©2004 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The definitive version of this paper has been published in 11th International Telecommunication Network Strategy and Planning Symposium (Networks2004), 2004, 10.1109\/netwks.2004.240885.

Capacity Assignment for NAC Budgets in Resilient Networks

Michael Menth, Jens Milbrandt, Stefan Kopf
Department of Distributed Systems, Institute of Computer Science, University of Würzburg
Am Hubland, D-97074 Würzburg, Germany
phone: (+49) 931-8886644, fax: (+49) 931-8886632, email:{menth|milbrandt|kopf}@informatik.uni-wuerzburg.de

Abstract

This paper addresses Quality of Service (QoS) in the presence of network failures. QoS is ensured by Network Admission Control (NAC) mechanisms along virtual tunnels through the network. Their capacity budgets must be set such that the expected traffic can be transported and that unintended overbooking of the physical network capacity is avoided. This paper presents an adaptation for resilient networks, i.e., traffic rerouting is respected in case of local network outages.

1 Introduction

Quality of Service (QoS) in terms of packet loss and delay probability is a prerequisite for the convergence of telephone and data networks which is required for economical reasons. It can be achieved on a technical level by capacity overdimensioning or by Admission Control (AC), which is the perspective of this paper. The network availability is indispensable for business customers since most business processes rely on communication. Therefore, telephone providers offer a 99.999% network availability at full QoS which can not be met by today's Internet. Hence, packet-switched networks need traffic rerouting to be resilient against local outages and smart capacity planing which anticipates potential failure scenarios.

Network AC (NAC) can be performed in many ways. However, in [**?**] we have shown that fault tolerant QoS can be achieved most resource efficiently if border-to-border (b2b) budgets (BBBs) are used for NAC, which corresponds to AC on virtual tunnels. For the deployment of that method, the capacity assignment to the BBBs is required such that the expected traffic can be transported and such that the physical network capacity is not overbooked unintentionally [2]. In this paper we extend this method for resilient networks, i.e., the traffic rerouting in case of failure scenarios is respected for the capacity assignment. As this involves significantly more computation complexity, we also present an improved algorithms for speedup purposes.

In Section 2 we present the technical basics for BBB NAC and the required capacity assignment. Section 3 proposes a trivial and a smart approach for capacity assignment under resilience requirements and Section 4 compares these methods. Finally, we summarize our paper by some concluding remarks.

This work was funded by the Bundesministerium für Bildung und Forschung of the Federal Republic of Germany (Förderkennzeichen 01AK045) and Siemens AG, Munich. The authors alone are responsible for the content of the paper.

2 Budget Assignment for QoS Tunnels

We present the idea of BBB NAC and explain our traffic model for capacity dimensioning. Finally, we show how capacity is assigned to BBBs in a fair and efficient manner.

2.1 Border-to-Border Budget Based Network Admission Control

We use the following notation in this paper. A network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ is given by a set of routers $\mathcal{V}$ and set of links $\mathcal{E}$. The BBB NAC sets up a b2b budget $BBB_{v,w}$ between any two ingress and egress routers $v, w \in \mathcal{V}$ that may be administered at its ingress router. It records the demand $c(f)$ of the admitted flows $f \in \mathcal{F}$ in place. When a new flow $f_{v,w}^{new}$ arrives, it checks whether its effective bandwidth $c(f_{v,w}^{new})$ together with the demand of already reserved flows fits within the capacity budget $BBB_{v,w}$.

$$c(f_{v,w}^{new}) + \sum_{f \in \mathcal{F}} c(f) \leq c(BBB_{v,w}). \qquad (1)$$

If so, the flow is accepted, otherwise it is rejected. The BBB NAC avoids states inside the network which has many advantages. In failure scenarios the traffic may be just rerouted and no reservation states inside the network must be restored like with RSVP-like approaches.

2.2 Capacity Dimensioning and Computation of Blocking Probabilities

Capacity dimensioning is a function calculating the required bandwidth for given traffic characteristics and a desired blocking probability. Based on that, budget and link capacities can be computed [3].

Conversely, blocking probabilities can be computed by a function $p_b(a, c)$ depending on the offered load and the provided capacity. The specific implementation of that function depends on the underlying traffic model. The following traffic model is only used for illustration purposes. Our assignment methods, which are presented afterwards, are general and require solely the computation of a blocking probability $p_b(a, c)$.

2.2.1 A Multi-rate Traffic Model

We assume Poisson arrivals of resource requests and a generally distributed holding time. Although typical Internet traffic has different characteristics on the packet level [4], the Poisson model, which is used in the telephony world, is more realistic for the resource request level of end-user driven real-time applications. The offered load a is the mean number of active flows, provided that no flow blocking occurs. In a multi-service world like the Internet, the request profile is multi-rate, so we take n_r different request types r_i, $0 \leq i < n_r$ with a bitrate $c(r_i)$. Given an offered load a, the respective request type specific offered load is $a(r_i) = p_a(r_i) \cdot a$, where p_a is the portion of a generated by r_i. In our studies, we assume a simplified multimedia real-time communication scenario with $n_r = 3$, $c(r_0) = 64$ Kbit/s, $c(r_1) = 256$ Kbit/s, and $c(r_2) = 2048$ Kbit/s, and a mean bitrate of $E[C] = \sum_{0 \leq i < n_r} c(r_i) \cdot p_a(r_i) = 256$ Kbit/s. The recursive solution by Kaufman and Roberts [5] allows for the computation of request type specific blocking probabilities $p_b(r_i)$ if a certain capacity c is provided. We use Equation (2) to relate the blocking probability p_b to the traffic volume instead to the number of flows:

$$p_b = 1 - \frac{\sum_{0 \leq i < n_r} (1 - p_b(r_i)) \cdot c(r_i) \cdot p_a(r_i)}{E[C]}. \quad (2)$$

The Kaufman and Roberts approach works on multiples c_u of a basic capacity unit u_c. Therefore, we denote in the following the capacity by $c_u(.)$.

