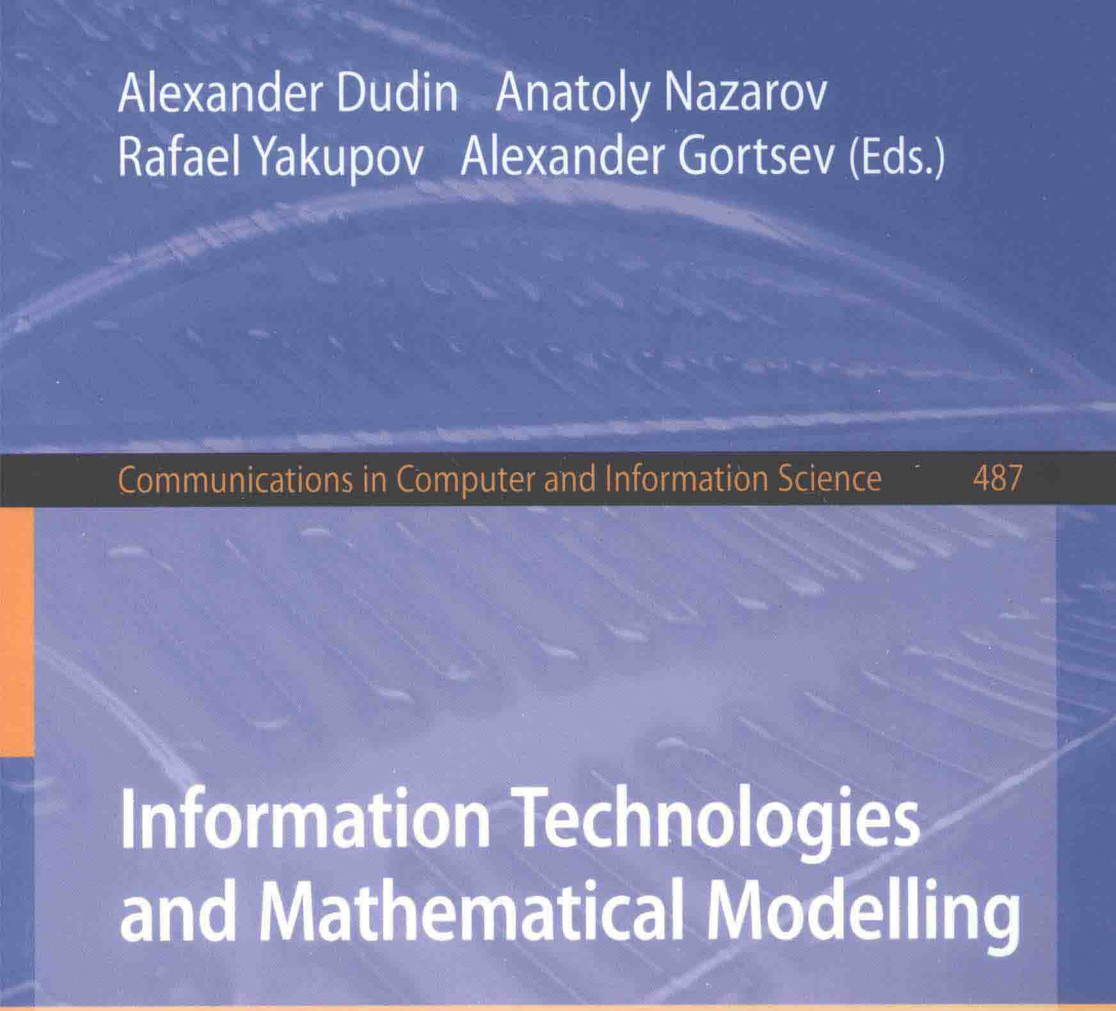

Alexander Dudin Anatoly Nazarov
Rafael Yakupov Alexander Gortsev (Eds.)

Communications in Computer and Information Science 487

# Information Technologies and Mathematical Modelling

13th International Scientific Conference, ITMM 2014
named after A.F. Terpugov
Anzhero-Sudzhensk, Russia, November 20–22, 2014
Proceedings

Springer

Alexander Dudin Anatoly Nazarov
Rafael Yakupov Alexander Gortsev (Eds.)

