	LOADED ACTIVATED	
mk01	NA	NA	NA	NA	NA	BATTON 12.	N/A	NA	NA	NA	NA	SUB_SETUPED	
mk02	NA	NA	PHENDELSE.	NA	NA	NA	ACTIVAT	N'A	NA	N/A	NA	ESTABLISHED SUSPENDED	
mk03	NA	NA	NA	NA	NA	NA	NA	NA	NA	ACTIVAT_	NA	DEACTIVATED	
	_											UNECADED	

Figure 8 Color Coded Component & Location Matrix View of MTO Channel System

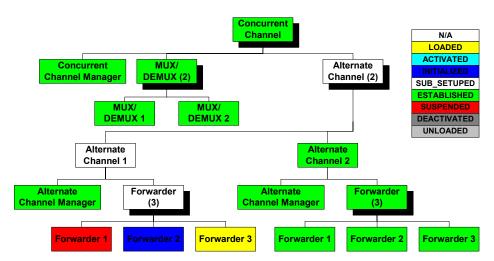


Figure 9 Hierarchical Service Tree View of MTO Channel System