Network Working Group
Request for Comments: 2676
Category: Experimental

G. Apostolopoulos
D. Williams
IBM
S. Kamat
Lucent
R. Guerin
UPenn
A. Orda
Technion
T. Przygienda
Siara Systems
August 1999

QoS Routing Mechanisms and OSPF Extensions

Status of this Memo

Copyright Notice

Abstract

This memo describes extensions to the OSPF [Moy98] protocol to support QoS routes. The focus of this document is on the algorithms used to compute QoS routes and on the necessary modifications to OSPF to support this function, e.g., the information needed, its format, how it is distributed, and how it is used by the QoS path selection process. Aspects related to how QoS routes are established and managed are also briefly discussed. The goal of this document is to identify a framework and possible approaches to allow deployment of QoS routing capabilities with the minimum possible impact to the existing routing infrastructure.

In addition, experience from an implementation of the proposed extensions in the GateD environment [Con], along with performance measurements is presented.

Table of Contents

1. Introduction

In this document, we describe a set of proposed additions to the OSPF routing protocol (these additions have been implemented on top of the GateD [Con] implementation of OSPF V2 [Moy98]) to support Quality-of-Service (QoS) routing in IP networks. Support for QoS routing can be viewed as consisting of three major components:

1. Obtain the information needed to compute QoS paths and select a path capable of meeting the QoS requirements of a given request,

2. Establish the path selected to accommodate a new request,

3. Maintain the path assigned for use by a given request.

Although we touch upon aspects related to the last two components, the focus of this document is on the first one. In particular, we discuss the metrics required to support QoS, the extension to the OSPF link state advertisement mechanism to propagate updates of QoS metrics, and the modifications to the path selection to accommodate QoS requests. The goal of the extensions described in this document is to improve performance for QoS flows (likelihood to be routed on a path capable of providing the requested QoS), with minimal impact on the existing OSPF protocol and its current implementation. Given the inherent complexity of QoS routing, achieving this goal obviously implies trading-off "optimality" for "simplicity", but we believe this to be required in order to facilitate deployment of QoS routing capabilities.

In addition to describing the proposed extensions to the OSPF protocol, this document also reports experimental data based on performance measurements of an implementation done on the GateD platform (see Section 4).

1.1. Overall Framework

We consider a network (1) that supports both best-effort packets and packets with QoS guarantees. The way in which the network resources are split between the two classes is irrelevant, except for the assumption that each QoS capable router in the network is able to dedicate some of its resources to satisfy the requirements of QoS packets. QoS capable routers are also assumed capable of identifying and advertising resources that remain available to new QoS flows. In addition, we limit ourselves to the case where all the routers involved support the QoS extensions described in this document, i.e., we do not consider the problem of establishing a route in a heterogeneous environment where some routers are QoS-capable and others are not. Furthermore, in this document, we focus on the case of unicast flows, although many of the additions we define are applicable to multicast flows as well.

We assume that a flow with QoS requirements specifies them in some fashion that is accessible to the routing protocol. For example, this could correspond to the arrival of an RSVP [RZB+97] PATH message, whose TSpec is passed to routing together with the destination address. After processing such a request, the routing protocol returns the path that it deems the most suitable given the flow’s requirements. Depending on the scope of the path selection process, this returned path could range from simply identifying the best next hop, i.e., a hop-by-hop path selection model, to specifying all intermediate nodes to the destination, i.e., an explicit route model. The nature of the path being returned impacts the operation of the path selection algorithm as it translates into different requirements for constructing and returning the appropriate path information. However, it does not affect the basic operation of the path selection algorithm (2).

For simplicity and also because it is the model currently supported in the implementation (see Section 4 for details), in the rest of this document we focus on the hop-by-hop path selection model. The additional modifications required to support an explicit routing model are discussed in appendix D, but are peripheral to the main focus of this document which concentrates on the specific extensions to the OPSF protocol to support computation of QoS routes.

In addition to the problem of selecting a QoS path and possibly reserving the corresponding resources, one should note that the successful delivery of QoS guarantees requires that the packets of the associated "QoS flow" be forwarded on the selected path. This typically requires the installation of corresponding forwarding state in the router. For example, with RSVP [RZB+97] flows a classifier entry is created based on the filter specs contained in the RESV message. In the case of a Differentiated Service [KNB98] setting, the classifier entry may be based on the destination address (or prefix) and the corresponding value of the DS byte. The mechanisms described in this document are at the control path level and are, therefore, independent of data path mechanisms such as the packet classification method used. Nevertheless, it is important to notice that consistent delivery of QoS guarantees implies stability of the data path. In particular, while it is possible that after a path is first selected, network conditions change and result in the appearance of "better" paths, such changes should be prevented from unnecessarily affecting existing paths. In particular, switching over to a new (and better) path should be limited to specific conditions, e.g., when the initial selection turns out to be inadequate or extremely "expensive". This aspect is beyond the scope of QoS routing and belongs to the realm of path management, which is outside the main focus of this document. However, because of its potentially significant impact on the usefulness of QoS routing, we briefly outline a possible approach to path management.

Avoiding unnecessary changes to QoS paths requires that state information be maintained for each QoS path after it has been selected. This state information is used to track the validity of the path, i.e., is the current path adequate or should QoS routing be queried again to generate a new and potentially better path. We say that a path is "pinned" when its state specifies that QoS routing need not be queried anew, while a path is considered "un-pinned" otherwise. The main issue is then to define how, when, and where path pinning and un-pinning is to take place, and this will typically depend on the mechanism used to request QoS routes. For example, when the RSVP protocol is the mechanism being used, it is desirable that path management be kept as synergetic as possible with the existing RSVP state management. In other words, pinning and un-pinning of paths should be coordinated with RSVP soft states, and structured so as to require minimal changes to RSVP processing rules. A broad RSVP-routing interface that enables this is described in [GKR97]. Use of such an interface in the context of reserving resources along an explicit path with RSVP is discussed in [GLG+97]. Details of path management and a means for avoiding loops in case of hop-by-hop path setup can be found in [GKH97], and are not addressed further in this document.

1.2. Simplifying Assumptions

In order to achieve our goal of minimizing impact to the existing protocol and implementation, we impose certain restrictions on the range of extensions we initially consider to support QoS. The first restriction is on the type of additional (QoS) metrics that will be added to Link State Advertisements (LSAs) for the purpose of distributing metrics updates. Specifically, the extensions to LSAs that we initially consider, include only available bandwidth and delay. In addition, path selection is itself limited to considering only bandwidth requirements. In particular, the path selection algorithm selects paths capable of satisfying the bandwidth requirement of flows, while at the same time trying to minimize the amount of network resources that need to be allocated, i.e., minimize the number of hops used.

This focus on bandwidth is adequate in most instances, and meant to keep initial complexity at an acceptable level. However, it does not fully capture the complete range of potential QoS requirements. For example, a delay-sensitive flow of an interactive application could be put on a path using a satellite link, if that link provided a

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE