Open Standard Based Visualization of Complex 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 involvement of large conglomeration of dynamic and internet-wide distributed computing resources. Visualization is an indispensable tool to confront the challenge of managing such complex systems. We present a monitor messaging framework for visualizing the lifecycle of a multiparty decentralized process. This design can be used as a blueprint for an open standard based visualization protocol 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. Research in topics such as composable services, active and programmable networking, etc. are pushing this envelop. In last four years, we have developed and experimented with a number of Ad-hoc active Internet Service Systems (made-to-order channel 1, transcoder channel 5, active prefetch-proxy, etc). Our experience with complex netcentric composable systems indicates that monitoring and management of complex distributed system is critical for enabling growth of complex distributed computing. In this paper we discuss a monitoring framework that we have successfully used in several of our systems Ad hoc Internet Service Systems (AISS). This infrastructure propagates the status information and control commands via multi-level decentralized agents and can be used to generate unified views of the overall operation of the services.

2 Design Considerations

Though traditional networking research has ignored visualization, but very recently there has been few pioneering works in the area of Grid visualization. Tierney et. al. [2] suggested an agent based monitoring system in Grid Environment. They use a direct connection between producer and consumer to reduce communication traffic. Waheed et. al [4] developed an monitoring infrastructure to share monitored data using common APIs. Another layer based visualization system was suggested by Bonnassieux et. al. [3]. They offer a flexible presentation layer for displaying monitoring data from a huge and heterogeneous environment. The framework we propose is particularly suitable for monitoring the lifecycle of loosely coupled and scalable complex multiparty active systems. Besides fulfilling the basic responsibility of process event reporting the design of our visualization framework offers the following distinguishing features. It allows sub-system to maintain its own status and control messages within it. A sub-system can further report its status and control messages to its upper system. Furthermore it supports several reporting modes to allow performance tuning. The time, type and content of messages all are decided initially by the system designer and can be overridden by the system operator or administrator at run time. A privileged user can freely controls and monitors the system status using a flexibly configurable multi-view visualization system from any authorized terminal.

3 Ad hoc Internet Service System Mode

3.1 Visualization of Control and Status Architecture

ABONE 6 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. The ANETD is one of the developed EEs and allows users to obtain secured and controlled access to the ABONE resources. The AISS uses the ANENTD as its run time base infrastructure. However, the AISS is independent from the ANETD. AISS runs on our Virtual Switching Machine (VSM) EE which runs via ANETD. The Kent VSM can deploy and lunch authorized system components to build a service. [5]

Fig. 1 described the symbolic representation of visualization system architecture. The monitoring components of AISS are run on Kent VSMs. They are deployed and executed on Kent VSM as a part of a service construction. A 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.

3.2 Dynamic Message Binding and Interpretation

Based on the AISS design considerations, the visualization system should support 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. The meaning of a status message is represented in a well formed status structure description language and is gathered by a status monitoring

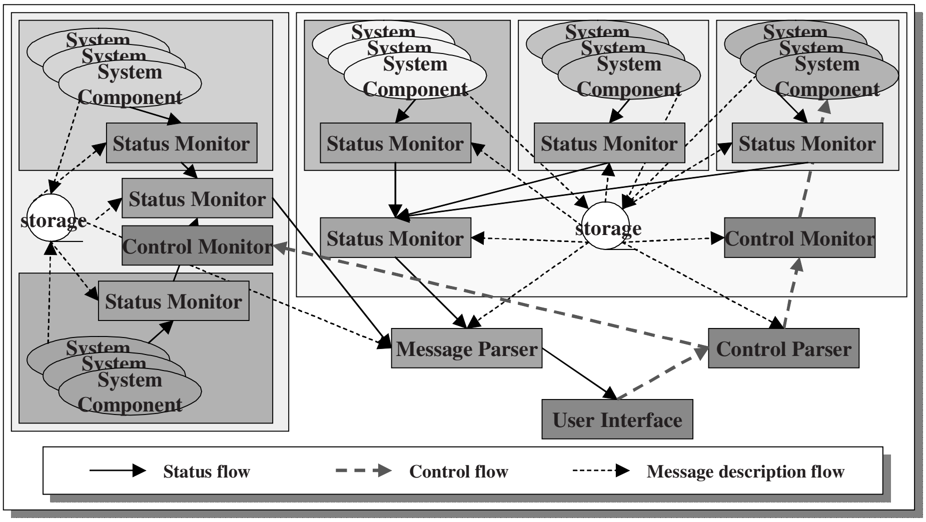

Fig. 1. Visualization System Architecture

system. When a new system component is developed, the descriptions of its status message structures and the descriptions of its state diagram are supplied together with the component. A status monitoring system and a visualization system dynamically binds and interprets the meaning of a status message with the given description.

