Optimization of Network Traffic Flow

Martin Hrubý*
Institute of Computer Systems and Networks
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
Ilkovičova 2, 842 16 Bratislava, Slovakia
hruby@fiit.stuba.sk

Abstract

In modern backbone networks with heterogeneous traffic, providing connectivity is taken for granted. The MPLS technology is a de-facto standard in modern backbone networks because of the applications it enables, in particular L2/L3 VPNs and traffic engineering. This paper deals with traffic engineering in MPLS backbone networks. We propose a method of gathering network performance parameters and a model for optimizing the flow of network traffic based on class-based TE tunnel selection (n-CUBE). We implement this model as part of a traffic engineering system and test it in a laboratory environment by conducting two sets of experiments.

Categories and Subject Descriptors

C.2 [**Computer-communication Networks**]: Network Protocols; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

Keywords

MPLS, QoS, statistics, traffic engineering

1. Introduction

Rapid development of network applications in the last decade and their incorporation into daily lives of a significant portion of the population, have imposed a burden on existing underlying computer networks. The Internet was designed with best effort service in mind where connectivity was the most important issue. Today, connectivity is taken for granted. New approaches and technologies had to be implemented to accommodate the various applications and their diverse requirements. Bandwidth over-provisioning though still widely implemented, does not solve the issues of time-sensitive network traffic with voice or multimedia content where traffic prioritization must be deployed. To address these issues, various QoS mechanisms where proposed and implemented. In recent years we observe a push toward better utilization of existing network resources on the part of service providers, who carry ever increasing loads of traffic on their backbone networks. This initiative termed "traffic engineering" is proving to be advantageous from both a QoS perspective and an economic perspective. Our thesis deals with the issues that we face in converged networks where data, voice, and other multimedia content is transmitted over the same logical channels. A new model, n-CUBE, is proposed and implemented with the objective to optimize the flow of network traffic. First however, we introduce the two basic areas of computer networks which form the body of our analysis, namely quality of services and traffic engineering.

2. Quality of Services

In the early days of computer networking, data applications constituted the bulk of Internet traffic. These applications mostly used TCP and the objective was to deliver the traffic to its destination in the shortest time possible [4]. This approach proved to be successful and allowed the Internet to grow rapidly. Naturally over time a new generation of applications and services emerged. A need for service guarantees and a certain "quality of service" was apparent. The Internet will remain a best effort environment without the introduction of more sophisticated routing control capabilities, other than simple unconstrained shortest path algorithms [3]. Quality of Services (QoS) refers to the capability of a network to provide better service to selected network traffic over various technologies, including Frame Relay, Asynchronous Transfer Mode (ATM), Ethernet and 802.1 networks, SONET/SDH, and IP-routed networks that may use any or all of these underlying technologies [1]. There have been multiple approaches to providing QoS, including the well-known IntServ (Integrated services) and DiffServ (Differentiated services) architectures. The architecture of Integrated services does not scale well and scalability is today one of the most important aspects of computer networks. It does however provide guarantees in terms of network performance parameters by utilizing resource reservation. New technologies are being designed to provide better QoS features, better scalability, reliability and security. The architecture of Differentiated services doesn't provide guarantees but is very scalable and its PHB (per-hop behavior) approach has proven to be very successful in delivering end-to-end QoS in large

*Recommended by thesis supervisor: Assoc. Professor Margaréta Kotočová. Defended at Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava on November 25, 2013.

Hrubý, M. Optimization of Network Traffic Flow. Information Sciences and Technologies Bulletin of the ACM Slovakia, Vol. 6, No. 1 (2014) 47-56

scale networks [16]. It is important to note that these two approaches to delivering quality of services are complimentary in nature. DiffServ enables better QoS scalability, while IntServ provides a guaranteed traffic delivery. Together, they form a robust QoS deployment. Both approaches have their drawbacks and advantages and current practice is showing that using the best of both IntServ and DiffServ might create a flexible framework for providing QoS in computer networks. Modern additions to area of QoS signaling are helping to bring IntServ and DiffServ closer together enabling the best of both concepts to be used together in one network, because QoS NSLP can signal for any QoS model. The general idea is always to improve the user's quality of experience (QoE) when using network services [12]. In the following chapter we analyze traffic engineering as a method of optimizing traffic flow through the network.

3. Traffic Engineering

Operating a large network with diverse services, like most service providers today do, requires continuous attention to the distribution of traffic over the network. Not only must the service level agreements with the customer be met but also traffic flowing between service providers must be in acceptable boundaries [7]. Network failures and changes in routing policies can trigger sudden shifts in the flow of traffic and thereby degrade the perceived QoS [6]. Traditional interior gateway routing protocols (e.g. OSPF, IS-IS) do not allow practical implementation of sophisticated traffic engineering policies in IP networks. With MPLS (multiprotocol label switching) some of these shortcomings can be addressed more directly. Most existing traffic engineering methods currently require manual configuration of link weights or tunnels which can be difficult to get right without prior knowledge of (future) traffic demands. For these purposes a multitude of approaches have been proposed, most of which center around populating a traffic matrix which is then used to make traffic engineering decisions [15]. "Traffic Engineering (TE) is concerned with performance optimization of operational networks. In general it encompasses the application of technology and scientific principles to the measurement, modeling, characterization, and control of Internet traffic, and the application of such knowledge and techniques to achieve specific performance objectives" [2]. To provide a systematic way to provision modern networks with traffic engineering capabilities while ensuring scalability and QoS, abstract models have been proposed. The authors in [3] have proposed a traffic engineering process model (see Figure 1) to illustrate the basic concepts of traffic engineering.

This general model is implemented by a multitude of traffic engineering approaches. These approaches differ in the method of network performance parameter discovery, method of configuration delivery and design suitability. A complete taxonomy of traffic engineering approaches can be found in Figure 2 and can be broken into several categories:

- From the aspect of traffic type, TE can be classified into unicast TE and multicast TE.
- From the aspect of traffic optimization scope, TE can be classified into intradomain TE and interdomain TE.

