A QoS Based Allocation and Routing Algorithm*

Li Zhongwen Xiong Guangze Liu Jinde

(College of Computer Science and Technology, UEST of China Chengdu 610054)

Abstract In this paper, a QoS scheme is given to support applications with QoS requirements under resource constraint in open system. Specifically, a series of abstract structure, describing characteristics of end-to-end applications and system resources, as well as algorithms supporting the realization of QoS, are designed in this paper. The scheme addresses common problems found in open systems supporting QoS guarantees, such as lack of harmony QoS expressiveness, ect. In order to let as many as possible applications get system service at the same time, the scheme also supports QoS dynamic degradation.

Key words open system; quality of service; dynamic QoS degradation; resource redistribution

Many applications have quality-of-service (QoS) requirements, such as distributed multimedia and real-time control systems. This is because their users always have requirements for stability and synchronism of video and audio, etc. Those requirements are users' QoS requirements that are provided to system in parameter form by users. Naturally, those parameters are called QoS parameters. Although application based on QoS in open systems always span several resources, even platforms, users are only interested in final results. Therefore, QoS requirements are end-to-end.

Generally speaking, most of the researches in QoS are not end-to-end, including many sorts of applications such as distributed multimedia. Those researches often focus on one or several aspect of the system[1,2]. At the first blush, the work on the application, resource, and middle-ware levels can be organized to provide end-to-end QoS. Unfortunately, most of them cannot yet be coordinated to provide that function because each is based upon different definitions, protocols, and models. Therefore, some researches in QoS are done directly from end-to-end perspective [3]. However their end-to-end QoS work aims at special application and special network (the most base ATM). Ref. [4] overcome those drawbacks, which includes three components: QoS application, QoS agent and resource reservation system. In addition, it employs the resource reservation protocols available in the target network and operating system. However, it fails to design its components in detail and describe how to organize those protocols to support the whole system. It also does not support QoS dynamic degradation, which is to be discussed in this paper.

1 QoS Architecture

The position ship of open system and QoS is shown in Fig.1. Generally speaking, open system includes several platforms[5] and it can support different kind of applications. According to those characteristics, our QoS architecture includes two components: user service platform and

Received March 15, 2000
* The project supported by National Defense Research Foundation

resource service platform, which is used to hide special characteristics of applications and system resources, respectively. Fig. 2 shows the external view of our QoS architecture.

2 User Service Platform

User service platform sends application with legal QoS parameters to a special application agent that it belongs to, such as video conferencing agent, etc. And application agent uses translation rules [4] to express application's QoS parameters with general form and draws application streaming graph Ap-S-Graph.

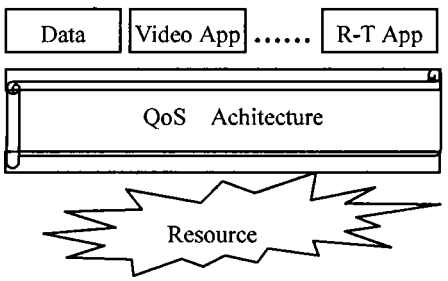

Fig. 1 QoS architecture in open system

Ap-S-Graph captures the application specifications at a logical level. It is a directed graph $GA=\{T,E\}$, where $T=\{t_1,t_2,\cdots,t_n\}$ represents a set of n processing steps and E represents the set of directed edges between processing steps. t_n is called destination processing step. Each graph node represents a processing step. The data flow from processing step t_i to processing step t_j is represented by a directed edge (t_i,t_j) between them. Each node in this graph is labeled with a vector called QoS_para:

$$\text{QoS\_para}(t_i)=[Q_1, Q_2, Q_3]$$

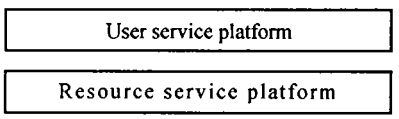

Fig. 2 The external view of QoS architecture

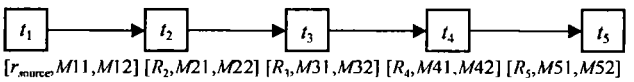

Fig.3 Example application streaming graph

For t_1, Q_1 represents the resource where the application is beginning, which is often called source resource, labeled with r_{source}. However, for $t_2 \cdots t_n$, Q_1 is the type of resource which processing step can execute. For every processing step, Q_2 and Q_3 represent its max and min QoS requirements. Fig.3 shows the structure of a simplified end-to-end Ap-S-Graph.