# Information Technologies and Mathematical Modelling

13th International Scientific Conference, ITMM 2014
named after A.F. Terpugov
Anzhero-Sudzhensk, Russia, November 20-22, 2014
Proceedings

Springer

Volume Editors

Alexander Dudin
Belarusian State University
Minsk, Belarus
E-mail: dudin@bsu.by

Anatoly Nazarov
Tomsk State University, Russia
E-mail: nazarov.tsu@gmail.com

Rafael Yakupov
Kemerovo State University
Anzhero-Sudzhensk, Russia
E-mail: yrt@asf.ru

Alexander Gortsev
National Research Tomsk State University, Russia
E-mail: gam@fpmk.tsu.ru

ISSN 1865-0929 e-ISSN 1865-0937
ISBN 978-3-319-13670-7 e-ISBN 978-3-319-13671-4
DOI 10.1007/978-3-319-13671-4
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014955253

© Springer International Publishing Switzerland 2014
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in ist current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

*Typesetting:* Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

# Preface

The series of scientific conferences "Information Technologies and Mathematical Modelling" (ITMM) started in 2002. In the beginning, it had the status of a national conference or national conference with international participation. In 2012, it was named after A. F. Terpugov, an outstanding scientist of the Tomsk State University, a leader of the famous Siberian school on applied probability, who was one of the first organizers of the conference.

Today we observe the process of globalization of the sciences involving Russian scientists into global science scene. This is why the ITMM conference was given an international status.

Traditionally, the conference has from eight to 12 sections in various fields of mathematical modelling and information technologies. A strong focus is on applied problems in education, economics, technology and management. Throughout the years, the sections on probabilistic methods and models, queueing theory, telecommunication systems, and software engineering have been the most popular at the conference.

This volume presents new results in the theory of random processes, methods of study of queueing systems, probabilistic methods and models, analysis of telecommunication systems and networks, software engineering, and others. It is targeted at specialists in probabilistic theory, random processes, mathematical modeling, as well as engineers engaged in logical and technical design and operational management of telecommunication and computer networks, contact centers, databases, software design, etc.

November 2014 Anatoly Nazarov

# Organization

The ITMM conferences are organized by the Anzhero-Sudzhensk Branch of Kemerovo State University together with Kemerovo State University, National Research Tomsk State University, Kemerovo Scientific Centre of SB RAS (the Siberian Branch of Russian Academy of Sciences), and the Institute of Computational Technologies of SB RAS since 2002.

In 2014 the XIII International Scientific Research and Practice Conference was organized named after A. F. Terpugov "Information Technologies and Mathematical Modelling."

## International Program Committee

A. Dudin, Belarus, Chair
A. Nazarov, Russia, Co-chair
R. Yakupov, Russia, Co-chair
I. Atencia, Spain
S. Chakravarthy, USA
B. Choi, Korea
T. Czachorski, Poland
N. Danilov, Russia
D. Efrosinin, Austria
A. Gortsev, Russia
K. Al-Begain, UK
Yu. Kharin, Belarus
C. Kim, Korea
A. Krishnamurthy, India
U. Krieger, Germany
A. Latkov, Latvia
Q. Li, China
G. Medvedev, Belarus
A. Melikov, Azerbaijan
R. Nobel, The Netherlands
E. Orsingher, Italy
M. Pagano, Italy
G. Saporta, France
K. Samuylov, Russia
Y. Shokin, Russia
J. Sztrik, Hungury
N. Temirgaliev, The Netherlands
H. Tijms, The Netherlands
O. Tikhonenko, Poland

## Local Organizing Committee

A. Dudin, Chair
A. Nazarov, Co-chair
R. Yakupov, Co-chair
O. Galazhinskaya
I. Garayshina
B. Gladkikh
E. Glukhova
R. Ivanovskiy
V. Ivnitskii
V. Kochetkov
Yu. Kostyuk
T. Lyubina
A. Moiseev
S. Moiseeva
V. Poddubnyi
V. Rykov
S. Senashov
A. Shkurkin
S. Suschenko
K. Voytikov
O. Zmeev

# Table of Contents

# A Novel Framework for the Design and Development of Software Routers

Davide Adami[1], Stefano Giordano[2], Michele Pagano[2], and Luis G. Zuliani[3]

[1] CNIT Research Unit - University of Pisa, Pisa, Italy
[2] Dept. of Information Engineering, University of Pisa, Pisa, Italy
[3] Link Protect GmbH, Münchner Straße 92, 85614 Kirchseeon, Germany
`m.pagano@iet.unipi.it`

**Abstract.** Flexibility and programmability are key features of open networking platforms to effectively design, develop and assess new Future Internet architectures. This paper introduces a novel framework, based on open source software and PC hardware, that defines all the building blocks necessary to provide a full set of advanced networking capabilities (including QoS support and Traffic Engineering). The framework implementation also takes into account software reuse by facilitating maintenance and customization of the network protocol stack in software routers. The proposed open framework is also available as live distributions, which allow network designers to take advantage of its capabilities with a short learning curve.