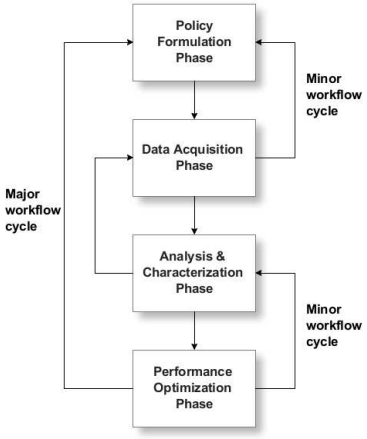

Figure 1: TE process model

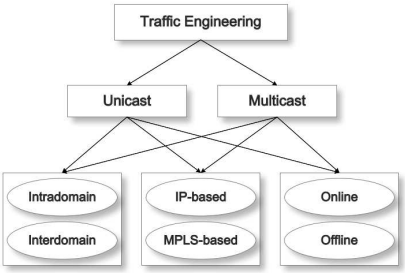

Figure 2: Classification of TE approaches

- From the aspect of routing enforcement mechanisms, TE can be classified into multiprotocol label switching (MPLS)-based TE and IP-based TE.
- From the aspect of availability of traffic demand or timescale of operations, TE can be classified into offline TE and online TE.

In our thesis we make a contribution to traffic engineering mechanisms in MPLS intradomain networks. A lot of traffic engineering approaches rely on the existence of the traffic matrix to achieve flow optimization. This is in our view a major drawback of current approaches because the traffic matrix is hard if not infeasible to obtain with fine granularity in large networks and traffic matrix prediction models increasingly inaccurate with time. A different method of obtaining information about traffic flows would maybe open new possibilities to traffic flow optimization, and this is what we aim to accomplish.

4. Optimization of traffic flow based on statistical modeling

Usually, traffic engineering approaches utilize the traffic matrix. However, acquiring accurate traffic matrices is infeasible in large networks. Secondly, a traffic matrix may store values of one observed parameter for all origin-destination pairs. Therefore, traffic engineering decisions can be made based observing one parameter (usually bandwidth requirement). In modern networks however, we must take multiple parameters in mind because some applications (especially VoIP and streaming video) have requirements on delay, jitter and also loss in addition to bandwidth requirements [10]. Thus the situation with traffic matrices complicates even more.

We propose a model which would not require a traffic matrix in order to achieve some functionality of traffic engineering. First we abstract from the network flow to a lower level of granularity - the network link. By performing a number of active measurements we can gather a statistical sample of multiple types of network performance parameters - for example delay and jitter. These parameters can be put into comparison by our model, which also solves the question of comparing paths based on multiple criteria. For example, which link should be preferred: the link with low delay and medium jitter or the link with medium delay and low jitter? Such questions can be answered by our model, the n-CUBE.

The model which we propose will provide multivariate normal distributions of measured parameters. The objective is to gather some measurement results for each link in the network and then compare statistical properties of multivariate normal probability density functions for these links, in our model. We assume that the number of (physical) links in a transit network is sharply lower than the number of flows traversing the network therefore our approach should provide a more computationally feasible solution to the traffic engineering problem than the traffic matrix. The model should be general and usable for any combination of network performance parameters, as defined in our analysis. Having obtained a measurement sample from the network, our model is used to perform statistical analysis of all network links. The output of this analysis is a set of weights assigned to the links. Finally, having assigned weights to network links based on our statistical analysis, we can find shortest paths across our transit network. For this purpose we propose an algorithm utilizing well-known Dijkstra's and Kruskal's approaches. The verification of our model will be performed in a simulation framework and in a laboratory environment on a real measurement apparatus with artificially generated network traffic. For verification purposes we will implement the n-CUBE model in the Matlab framework and later use this as part of a traffic engineering system which will be used to verify traffic engineering capabilities in a real network. In the Matlab framework we will verify correctness of operation and computational feasibility as well as performance. In the real network environment, as part of a traffic engineering system, we will verify the correctness of operation and evaluate traffic engineering capabilities. Our results will be compared to the basic scenario, in which the path is chosen by the routing protocol. To summarize, we aim to accomplish several tasks as part of our thesis. Firstly we will propose and describe our model, the n-CUBE. Secondly we will implement this model and experiment with it in a simulation environment. Next, we will propose and implement a working application of a traffic engineering system which will leverage our n-CUBE model and deliver traffic engineering capabilities into a computer network. Finally we will analyze performance results gathered from the simulation framework, and compare our results gathered from experiments in a real network environment with referential results.

4.1 Multivariate normal distribution

In previous sections when describing our contribution - the n-CUBE model, we introduced modelling of measured NPP (network performance parameters) using the multivariate normal distribution. At this point we prove that network performance parameters follow a normal distribution by conducting live measurements on five backbone links in a service provider backbone. These measurements were carried out by generating Cisco IP SLA measurement probes every minute for a period of 24 hours. Results were analyzed both graphically and analytically. Presently there are no direct multivariate normality tests known to us. Therefore we generally test the normality of individual variables and assume they are multivariate normal if they are individually normal. Presently there are both graphical and statistical methods for evaluating normality. Graphical methods include the histogram, normal probability plot and the "Q-Q" plot (quantile to quantile). Statistical methods include hypothesis tests and multiple different types of tests have been designed. It is important to say that none of these methods is absolutely definitive. We first explore three graphical methods starting with the histogram, where a set of 40 samples is plotted in Figure 3. The histogram represents unique samples in a set (x axis) together with the total number of each particular sample inside the set (y axis). Graphical observation hints to the fact that the samples might follow a normal distribution.