3 Resource Service Platform

The main function of resource service platform is to hide physic characteristics of resource and to assign resource to application. If it is necessary, resource service platform must adjust resource allocation between application's max and min QoS parameters to realize QoS degradation and let more applications get system service at the same time. To do this, three abstract structures and two resource allocation algorithms are designed in this section.

3.1 Abstract Structures

The resource distribution graph Re-D-Graph=$\{R, E\}$ is an undirected graph, where $R=\{r_1,r_2,\cdots,r_m\}$ represents a set of m resources and E represents the set of edges which are the connection between two resources. Each graph node, representing a resource in this graph, is labeled with a vector called Res_para: $\text{Res\_para}(r_i)=[re_1, re_2, re_3, re_4]$.

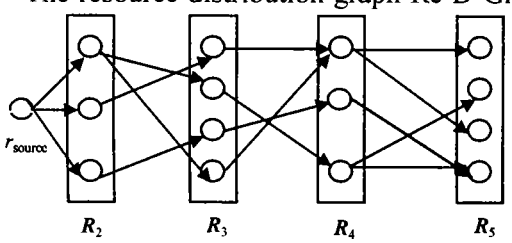

Fig.4 Re-L-Graph of Fig.3

re_1 represents the type of r_i. re_2 represents the QoS that r_i can provide now. re_3, re_4 represent the max, min QoS that r_i can provide, respectively.

Modifying Re-D-Graph, we can get resource allocation graph Re-A-Graph which records resource allocation of every application in system: for each edge (r_i, r_j) in Re-A-Graph, if resource r_i and r_j are assigned to the same application, we make a special symbol on it.

Resource Layer Graph=$\{r_{source}, G, E\}$ is a directed graph. Here $G=\{\boldsymbol{R}_1,\boldsymbol{R}_2,\cdots,\boldsymbol{R}_w\}$ represents a set of w resource type which application can execute. We use rectangle to express them. Of course, $\boldsymbol{R}_i$ is a vector and its every element represents a resource which belongs to the same resource type $\boldsymbol{R}_i$. We use circle to express these elements. $\boldsymbol{E}$ represents the set of directed edges. Of course, edge is the connection between the resources and arrow is the direction of the data flow. Fig.4 shows the Re-L-Graph of Fig.3.

3.2 Resource Allocation Algorithm

Unlike scheduling algorithm[6], our algorithm considers resource constraint, which includes two sub-algorithm: Co-D and Do-D. When system resources meet min QoS requirement, algorithm Co-D can assign resources to application successfully, otherwise algorithm Ro-D must be used to realize QoS degradation.

3.2.1 Co-D Algorithm

It is assumed that application A requires system service. Then Co-D tries to find a path from resource r_{source} to destination resource which the destination processing step of A can execute in Re-L-Graph of A. The main process of Co-D is as follows:

1) According to Re-D-Graph and Ap-S-Graph of A, Co-D draws the Re-L-Graph of A.

2) For each destination resource, repeat step 3), 4).

3) Use a set of temporary variable TEMP to record re_2 of resources except r_{source} in Re-L-Graph of A.

4) For each directed edge (r_i, r_j) of Re-L-Graph of A, assume that the weigh on the edge is the reciprocal of r_j's TEMP. Using Dijkstra's shortest path algorithms in Re-L-Graph of A, we can find out a shortest path from r_{source} to destination resource. Then, we subtract QoS that is allocated to A temporarily from its TEMP.

5) The shortest path among these paths that we get during step from 2) to 4) is the path we need.

6) If the work above succeeds, then allocate resources on the shortest path to processing steps, respectively, and change re_2 of these resources. Otherwise, call Ro-D to realize QoS degradation.