**Keywords:** software routers, open frameworks, QoS, traffic engineering.

## 1 Introduction

Originally conceived as an experimental packet-switched network, the Internet has become a global communication infrastructure. As scalability and robustness are critical requirements of this evolutionary process, academia and industry are increasingly relying on field trials in order to develop and assess novel network solutions [16]. In this context, PCs started to be used as platforms for the development of router prototypes thanks to the availability of powerful and cheap commodity hardware and the wide spread of open source software. When playing such a role, PCs are called SRs (Software Routers) [11].

Usually, SRs are based on open source, Unix-like OSes, such as GNU/Linux and BSD variants. These systems, in addition to routing and forwarding capabilities, may offer complete solutions to filter and manipulate layer 2 and above information, with a level of flexibility which is comparable only to costly, commercial routers. Moreover, SRs have programmability features, enabling the development of third party extensions and completely new functionalities as well as fine-tuning of existing ones.

During the last years, SRs have started playing an important role even in the market. Leading network vendors are exploring the SOHO (Small Office/Home

A. Dudin et al. (Eds.): ITMM 2014, CCIS 487, pp. 1–10, 2014.

Office) market with embedded SRs, using mixed open and closed source solutions (for the OS and protocol stack, respectively [9]). More recently, SRs are being used for the deployment of NGNs (Next Generation Networks) and also as learning tools (see, for instance, [6]). Despite their increasing popularity, SRs have relevant weaknesses. On the one hand, off-the-shelf SRs are often shipped with closed networking stack that limits (or even prevents) the development of new capabilities. On the other hand, completely open SRs are often developed for specific purposes, like the optimization of an existing routing protocol or the design of a brand new scheduler. These enhancements tend to be developed as ad-hoc, intrusive hacks in the original source code, making its maintenance and reuse a complex and time-consuming task. QoS (Quality of Service) support is a concrete example of this deficiency: there is a large number of open source tools to enforce QoS, but a clear lack of orchestrating procedures to achieve a specific behaviour.

Taking into account the above-mentioned weaknesses, this paper deals with the design and development of a novel open framework for SRs with advanced network functionalities. The goal is twofold: to provide flexible tools for the configuration and management of QoS–aware networks and to allow the evaluation of NGN architectures through fully-operational trials. In more detail, the framework is exclusively based on free and open source software, and delivers out-of-the-box automatic management of virtual circuits and traffic differentiation, by effectively combining the DS (DiffServ) and MPLS (MultiProtocol Label Switching) IETF architectures, still the most advanced QoS and TE (Traffic Engineering) solutions for IP networks.

Indeed, MPLS [18] allows the management of pseudo-virtual circuits over a wide range of layer 2 and 3 protocols, including IP: in an MPLS network the IP header is used exclusively at edge routers to map packets into a FEC (Forwarding Equivalence Class). All packets associated to a given FEC are “tagged” with an identical label; for instance, in an IP over Ethernet network, labels are inserted between the Ethernet and the IP packet headers, in a structure called Shim Header. Inside an MPLS cloud, only labels are used to forward packets, according to label forwarding tables (much simpler than IP routing tables). The path that all packets belonging to the same FEC use to traverse the network is called LSP (Label Switched Path) and the network nodes are denoted as LSRs (Label Switching Routers). TE capabilities of MPLS allow to avoid congestion both in the steady–state and in failure scenarios by establishing LSPs along links with available bandwidth and providing resilience in case of failure by means of built–in mechanisms, such as link protection and fast reroute.

DS [12] maps the incoming packets in classes, and resource reservation is performed on a per-class basis. These service classes, known as PHBs (Per Hop Behaviours), specify how packets must be treated by a router (i.e., how a router must distribute its resources, prioritizing certain classes on detriment of others). Packets at ingress routers are classified in one of the defined PHBs: EF (Expedited Forwarding), AF (Assured Forwarding) and Default PHB, typically used for Best Effort (BE) traffic, by setting the suitable DSCP (DS Code Point) in the IP header TOS field, renamed DS field. While DS assures data delivery with "relative" QoS, there is no TE control in DS networks.

