

# QoS Assurance of Next Generation Network (NGN) Applications

Swapna S. Gokhale  
Dept. of Computer Science and Engineering  
University of Connecticut, Storrs, CT 06269  
Email: ssg@enr.uconn.edu

## 1 Introduction and Motivation

A Next Generation Network (NGN) application is an application that provides a value-added service seamlessly across an integrated network consisting of a combination of packet-switched, circuit-switched, wireless and wireline networks. Such an application is likely to be developed using a set of open, standard APIs. These APIs provide a level of abstraction to shield the application programmers from the underlying complexities of heterogeneous networks. As a result, these APIs facilitate rapid service development by allowing an application developer to focus on the functionality (that is, what is provided), rather than on the implementation (that is, how it is provided). The importance of open, standard APIs in encouraging the growth of a community of application developers is evident from a number of initiatives to develop such APIs and environments providing these APIs. These initiatives include JAIN [2]<sup>TM,1</sup>, OSA/PARLAY [3], and 3GPP [1]. Although these APIs facilitate rapid service creation by eliminating the need to understand the peculiarities of the underlying network infrastructures, the quality assurance of these services will require a service developer to become fully cognizant of such details. This requirement, which is primarily due to the lack of a consistent methodology for quality assurance, significantly mitigates the advantage of decreased development time gained by using open, standard APIs. A systematic framework for QoS assurance based on open, standard APIs provides an integrated approach for rapid service creation as well as quality assurance of these services and is the focus of this paper. The QoS assurance framework described in this paper is based on the use of JAIN<sup>TM,2</sup> APIs.

## 2 Java APIs for Integrated Networks (JAIN)

JAIN is a community of companies led by Sun Microsystems under the Java Community Process that is developing

<sup>1</sup>JAIN is a trademark of Sun Microsystems.

<sup>2</sup>JAIN is a trademark of Sun Microsystems.

standard, open, published Java APIs for NGNs. The JAIN approach integrates wireless, wireline, and packet-based networks by separating service-based logic from network-based logic. Based on this point of view, JAIN APIs are divided into two layers, namely, protocol layer and application layer. The protocol layer in JAIN is based on Java standardization of specific protocols, for example, Session Initiation Protocol (SIP), H.323, and Media Gateway Control Protocol (MGCP). The application layer APIs provide a single call or session model for multiparty, multimedia, and multiprotocol sessions for service components in the application layer. JAIN is based on a provider/listener event driven model. An event model is a software architecture which enables components to notify a state change to other components. In case of JAIN, the platforms implementing the JAIN APIs raise events to inform the interested applications of the relevant happenings in the network. JAIN provides a mechanism for applications which implement services to register with the platform to receive the notification of such relevant events. The applications then react to these events in order to provide the desired services. We exploit the event driven software architecture underlying the JAIN framework to facilitate the reuse of these APIs for the purpose of QoS assurance as described in the next section.

## 3 QoS Assurance Framework

The backbone of the QoS assurance framework consists of recording and collecting the events raised by the platforms implementing the JAIN APIs. Central to this framework are event collection and aggregation applications (ECAs) which will register with the JAIN platforms to be notified of the events generated by the platform. The ECAs will then receive a stream of JAIN events generated by the platform. This raw stream of events will be aggregated in two ways. In the first type of aggregation, event counts for different types of events will be generated for a specified interval. For example, an event count indicating the number of call origination attempts over a 30 minute interval may be generated. In the second type of aggregation,

the events generated from a single session will be aggregated to track the progress of the session. These aggregated events may be subject to various types of analysis to assess the quality of the services provided by the various applications and the JAIN platform. Some of these analyses types include:

- *Performance and dependability analysis:* Aggregation of events from a single session will allow us to determine the performance of the service requested by that session. Also, the number of successfully completed service requests for a given service will provide a view of the reliability of that service.
- *Statistical anomaly detection:* Event counts of different types of events will enable us to obtain a view of the expected number of each type of service requests over a specified service interval. A departure from this expectation can then be detected as an anomaly.
- *Pattern based misuse detection:* Pattern based misuse detection seeks to detect abnormalities by observing known patterns of anomalous behavior such as calls originating from a given set of numbers. Analysis of the events from a single session can enable us to observe such known patterns and detect intentional or unintentional misuse.

In addition to the above three types of analysis, various other types are possible, namely, provisioning and capacity planning, assessing “what-if” scenarios, and evaluating competing alternative implementations of a service.

The event collection, aggregation and analysis framework will follow a loosely coupled, distributed system architecture. In general, the ECAs will be hosted remotely from the platform supporting the JAIN APIs. The JAIN platform will notify the ECAs of the relevant events using a middleware technology such as Java RMI [4]. The ECAs will process the incoming raw stream of JAIN events and generate the two types of aggregated events described earlier. These aggregated events will be posted on a “whiteboard”. Various types of analysis engines will obtain these aggregated events as per their needs from the whiteboard. Several technologies such as JavaSpaces [5] and Java Messaging Service [6] can be used to implement the whiteboard. Use of a whiteboard for communication between ECAs and analysis engines enables loose coupling between these two entities, allowing the development and deployment of the QoS assurance system in an incremental fashion, adding one analysis engine at a time. This also offers maximum flexibility for future enhancements and additions. A schematic of the proposed system is shown in Figure 1.

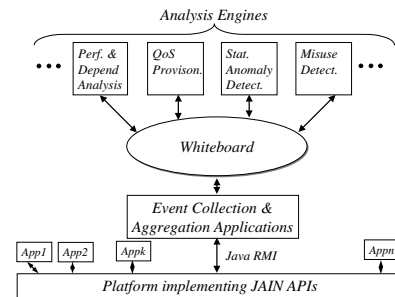


Figure 1. Architecture of the QoS assurance framework

## 4 Conclusions

In this paper we have described an architecture for QoS assurance of NGN applications using open, standard APIs. Reusing the APIs which were originally designed to reduce the application development time for the purposes of QoS assurance provides an integrated approach to rapid service creation as well as to measurement and prediction of the quality of these services. We note that QoS assurance as described here is based on the use of application-layer information as opposed to network-layer data or resource consumption data. Since the application-layer data has semantic information associated with it, the results obtained from the analyses of such data can provide a view of the quality of the service provided by the applications as perceived by the user. Service-level impact has to be inferred from the analysis of the data collected from the other layers. However, we note that certain types of analyses are possible only using data collected from the other layers, and hence application-layer QoS assurance should be viewed as complementary to QoS assurance mechanisms based on other types of information.

## References

- [1] <http://www.3gpp.org>
- [2] “The JAIN APIs: Integrated network APIs for the Java platform” <http://java.sun.com/products/jain>
- [3] Parlay Group. “Parlay API Spec. 1.2”. <http://www.parlay.org>, Sept. 1999.
- [4] <http://java.sun.com/products/jdk/rmi>
- [5] <http://java.sun.com/products/javaspaces>
- [6] <http://java.sun.com/products/jms>