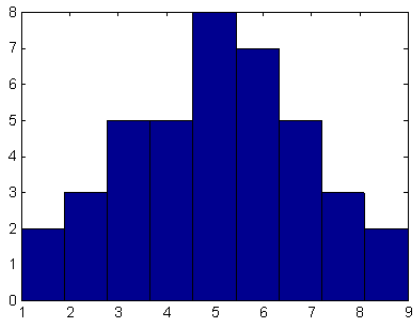

Figure 3: Histogram of a set of 40 samples

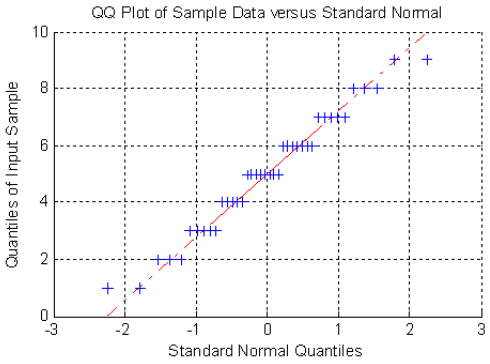

Figure 4: QQ plot of a set of 40 samples

The QQ plot (quantile to quantile) is another graphical method used for comparing two probability distributions by plotting their quantiles against each other. It can be used to assess whether the sampled data is approximately normally distributed. In a QQ plot in Figure 4, the sampled data is plotted against a theoretical normal distribution. Samples represent points on the plot and should form an approximate line. Deviations from this line indicate deviation from normality.

Taking into consideration these preliminary findings, we now construct a formal hypothesis test during which we

Table 1: Summary of statistical tests

Jarque-Bera test	Set 1	Set 2
Number of samples	40	300
p-value	0.575	0.693
significance level α	0.05	0.05
result	is normal	is normal
Lilliefors test	Set 1	Set 2
Number of samples	40	300
p-value	0.1522	0.5
significance level α	0.05	0.05
result	is normal	is normal
Shapiro-Wilk test	Set 1	Set 2
Number of samples	40	300
p-value	0.6708	0.4952
significance level α	0.05	0.05
result	is normal	is normal

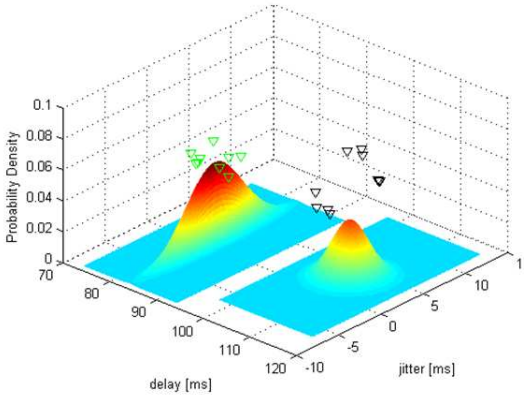

Figure 5: Example of two links plotted in the n-Cube model

prove that the sampled data (measured values of one-way delay) represents a statistic which follows the Normal (Gaussian) distribution. There are a number of tests we can use to verify normality of our sample, for instance the Lilliefors test [13], Shapiro-Wilk test [17] or the Jarque-Bera test [11]. We construct a null hypothesis that our sample follows a normal distribution and verify it using all of these tests. The results are summarized in Table 1.

Based on both the graphical results as well as verification of our statistical hypothesis listed above, we conclude that the measured samples of one-way delay on service providers' backbone links follow a normal distribution.

4.2 n-CUBE model

At this point we propose an n-CUBE model for traffic flow optimization based on statistical modeling. As part of this model, NPP are periodically acquired by means of active network measurements and fed into the model.

We establish the multivariate normal distribution which is a generalization of the one-dimensional (univariate) normal distribution to higher dimensions. A random vector $\boldsymbol{X} = [X_1, \cdots, X_n]^T$ is said to have a multivariate normal distribution with mean $\mu \in R^n$ and covariance matrix σ if its probability density function [5] is given by

$$f(\boldsymbol{X}; \mu; \sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\sigma|^{\frac{1}{2}}} exp(-\frac{1}{2}(\boldsymbol{X} - \mu)^T \sigma^{-1} (\boldsymbol{X} - \mu)) \quad (1)$$

A random vector $\boldsymbol{X}$ is said to be n-variate normally distributed if every linear combination of its n components has a univariate normal distribution. The probability density function of the bivariate normal distribution, the covariance matrix (of vectors delay and jitter for each link) and the vector of expected values form the basis of our proposed CUBE model. Having obtained a small sample of network performance parameters of interest (e.g. delay and jitter) we are able to model the probability density for each link and this property describes the space in which network performance parameters take on values for a particular link with a given probability. This enables us to make a small number of active network measurements at first, and then make statistical assumptions about future values of network performance parameters. It also enables us to put various dissimilar network performance parameters into comparison. In the following paragraph we will explain the formation of the n-CUBE model. To be able to make decisions about link properties, it is necessary to plot measurement results in a specific model. We have made assumptions of gathering two types of network performance parameters and putting these pairs into comparison. Thus for our purposes we use model of a cube. As it is 3D model in which we are able to use each of the 3 axes for a separate attribute.

- axis X - variable jitter (vector X)
- axis Y - variable delay (vector Y)
- axis Z - probability density

In general though, it is possible to assign any number and type of network performance parameters to our model. Thus the n-CUBE designation is because of n parameters. For n parameters however, the resulting hypercube is not convenient for direct observation, but the advantages of probability density modeling still apply. Each link will be represented with a multivariate normal distribution function inside the model. Axes X and Z will represent the base of this distribution while axis Y will represent the probability density of delay-jitter pairs (for our example of two parameters). All these multivariate normal distributions are merged into one cube and processed to determine the best links. An example of two links plotted in our n-CUBE model can be seen in Figure 5. Our objective is to identify specific areas of the plotted surface inside the cube model, which are characterized as peaks of a multivariate normal distribution that represent (most probable) discovered values of delay and jitter, as depicted in the Figure 5.