In more detail, two different solutions, DiffServ over MPLS [15] and DS-TE (DiffServ-aware MPLS TE, [13,14]) have been standardized to make DS and MPLS interwork. Thus, DS packets are transported inside LSPs and the network is enabled to perform TE while QoS is guaranteed. More specifically, DiffServ over MPLS allows the creation of TE–LSPs that carry DS marked IP packets from a single or multiple classes (L-LSP and E-LSP, respectively), with assured bandwidth. However, the main problem is that such integration of DS and MPLS is not able to guarantee end-to-end QoS to TE-LSPs on a per-class basis under any operating conditions because QoS is only provided at node level and MPLS is unaware of traffic classes. DS-TE makes MPLS aware of traffic classes, allowing end-to-end resource reservation with traffic class granularity and providing the fault tolerance of MPLS at traffic class level. To achieve such differentiated treatment, in [14] the concept of CT (Class Type) is introduced. A CT is the set of traffic trunks crossing a link that is governed by a specific set of BCs (Bandwidth Constraints). CTs are used for link bandwidth allocation, constraint based routing and admission control.

The rest of the paper is organized as follows. Section 2 describes the general features of the proposed open framework architecture; then, Section 3 and Section 4 detail the Data Plane and Control/Management Plane functionalities, respectively. Finally, conclusions are drawn in Section 5.

## 2 Open Framework Architecture and Features

Our open framework (see Figure 1) is based on a sharp separation among the functionalities of data, control and management planes. The next subsections detail these functionalities and provide an overview of the key features of DS-MPLS nodes, which will be denoted in the following as DS-LSRs.

### 2.1 Data Plane

The data plane is responsible for data forwarding in accordance with the rules established by the control and management planes. Its components are briefly described in the following.

1. **Traffic Control:** enforces the QoS of data flows. It is responsible for performing label–based packet switching and includes the following entities:
   - Policy Enforcer: available in edge DS-LSRs, it filters the incoming packets, admitting only those from authorized parties and at rates that are in conformance with Service Level Agreements (SLAs).
   - Classifier: processes incoming packets in a pre-routing phase, preparing them to be forwarded using class-specific resources.
   - Lookup: instead of destination-based routing, it uses multiple routing tables to route packets in LSPs, also taking into account other parameters such as DS classes.

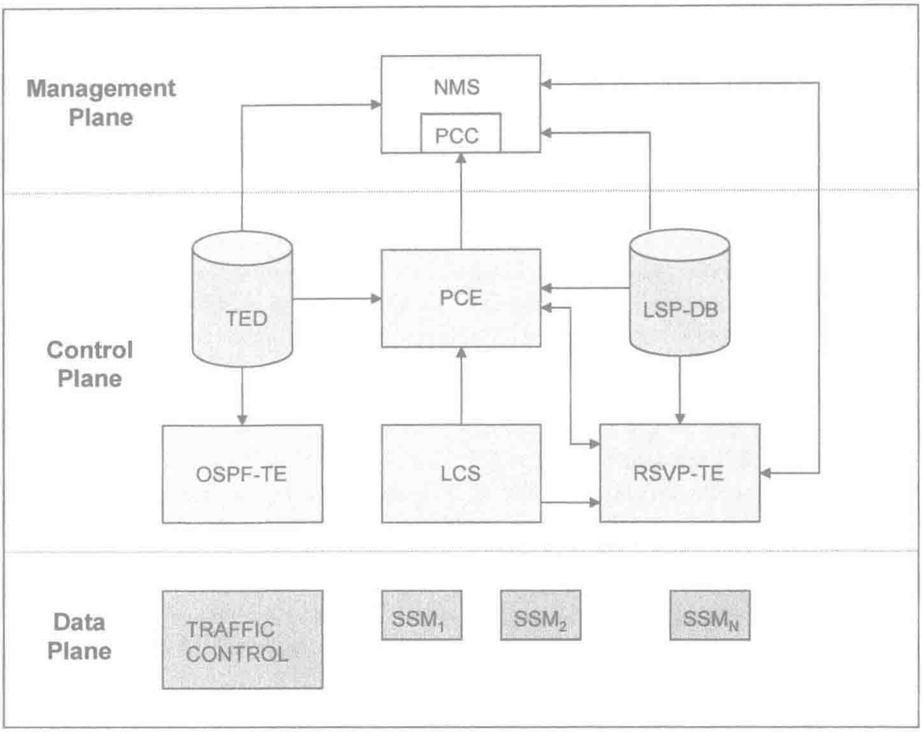

**Fig. 1.** DS-MPLS Open Framework architecture

   - Forwarder: it uses a set of schedulers to forward MPLS packets according to DS classes specification on a per-LSP basis.
2. **SSMs (Service-Specific modules):** are external programs developed to bring intelligence to the network. This way, other than just forwarding packets, SRs can elaborate data, simplifying the creation of upper layer applications and overlays. SSMs can perform basic services, such as cryptography and compression, as also more advanced functions, such as video transcoding.

### 2.2 Control Plane

The key part of the open framework architecture is the control plane, that has been completely designed and implemented to support both DiffServ over MPLS and DS-TE. Roughly speaking, it collects the network state information, computes the suitable paths for the user traffic and sets up the corresponding LSPs. Its main components are the following:

1. **TED (Traffic Engineering Database):** describes the properties and state of network links and routers. It is defined by using XML.
2. **LSP-DB (LSP Database):** details the characteristics of the active LSPs. As the TED, it is defined by using XML.

3. **PCE (Path Computation Element):** is responsible for processing LSP setup requests (in case of bidirectional circuit requests, the PCE computes a path that is able to accommodate two LSPs in opposite directions). This module may employ different Path Computation Algorithms, working in a centralized or distributed manner [10]. In case of DS-TE architecture, it supports MAM (Maximum Allocation Model) and RDM (Russian Dolls Model) bandwidth allocation models [17]. After the path computation phase, LSPs are established.
4. **LCS (LSR Control System):** configures all routers involved in LSP setup or teardown requests. It uses a secure, centralized approach to contact and configure DS-LSRs. The LCS is formed by two distinct components: the LCA (LSR Control Agent) and the LLEA (LSR Local Enforcement Agent). The LCA is always in the same node as the PCE, no matter if the node is a DS-LSR or a host external to the network. It is responsible for receiving the LSP information from the PCE and translating it into configuration commands for all the routers in the path. The LCA controls all the DS-LSR specific information such as the labels to be configured in the interfaces. Once the set of commands that each router must execute to configure the LSP(s) are defined, the LCA contacts the LLEAs entities in every DS-LSR along the path. Each LLEA is responsible for locally executing the set of commands to configure the LSP(s), and then to report back to the LCA the state of the operation.
5. **RSVP-TE:** is used to perform robust, automatic provisioning of LSPs.
6. **OSPF-TE:** is responsible for maintaining, updating and synchronizing the TED across the DS domain.

### 2.3 Management Plane

The management plane provides the system interfaces to control and query all other entities. It consists of the following components:

1. **NMS (Network Management System):** provides graphical and command-line user interfaces for network resources administration. NMS users can setup and teardown LSPs, manage FEC-LSP associations, and analyse the operational state of DS-LSRs, links, protocols and virtual circuits.
2. **PCC (Path Computation Client):** generates LSP setup requests and sends them to the PCE.

## 3 Data Plane Functionalities

The open framework data plane uses both standard GNU/Linux routing and traffic control functionalities, as well as a number of experimental tools to add new advanced capabilities, such as label-switching routing. Usually, these external add-ons require intrusive kernel patching and hacking of userspace tools. In the open framework to build up a full-featured data plane, customized patches have been developed to circumvent version mismatches.

### 3.1 LSP Establishment

MPLS support in the open framework is granted by the MPLS for Linux project [4]. It consists of several patches in the Linux kernel and in the following userspace tools: iproute2 (traffic control in Linux) [2], iptables (layer 3 packet filtering) [5], and ebtables (layer 2 cell/frame filtering) [3]. A new tool called `mpls` allows to manage LSPs.

To establish E-LSP and L-LSPs, our open framework takes advantage of an `mpls` tool feature that allows to set the EXP field of the shim header and the TC index field of the packet buffer descriptor (internal to the DS-LSR) in function of the DSCP value in the IP header. The TC index plays a key role when assigning MPLS packets to schedulers.

To setup an LSP, the following steps are mandatory:

- **ingress and core DS-LSRs:** creation of an entry in the NHLFE (Next Hop Label Forwarding Entry) table, with the corresponding mapping from DSCP to EXP and TC index;
- **core and egress DS-LSRs:** creation of an entry in the ILM (Incoming Label Map) table, to map incoming and outgoing labels;
- **core DS-LSRs:** cross-connection between ILM and NHLFE entries.