3.2.2 Do-D algorithm

Ro-D must consider two problems: which application should be degraded and how should the degradation be done? Assume that applications $A_1,A_2,\cdots,A_n$ are in system when new application A fails to request resource. Now, algorithm Ro-D is called. Name resources causing failure as clash resources, and assume priority of $A_1,A_2,\cdots,A_m$ ($m\leqslant n$) is lower than that of A, therefore, the degradation work is beginning from the lowest priority application among them. The process of Ro-D is as follows:

1) $A_1,A_2,\cdots,A_m$ are queued from low priority to high priority. Applications that share the same priority are queued according to the quantity of clash resource from low to high.

2) If this queue is empty, Ro-D is over and the system tells the user to reduce its QoS requirement, or fetch the left application and label it with B. According to Re-D-Graph, Ro-D finds out all of clash resources which we assume they are $r_i, r_j,\cdots,r_m$ and they belong to resource type $\boldsymbol{R}_i, \boldsymbol{R}_j,\cdots,\boldsymbol{R}_m$, respectively.

3) Deprive temporarily resources $r_i, r_j,\cdots,r_m$ from application B. Now, if the system still cannot meet the min QoS requirement of A, go to step 2), or next step.

4) According to the Re-L-Graph of B, allocate temporarily resource r_i' ($r_j',\cdots,r_m'$) which belongs to $\boldsymbol{R}_i$, ($\boldsymbol{R}_j,\cdots,\boldsymbol{R}_m$) and provides the smallest QoS for B among the others in $\boldsymbol{R}_i$, ($\boldsymbol{R}_j,\cdots,\boldsymbol{R}_m$). Then find a path which passes r_i' ($r_j',\cdots,r_m'$). If there are many paths, choose one that enables the resources to change B to minimize. Then, allocate resources to B along that path and allocate resources to B. If 4) succeeds, Ro-D is over, or go to step 5).

5) Recovery resources distribution of A, go to step 2).

4 Conclusions

A QoS mechanism, which includes three abstract structures and two algorithms, is designed to support application under resource constraint in open systems in this paper. Our QoS architecture is highly extensible and platform-independent. In the future, we will discuss QoS in the area of real time system.

References

1 Wang Dongxia, Dou Wenhua, Zhou Xingming. The QoS architecture of the survivable high-speed multimedia network. Journal of China Institute of Communications, 1999, 20(5): 84~88

2 Dharanikota S, Maly K. Quanta: quality of service architecture for native TCP/IP over ATM networks. Computer Science Department of Old Dominion University, Technical report:TR-96-01, 1996,(5): 9~37

3 Lazar A A, Lim K S, Marcocini F. Realizing a foundation for programmability of ATM network with the binding architecture. IEEE Journal of Selected Areas in Communication, 1996,9(7): 1 214~1 227

4 Siqueira F, CahillSiqueira V. Quartz: supporting QoS-constrained services in heterogeneous environments. Proceedings of the 19th IEEE Real Time Systems Symposium (RTSS'98), Work in Progress Session, Madrid, Spain, 1998: 23~34

5 苏 森，唐雪飞，刘锦德. 面向对象的互操作技术. 电子科技大学学报, 1998, 27(1): 90~94

6 王志平，熊光泽. 实时调度算法研究. 电子科技大学学报, 2000, 29(2): 205~208

基于分配和寻径算法的 QoS 方案*

黎忠文** 熊光泽 刘锦德

(电子科技大学计算机学院 成都 610054)

【摘要】 讨论了一种基于开放式环境的 QoS 实现方案，它是在开放系统中资源有限的条件下，为具有 QoS 需求的应用提供服务。该方案设计了一系列用来描述端到端应用、系统资源特征的抽象结构和 QoS 的实现算法。其结果解决了开放系统中为支持 QoS 所引起的诸如系统各层中缺乏 QoS 的协调表达等一些重要问题。系统还支持 QoS 动态重协商，能够同时为尽可能多的应用提供服务。

关 键 词 开放系统； QoS； QoS 动态降级； 资源重分配

中图分类号 TP393

2000年3月15日收稿

* 总装备部预研基金资助项目

** 女 30岁 博士生