Representation of the network links as graph edges with directly measured parameters (weights) in the n-cube model has several advantages. First, the trustworthiness of a link can be determined by the σ parameter . The σ parameter corresponds, in the case of bivariate normal distribution, is represented by a matrix. The elements of this matrix can be viewed as parameters defining the zone of the normal distribution peak. This hints to the steepness of the distribution. Of course, the steepness of the distribution is best determined by examining

the kurtosis and determining whether the distribution is platykurtic, mesokurtic or leptokurtic. Those edges with steeper modeled distributions experience low variation in delay and jitter as opposed to those with gradually distributed probability function.

$$\Gamma_1 = \begin{bmatrix} x_1 & 0 & 0 \\ 0 & y_1 & 0 \\ 0 & 0 & z_1 \\ 0 & y_1 & z_1 \\ x_1 & 0 & z_1 \\ x_1 & y_1 & 0 \\ x_1 & y_1 & z_1 \end{bmatrix} \tag{2}$$

$$\Gamma_2 = \begin{bmatrix} x_2 & 0 & 0 \\ 0 & y_2 & 0 \\ 0 & 0 & z_2 \\ 0 & y_2 & z_2 \\ x_2 & 0 & z_2 \\ x_2 & y_2 & 0 \\ x_2 & y_2 & z_2 \end{bmatrix} \tag{3}$$

$$\Gamma_3 = \begin{bmatrix} x_3 & 0 & 0 \\ 0 & y_3 & 0 \\ 0 & 0 & z_3 \\ 0 & y_3 & z_3 \\ x_3 & 0 & z_3 \\ x_3 & y_3 & 0 \\ x_3 & y_3 & z_3 \end{bmatrix} \tag{4}$$

where $x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3 \in \mathbb{R}_{\geq 0}$

$$0 < x_1 < x_2 < x_3 \tag{5}$$

$$0 < y_1 < y_2 < y_3 \tag{6}$$

$$z_1 > z_2 > z_3 > 0 \tag{7}$$

$$x_3 = sup\{\textbf{jitters}\} \tag{8}$$

$$y_3 = sup\{\textbf{delays}\} \tag{9}$$

$$z_1 = sup\{f(\mathbf{X}, \mu, \sigma)\} \tag{10}$$

The values of coordinates are determined "on the run" but their relationships must be configured ahead. Of particular interest are values of parameters x_3, y_3 and z_1. These parameters contain maximum values of measured delays and jitters (in case of x_3, y_3) or the maximum value of any probability density function (in case of z_1). Other parameters are specified in relation to these values and

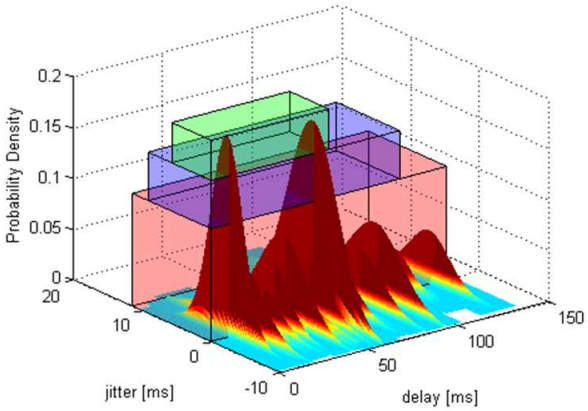

Figure 6: n-Cube model with regions

thus "carve out" the regions. For our purposes we chose the following values of parameters x_1, x_2, y_1, y_2, z_2 and z_3:

$$x_2 = c_1 x_3 \tag{11}$$

$$y_2 = c_1 y_3 \tag{12}$$

$$z_2 = c_1 z_1 \tag{13}$$

$$x_1 = c_2 x_2 \tag{14}$$

$$y_1 = c_2 y_2 \tag{15}$$

$$z_3 = c_2 z_2 \tag{16}$$

where $\{c_1, c_2 \in \mathbb{R} | 0 < c_1, c_2 < 1\}$.

This provides a distribution of regions, as the one depicted in Figure 6. Coefficients c_1 and c_2 are configurable, however in our thesis we chose values for these coefficients to be $c_1 = 0.8$ and $c_2 = 0.7$. This provides nicely spaced regions and has a good probability that each region would contain links. For instance, if the coefficients were chosen more aggressively, a region might "starve" in a sense that no links would be a part of it. In such a scenario there would be no benefit in segmenting the model into areas. The values of coefficients can be altered in our working implementation of the n-CUBE model and ideal values might depend on environment in which the model is deployed. The general idea is always to differentiate between good reliable links and others.

Our defined thresholds will divide the n-CUBE model into 3 regions (see Figure 6). Green region contains the best links (region Γ_1). For future processing, these links have

metric of 1. Purple region (Γ_2) contains link which aren't as good (comparatively). All these links have metric of 2. Last region (Γ_3), pink region represents the rest of the links which aren't suitable for transmission of traffic which is affected by high values of observed performance parameters and should be used only as a temporary solution. These links have metric of 5 in subsequent calculations. These metrics have been chosen in this way due to the nature of our best-path algorithm (further text). We want to make the best links (those part of region Γ_1) to be most preferred when choosing the path therefore a minimal weight of 1 is assigned to them. Subsequently the links assigned to region Γ_2 will have a weight of 2 (slightly less preferable) and links assigned to region Γ_3 a weight of 5 (least preferable). These values of metrics are closely related to the values of coefficients c_1 and c_2 and are also adjustable in our working implementation of the n-CUBE model. The general idea is to provide enough differentiation of the links based on region membership. Having assigned each link a weight we can now proceed to finding the shortest paths for our traffic.