### 3.2 FEC to LSP Binding

Since the standard GNU/Linux networking stack was not designed with MPLS support in mind, routing decisions can only be taken based on the information contained in the IP routing table. The availability of a single routing table is a significant limitation in case TE policies should be applied and, for instance, two packets with the same source and destination IP addresses should be sent towards the destination by means of different LSPs (maybe configured on the same outgoing interface).

On the ingress DS-LSR, the FEC-to-NHLFE table allows to associate each incoming packet belonging to a specific FEC with a set of instructions (NHLFE) that indicate how to forward the packet to the next LSR. Nevertheless, due to limitations of the userspace Linux tools, IP packets can not be directly assigned to FECs at ingress DS-LSRs.

To overcome these issues, the open framework provides a great amount of flexibility regarding FEC definition. Almost every field of layers 2, 3 and 4 packet headers can be used to define a FEC. Moreover, the framework introduces multiple routing tables, whose lookup priorities are higher with respect to the default IP routing table.

In more detail, by using forward marks, each FEC is associated with a specific routing table, which usually contains a single entry related to a specific LSP (in fact, its NHLFE). This solution enables the finest QoS routing granularity, although it is also the most expensive in terms of resources. The open framework also allows the use of multiple NHLFE entries per table.

### 3.3 Hierarchical Scheduling

The open framework provides QoS support through a modular tree of hierarchical packet schedulers, available on every interface of DS-LSRs. The hierarchical scheduler tree is illustrated in Figure 2.

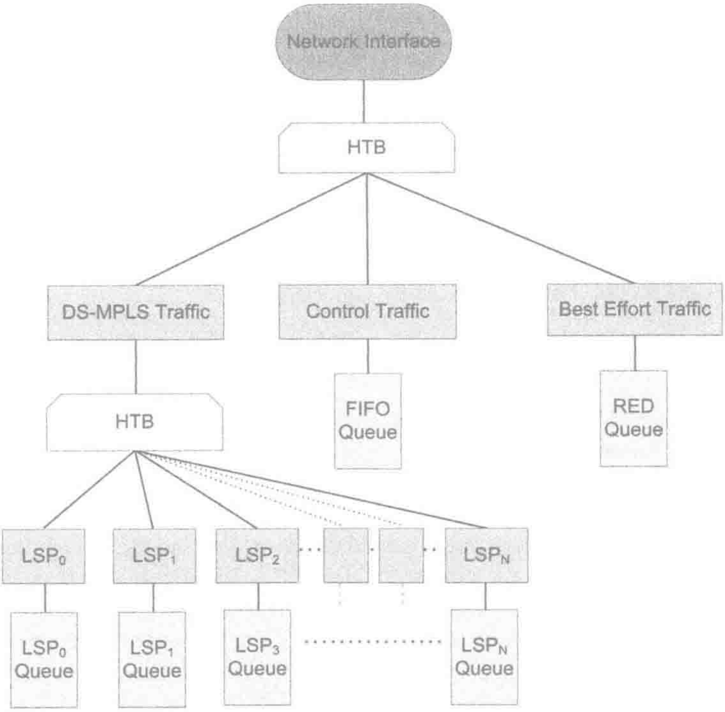

**Fig. 2.** Hierarchical scheduler tree

The reservable bandwidth for all interfaces is initially divided between three types of traffic: DS-MPLS, control and BE. By reserving a fraction of bandwidth to control traffic, the control plane is isolated from the data plane (out-of-band approach). Also, by reserving a small amount of bandwidth to BE traffic, a minimal service level is guaranteed, avoiding complete starvation of the lowest class. To enable bandwidth separation, the HTB (Hierarchical Token Bucket) [1] packet scheduler is used. While the interface bandwidth share reserved to control traffic is managed by a FIFO queue, the BE traffic is sent to a RED (Random Early Detection) queue. The DS-MPLS traffic bandwidth share is managed by another HTB scheduler, which is responsible for guaranteeing the nominal bandwidth to LSPs.

When establishing a new LSP, a specific hierarchical scheduler subtree is configured according to the type of the LSP. The default scheduler subtrees for E-LSPs and L-LSPs (carrying AF and EF traffic) are depicted in Figure 3.