The Visualizer integrates with a code server based hierarchical code server-based service deployment framework. Each system can have isomorphic sub-systems and/or code modules. Each time a system is installed (i.e. all of its sub-systems are launched) a hypothetical state-monitor is assumed to be concurrently instantiated. A set of messages are generated towards this state-monitor in sub-system'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 various perspective frameworks within which an active service operates along with the actual status information. Below we provide a 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 programmer who has coded the active modules. This identifier set has to be *hierarchically unique* within a specific version of a specific software. The identifier set vi-vii are to be assigned are to be assigned by the active service administration system (such as EEs/ ANETDs) while 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 owner. These are assigned when a node joins an active network domain. The status information iv and v is computed by the code modules at run time and thus its value is designed 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 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 real-time 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 real-time 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).

```
<!ELEMENT STAT_MSG (SYS_ID, SUBSYS_MOD_ID,
   SYS_STAT_ID, STAT_EXE_CNT, SYS_STAT,
   SVC_INST_ID, SVC_SUB_INST_ID, SVC_INST_STAT?,
   SVC_LOC_INST_ID, PFM_ID, PFM_STAT?,
   RPT_MODE?, PERIOD?) >
<!ELEMENT SYS_ID (MSG_ST16) >
<!ELEMENT SUBSYS_MOD_ID (MSG_ST16) >
<!ELEMENT SYS_STAT_ID (MSG_ST16) >
<!ELEMENT STAT_EXE_CNT (MSG_ST4) >
<!ELEMENT SYS_STAT (MSG_ST4) >
<!ELEMENT SVC_INST_ID (MSG_ST16) >
<!ELEMENT SVC_SUB_INST_ID (MSG_ST16) >
<!ELEMENT SVC_INST_STAT (MSG_ST4) >
<!ELEMENT SVC_LOC_INST_ID (MSG_ST16) >
<!ELEMENT PFM_ID (MSG_ST16) >
<!ELEMENT PFM_STAT (MSG_ST4) >
<!ELEMENT RPT_MODE (REALTIME|BATCH|TRACE) >
<!ELEMENT PERIOD (#PCDATA) >
<!ELEMENT MSG_ST16 (HD_ID16, HD_DESC16) >
<!ELEMENT MSG_ST4 (HD_ID4, HD_DESC12) >
<!ELEMENT HD_ID16 (#PCDATA) >
   <!ATTLIST HD_ID16 size CDATA #FIXED "16">
<!ELEMENT HD_DESC16 (#PCDATA) >
   <!ATTLIST HD_DESC16 size CDATA #FIXED "16">
<!ELEMENT HD_ID4 (#PCDATA) >
   <!ATTLIST HD_ID4 size CDATA #FIXED "4">
<!ELEMENT HD_DESC12 (#PCDATA) >
   <!ATTLIST HD_DESC12 size CDATA #FIXED "12">
```

(a)

0		16	31
100		ALT_CHANNEL	
10		ALT_FORWARDER	
5		print	
1000		print_cnt	
3	RUN	005	
Alternate Routing Channel SVC		004	
Fowarder		7	ESTABLISHED
6		node6	
5000		KENT AN #001	
10	FULL_SERVICE	1	REALTIME

(b)

Fig. 2. A Status Message Structure and a Status Message

4 Conclusion

The suggested visualization system for Ad hoc Internet Service System gives intuitive status report and simple control using decentralized reporting for a multi-component multi-location internet service. Just like other IP based messaging (such as ICMP or TCP signaling, etc.) this monitor messaging should be wrapped with an authentication protocol. This overall design can be used as a blueprint for an open standard based visualization protocol for emerging internet computing systems. This work has been funded by the DARPA Grant F30602-99-1-0515 under its Active Network initiative.

References

1. 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.
2. 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
3. Bonnassieux F., Harakaly R., Primet P.: MapCenter: an Open GRID Status Visualization Tool, proceedings of ISCA 15th Int. Conf. on parallel and distributed computing systems, Louisville, Kentucky, USA, September 2002.
4. A. Waheed, W. Smith, J. George, J. Yan, An Infrastructure for Monitoring and Management in Computational Grids.
5. Seung S. Yang and Javed I. Khan, Delay and Jitter Minimization in Active Diffusion Computing, Int. Symp. on Applications and the Internet, Jan. 27-31 2003, Orlando, Florida.
6. 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.