4.3 Least-cost paths

The process of calculating shortest paths is based on graph theory and we will utilize some aspects of graph theory in our implementation of the n-CUBE model. First, let's introduce the graph. A graph is an ordered pair G = (V, E) comprising a set V of vertices or nodes together with a set E of edges or lines, which are 2-element subsets of V (i.e., an edge is related with two vertices, and the relation is represented as unordered pair of the vertices with respect to the particular edge).We will represent the graph as defined in [9] as a data structure containing both edges and vertices. Edges are stored in variable EDGES and vertices are stored in variable VERTICES, thus representing 2-element subsets (VERTICES, EDGES) of the graph [14], [8]. Each edge stored in variable EDGES has the following properties:

- Metric - represents the weight of the edge (1,2,5)
- Visited - Boolean value (TRUE, FALSE) to identify links which were already processed
- Tree - Boolean value (TRUE, FALSE) to identify links which were chosen into the final tree

Each vertex stored in variable VERTICES has the following properties:

- Border - holds a Boolean value (TRUE, FALSE) to identify whether this vertex represents a border router (on the edge of the transit network)
- Edges - contains a list of adjacent edges

Our algorithm is further described by the following pseudocode.

```
void function fetch_subgraphs (VERTEX, SUBGR_ID){
  if  VERTEX  has  no  EDGE  in  VERTEX.EDGES
      with  Metric  ==  1  return;
  if  VERTEX  has  no  EDGE  in  VERTEX.EDGES
      with  EDGE.VISITED  ==  false  return;
  SUBGR[SUBGR_ID]  +=  VERTEX;
  choose any  EDGE  from  VERTEX.EDGES
     where Metric == 1 and EDGE.VISITED == false;
  EDGE.VISITED  =  true;
  EDGE.Tree  =  true;
  VERTEX_2  =  OTHERSIDE(EDGE, VERTEX);
  fetch_subgraph  (VERTEX_2);
}

void  function  fetch_tree  (EDGES,  VERTICES){
  SUBGR_ID = 0;
  foreach  VERTEX  in  VERTICES  {
  if  VERTEX.BORDER  ==  true  {
     SUBGR[SUBGR_ID]  +=  VERTEX;
     }  else  fetch_subgraph(VERTEX, SUBGR_ID);
     SUBGR_ID++;
  }

MAIN_GRAPH_SPF  =  Dijkstra(VERTICES);

foreach  VERTEX_A  from  SUBGR[x]  {
  foreach  VERTEX_B  from  SUBGR[y]  {
    GR += find min MAIN_GRAPH_SPF(VERTEX_A,
        VERTEX_B);
    y++;
    }
  x++;
  }
RESULT_GR  =  Kruskal(GR);
}
```

5. Implementation

In this section we briefly describe the implementation of our proposed n-CUBE model. For implementation purposes we have decided to break up functional elements into blocks. The elements (see Figure 7) will function as separate entities with possible multiple instances per element. Blocks are defined as follows:

- Application in the NMC - this application leverages a common existing infrastructure of the network management center which has access to all managed routers from a set of centralized jump servers. The application will feature three autonomous processes working asynchronously and achieving:
  - Polling - fetching runtime variables from routers via SNMPv3 (mostly IP SLA statistics)
  - Update - pushing updated configurations to routers via SSH (modification of link weights as a result of n-CUBE model weight assignment)
  - Data interface - storing the fetched runtime variables into a common repository, which is accessible to all (e.g.: file, SQL database, etc.)
- Common repository - this should be a universally, but securely accessible resource which enables fast data storage and retrieval. Consistency of data and shared access is guaranteed by underlying methods (operating system or data base)
- n-CUBE modeling - this application will retrieve runtime data stored in the common repository and model link weights based on multivariate normal distribution of desired network performance parameters. This will be achieved by a Matlab application

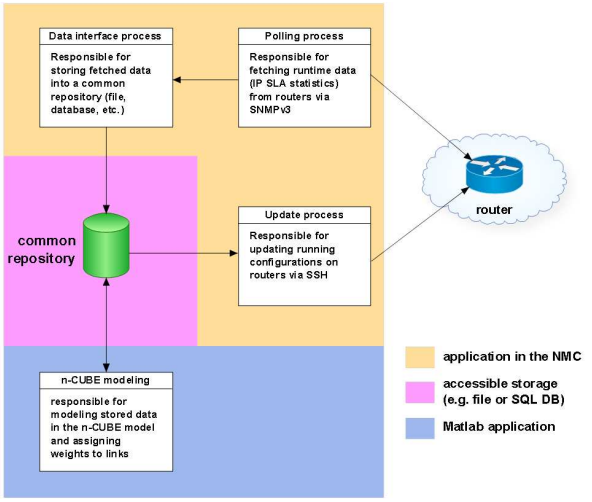

Figure 7: Implementation separated into functional blocks

For verification purposes, the entire solution will be deployed in a network laboratory environment. Our design supports clustering and segmentation and thus enables scalability for long-term use. A more detailed design diagram can be seen in Figure 8 where functional blocks are interconnected with managed routing areas (under common control).

Polling and Update processes are periodically managing their designated area (all routers within an area) at specified intervals. The interval value is configurable and it is advised to synchronize this interval to the IP SLA object frequency. The Polling process periodically sends SNMP GET messages to managed routers inside a common area to retrieve latest measurement details (IP SLA statistics) and stores each new value inside the data interface. The Update process maintains a last-known copy of the data interface (file or SQL database table) and only applies configuration changes to managed network area in case of a change and only to routers which are affected by the change. The configuration change is done by an automated agent who logs into the managed router via a secure SSH channel and applies a change configuration. Compute servers are running a Matlab application instance accessing a particular section of the data interface (e.g. file or SQL database table). Modeling the n-CUBE is done in real-time and the data is periodically updated in configurable time intervals (it is advisable that these time intervals be configured to match with IP SLA object frequency). Link weight updates are being fed back to the common repository at these intervals whenever a change is determined by the n-CUBE model.

Once the optimization is triggered, based on preferences set, the prepared templates are configured with data and sent to the routers which are affected by the optimization. The templates contain commands for Cisco IOS routers which accomplish two things:

- configure an explicit path, this is a construct containing hop by hop information about the path through which the traffic-engineering tunnel will be set up. The hop by hop IP addresses are extracted from the discovered IP SLA probes and the path is determined by our algorithm as described in previous chapters
- configure the Tunnel interface, this is a special type of virtual interface used to establish the traffic engineering tunnel and configure session parameters.

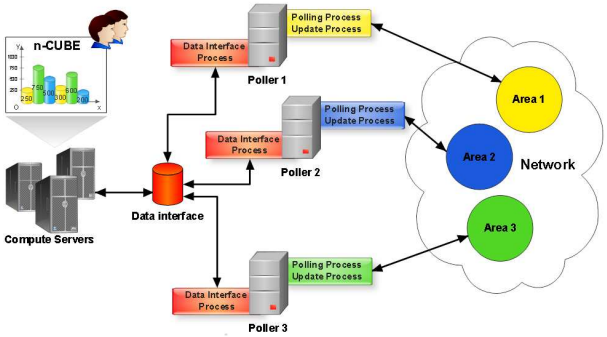

Figure 8: Data flow in the logical design

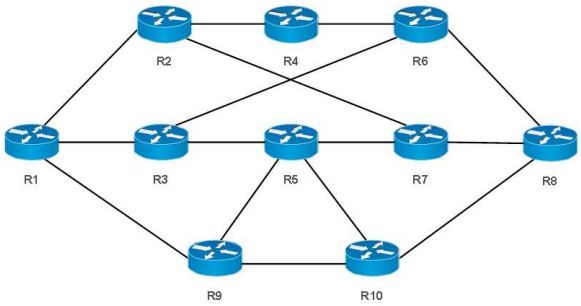

Figure 9: Physical topology for use in verification

6. Verification and comparison

In this section we provide results and comparison of experiments with our proposed n-CUBE model. For implementation purposes, an experimental topology was used specified in our thesis and traffic was artificially generated into the network by means of the Iperf and D-ITG traffic generators. IP SLA probes were setup to measure, in addition to per-link measurements between routers, end-to-end network performance parameters as specified in our thesis.

6.1 Experiment 1

In the first experiment we will generate traffic artificially as described by the proposal and we use the topology depicted in Figure 9. During the experiment, IP SLA measurement probes will be sent (in addition to being sent on every link as part of data gathering) end-to-end to quantify end-to-end performance parameters between communicating parties. All generated traffic except the IP SLA probes will be part of the same class and bears the same marking (in the MPLS domain it will be EXP=1). The IP SLA probes will however be sent with a different marking (in the MPLS domain it will be EXP=2). This is to observe the effect of optimization on IP SLA measurement probes which will continue to flow through the same path as artificially generated traffic prior to optimization. Traffic optimization will therefore be done for all IP SLA traffic (EXP=2).

Table 2: Timeline of experiment 1

time	action
0:00	three stations start generating traffic at maximum possible rate
10:00	Traffic optimization is triggered

In Figure 10 we observe the measured round-trip time and jitter, gathered throughout our experiment 1. From the start of measurement the artificially generated traffic was flowing over the same path as IP SLA measurement probes. Approximately 10 minutes into the experiment, the optimization was triggered. Once the optimization was triggered, the n-CUBE model (which was modeling links from the start of the measurement) is used to calculate link metrics based on region membership and calculate the least cost path. This information is fed into the shared data storage where it is retrieved by the Update process. Once the path is known to the Update process, it uses this information to populate an appropriate template with predefined configuration statements. The template represent a configuration which is immediately pushed to routers R1 and R8 (border routers) via an automated SSH agent. The newly created traffic engineering tunnel for the artificially generated traffic is created along the explicitly defined path. Once up and running, traffic immediately starts flowing through a new path. The general timeline of the experiment 1 can be seen in Table 2.

6.2 Experiment 2

In the second experiment we will generate traffic artificially as described by our proposal. During the experiment, in addition to sending IP SLA probes on each link as part of data gathering, we will send two distinct IP SLA measurement probes. One probe will be marked as EXP=4 in the MPLS domain and the other as EXP=5. Traffic generated from PC-1 to PC-4 will be marked as EXP=5 in the MPLS domain and all other generated traffic (between PC-2 and PC-5, and PC-3 and PC-6) will be marked as EXP=4. Traffic optimization will be done for traffic marked with EXP=5 and a class-based tunnel will be created to transmit all traffic marked as such. All other traffic will be sent over the original path (before or after optimization).

Depicted in Figure 11 is the round-trip time for artificially generated traffic (with different traffic class markings) throughout the experiment. Similarly in Figure 11 we can observe the traffic load imposed on our network in terms of throughput. The general timeline of our experiment 2 can be seen in Table 3. Once the optimization was triggered, traffic marked as EXP=5 in the MPLS domain was forwarded along a new, traffic-engineered path. Thus we have redirected the flow of traffic away from the congested path. As can be seen in Figure 11, once EXP=5 traffic was redirected to a new path toward its destination, the throughput improved also for the rest of traffic (marked EXP=4).

In experiment 2 we showed the measured network performance parameters (round-trip time and jitter) for our artificially generated traffic prior to and immediately after optimization. The optimization achieved two things:

Table 3: Timeline of experiment 2

time	action
2:00	two stations start generating traffic marked as EXP=4 at maximum possible rate
6:00	one station starts generating traffic marked as EXP=5 at maximum possible rate
10:20	Traffic optimization is triggered
17:00	all stations stop generating traffic

- a new path was found for traffic marked as MPLS EXP=5 by our n-CUBE model (as a direct result of our modeling)
- based on this modeling the configuration template was populated with data and the configuration was delivered by our Update process (via an automated SSH client) to the affected border routers R1 and R8 thus creating the explicitly defined traffic engineering tunnel over which the traffic should be flowing

7. Conclusions

In this paper we provided a brief overview of two major areas of quality assurance and traffic flow optimization: QoS and traffic engineering. In modern networks with increasing amounts of multimedia traffic (VoIP and streaming video) and businesses relying on service level agreements with service providers, the use of QoS mechanisms has become mandatory. Traffic engineering comes into play in large (global span) networks were large amounts of traffic are concentrated into trunks with high throughput. It is necessary to utilize available network resources intelligently and route network traffic with regard to traffic type and QoS requirement, which common IGP scenarios are unable to accomplish.

We proposed a technique to optimize the flow of traffic in large service provider networks. We do this by postulating a method of performing a small number of active network measurements inside the network and gathering sample network performance parameters for network links. Then we analyze this data in our proposed CUBE model and get a multitude of useful information. Not only can we decide which links are best suited for routing VoIP traffic (comparing a multi-parametric system, because we can compare multiple unrelated parameters at once) we can also identify links with higher probability of unpredictability (i.e. we cannot estimate future network performance parameter values with enough confidence) and links suitable for network observation and upgrade. We utilize our model to assign weights to links in the network and then use a modification of Dijkstra's and Kruskal's algorithm to generate a traffic-engineered path suitable for routing traffic which is sensitive to the observed parameters. The traffic-engineered path (or tunnel) is an overlay network our model provides a visualization of this path in the real network.

We conducted experiments with our proposed model which is implemented in the MATLAB R2010a framework in a real network laboratory environment. We have done a preliminary experiment to show how the values of round-trip time and jitter are changing with increased network

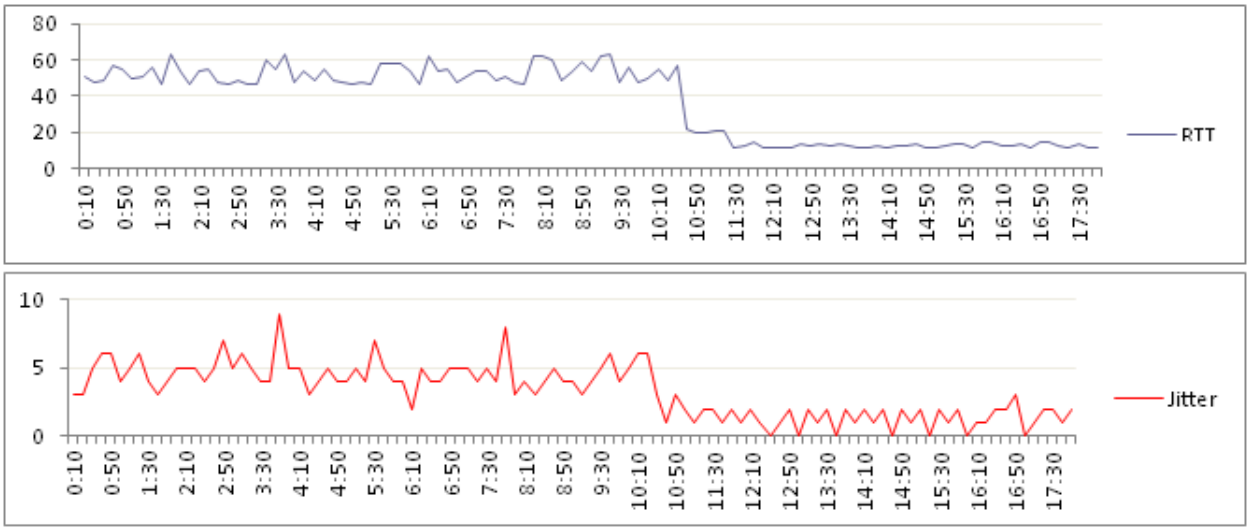

Figure 10: Experiment 1 - measured round-trip time in ms (top figure) and jitter in ms (bottom figure)

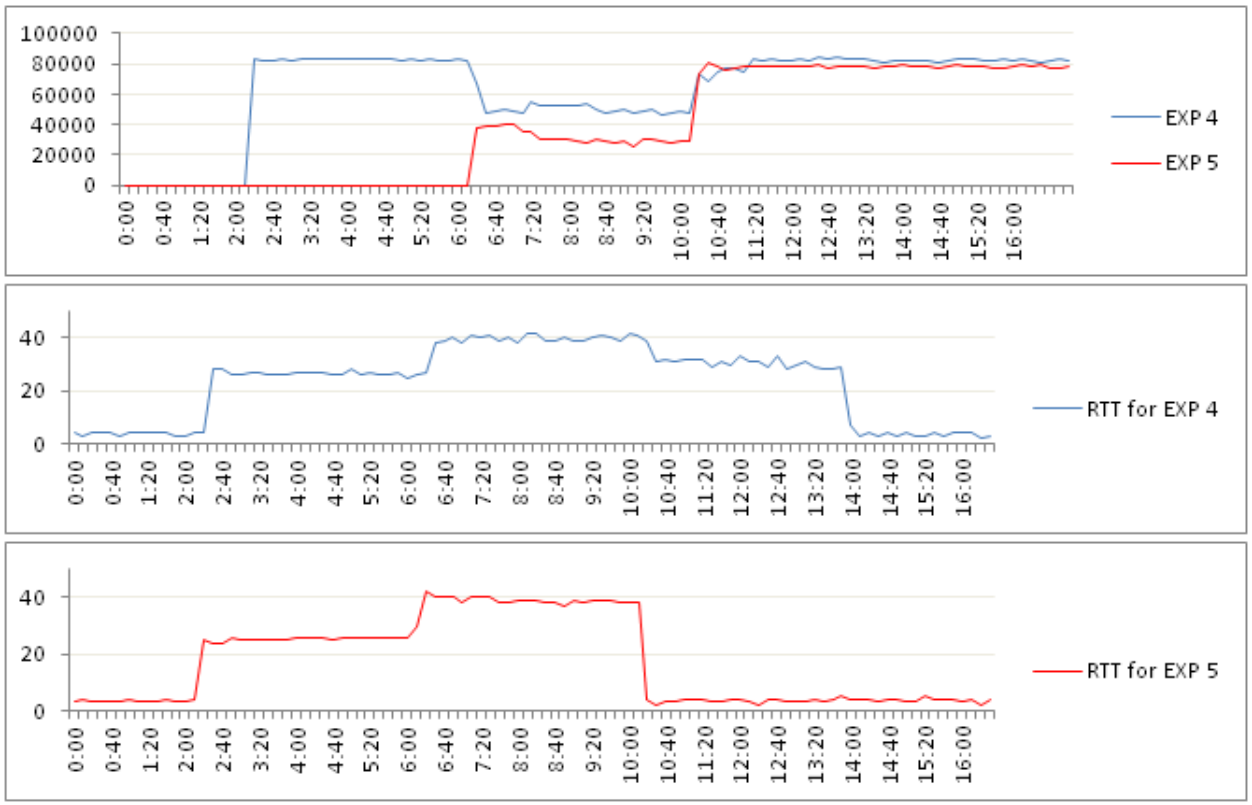

Figure 11: Experiment 2 - measured throughput in kb/s (top figure) round-trip time for EXP=4 traffic in ms (middle figure) and round-trip time for EXP=5 traffic in ms (bottom figure)

load. In subsequent experiments we have generated artificial network traffic into a measurement topology and then triggered network optimization by allowing our model to feed new configurations into the network. The results were listed in graphical output form. Some of our experimental work was also published on well established scientific conferences.

Acknowledgements. The author would like to thank his supervisor Assoc. Professor Margaréta Kotočová for her valuable guidance during our research. This work was partially supported by the Slovak VEGA Grant Agency Grant No. 1/0676/12.

References

[1] T. Allen. *Internetworking Technologies Handbook*. Cisco Press, San Francisco, USA, 2003.

[2] D. O. Awduche. *Requirements for Traffic Engineering Over MPLS*. RFC 2702, September 1999.

[3] D. O. Awduche and B. Jabbari. Internet traffic engineering using multi-protocol label switching (mpls). *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 40(1):1389–1286, September 2002.

[4] V. G. Cerf and R. E. Kanh. A protocol for packet network intercommunication. *IEEE Transactions on Communications*, COM-22:627–641, May 1974.

[5] C. B. Do. "the multivariate gaussian distribution [online at: http://cs229.stanford.edu/section/gaussians.pdf]", October 2008.

[6] C. Feng and S. Malik. Vulnerability analysis and best practices for adopting ip telephony in critical infrastructure sectors. *IEEE Communications Magazine*, 44(4):138–145, April 2006.

[7] J. L. Garcia-Dorado, A. Finamore, M. Mellia, M. Meo, and M. Munafo. Characterization of isp traffic: Trends, user habits, and access technology impact. *IEEE Transactions on Network and Service Management*, 9(2):142–155, June 2012.

[8] P. B. Guttoski, M. S. Sunye, and F. Silva. Kruskal's algorithm for query tree optimization. *Proceedings of the 11th International Database Engineering and Applications Symposium*, pages 296–302, September 2007.

[9] M. Hruby, M. Olsovsky, and M. Kotocova. Routing voip traffic in large networks. *In Proceedings of the World Congress on Engineering (WCE 2012)*, pages 798–803, July 2012.

[10] S. Hyeongu and L. Youngseok. Detecting anomaly traffic using flow data in the real voip network. *Proceedings of the 10th IEEE/IPSJ International Symposium on Applications and the Internet (SAINT'10)*, pages 253–256, July 2010.

[11] C. M. Jarque and A. K. Bera. Efficient tests for normality, homoscedasticity and serial independence of regression residuals. *Economics Letters*, 6(3):255–259, 1980.

[12] H. J. Kim and S. G. Choi. A study on a qos/qoe correlation model for qoe evaluation on iptv service. *Proceedings of the 2010 12th International Conference on Advanced Communication Technology (ICACT '10)*, pages 1377–1382, February 2010.

[13] H. W. Lilliefors. On the kolmogorov-smirnov test for the exponential distribution with mean unknown. *Journal of the American Statistical Association*, 64:387–389, 1969.

[14] B. Liu and et al. Finding the shortest route using cases, knowledge, and djikstra's algorithm. *IEEE Expert*, pages 7–11, October 1994.

[15] P. P. Casas, S. Vaton, L. Fillatre, and L. Chonavel. Efficient methods for traffic matrix modeling and on-line estimation in large-scale ip networks. *Proceedings of the 21st International Teletraffic Congress (ITC'09)*, pages 1–8, September 2009.

[16] T. Samak, A. El-Atawy, and E. Al-Shaer. Qos policy verification for diffserv networks. *Proceedings of the 2011 19th International Workshop on Quality of Service (IWQoS '11)*, pages 3–8, June 2011.

[17] S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965.

Selected Papers by the Author

M. Hrubý, M. Olšovský, M. Kotočová. A Novel Technique for TCP Based Congestion Control. In *Proceedings of the 4th International Conference on Data Communication Networking*, Reykjavík, Iceland 29-31 July, 2013. SciTePress, 2013.

M. Hrubý, M. Olšovský, M. Kotočová. Algorithm for Dynamic Traffic Rerouting and Congestion Prevention in IP Networks. In *Lecture Notes in Electrical Engineering. - ISSN 1876-1100. - Vol. 152 Innovations and Advances in Computer, Information, Systems Sciences, and Engineering*, pages 649–660. New York. Springer, 2013.

M. Hrubý, M. Olšovský, M. Kotočová. Routing VoIP traffic in Large Networks. In *Proceedings of WCE 2012 World Congress on Engineering, Vol. 2*, pages 798–803. 4-6 July, London, U.K. IAENG, 2012.

I. Grellneth, M. Hrubý, P. Vilhan. Emulating Cisco Network Laboratory Topologies in the Cloud. In *Proceedings of ICETA 2011 : 9th IEEE International Conference on Emerging eLearning Technologies and Applications*, pp. 67–92. Stará Lesná, Slovakia. Piscataway : IEEE, 2011.

M. Hrubý, M. Olšovský, M. Kotočová. An Iterative Statistical Method for Congestion Prevention in Transit Networks. In *Proceedings of SAMI 2012 : IEEE 10th Jubilee International Symposium on Aplied Machine Intelligence and Informatics*, 26-28 January, 2012. Herľany, Slovakia. Budapest : Óbuda University, 2012.

M. Hrubý. Evaluating Delay and Jitter on D-ITG Generated VoIP traffic. In *Proceedings of Student Research Conference 2012. Vol. 2 : 8th Student Research Conference in Informatics and Information Technologies*, pp. 308–314. 4 May, 2011, Bratislava, Slovakia. Bratislava : Nakladatel'stvo STU, 2011.