2.2.2 Adaptation to Border-to-Border Budgets

We abstract from a special budget $BBB_{v,w}$ to a general budget b. As the BBB NAC works on isolated virtual tunnels (which may also have multi-path structure), the blocking probability for the corresponding b2b traffic aggregate can be computed by $p_b(a(b), c_u[b])$, where $a(b)$ is the offered load and $c_u[b]$ [1] the capacity of a budget b.

[1]We denote a functional dependency using parentheses, e.g. $a(b)$, while we use brackets for depending values which can be set and changed by our algorithms, e.g. $c_u[b]$.

2.3 Fair and Efficient Capacity Assignment

Without AC, the capacity of a network is automatically shared by many flows but in case of BBB NAC the capacity must be assigned to the BBBs beforehand to be usable by flows. Here, the BBBs may compete for the same link capacities $c_u(l)$. Budgets with large offered load can use their capacity more efficiently to achieve the same blocking probabilities for their offered traffic than those with only little offered load. Therefore, if unintentional overbooking must be avoided, the assignment of the physical network capacity to the virtual budget capacity is an interesting problem.

The traffic routing is essential for that problem. The function $u(l, b)$ indicates the percentage of the traffic pertaining to budget b using link l. It is able to reflect both single- and multi-path routing.

The algorithm for fair and efficient capacity assignment to all budgets is given in Algorithm 1 [2]. We denote the set of all budgets by $\mathcal{B}$. At the beginning, all budgets are unassigned ($c_u[b] = 0$) and $\mathcal{B}_{hot} = \mathcal{B}$. The free capacity of a link l is $c_u^{free}(l) = c_u(l) - \sum_{b \in \mathcal{B}} u(l, b) \cdot c_u[b]$. To increase the budgets successively, a budget b^* with the currently largest blocking probability $p_b(a(b), c[b])$ is chosen and in case of ambiguity, a budget among them with a maximum offered load is taken. If there is enough capacity on all links supporting budget b^* ($\forall l \in \mathcal{E} : c_u^{free}(l) \geq c_u^{inc} \cdot u(l, b)$), the budget capacity is enlarged by c_u^{inc}. Otherwise, the budget is removed from $\mathcal{B}_{hot}$. We used intelligent data structures to speed up the algorithm but we do not discuss them here for clarity reasons.

```
Input:   (implicitly: topology, routing,
         budgets)
  $\mathcal{B}_{hot} := \mathcal{B}$
  while $\mathcal{B}_{hot} \neq \emptyset$ do
    choose $b^* \in \mathcal{B}_{hot}$ with largest blocking probability and take a budget with maximum offered load for tie breaking
    $c_u^{inc} := 1$
    if $\left(\forall l \in \mathcal{E} : c_u^{free}(l) \geq c_u^{inc} \cdot u(l, b^*)\right)$ then
      $c_u[b^*] := c_u[b^*] + c_u^{inc}$
    else
      $\mathcal{B}_{hot} := \mathcal{B}_{hot} \setminus b^*$
    end if
  end while
Output:  assigned budget capacities
         $c_u[b], b \in \mathcal{B}$
```

Algorithm 1: Capacity assignment (CAPASS).

3 Adaptation for Resilient Networks

In this section, we present an acceleration of the above algorithm which is required for networks with large offered load. Then, we extend this algorithm for resilience requirements.

3.1 Acceleration of the Algorithm

To speed up the above algorithm we maximize the capacity increment c_u^{inc}. A simple approach is setting it such that the free bandwidth is shared proportionally among the hot budgets of a link ($c_u^{inc} = \max(1, \lfloor \frac{q(l)a(b)}{h} \rfloor)$ with $a_{hot}(l) = \sum_{b \in \mathcal{B}_{hot}(l)} a(b) \cdot u(l,b)$ and $q(l) = \frac{c_u^{free}(l)}{a_{hot}(l)}$). This is problematic because budgets with little offered load need relatively more capacity than budgets with large offered load to achieve the same blocking probability. Hence, if budgets with large offered load take a large portion of the bandwidth, they can already achieve a very low blocking probability while other budgets with little offered load can only reach quite large blocking probabilities if they share the remaining capacity.

3.1.1 Safe Acceleration

We first present a safe acceleration of Algorithm 1 which is based on the above idea and that avoids the starvation of budgets with little offered load. To that aim, Algorithm 2 computes safe link-dependent capacity increments. A link-dependent capacity increment is safe, if it is so small that it decreases the candidate budget only to such an extent that any budget $b \in \mathcal{B}_{hot}$ increased by its fair share can undergo the resulting blocking probability. The variable q^{dec} controls the granularity and the speed of the algorithm. Algorithm 3 is a modification of Algorithm 1 and uses only safe capacity increments.

```
Input:   Link $l$ (implicitly: topology,
         routing, budgets)

  if $|\{b : b \in \mathcal{B}_{hot} \wedge u(l,b) > 0\}| > 0$ then
    choose $b^* \in \mathcal{B}_{hot} : u(l,b^*) > 0$ with
    largest blocking probability, use smallest of-
    fered load for tie breaking
    $c_u^* := \lfloor q(l) \cdot a(b^*) \rfloor$
    $p_b^* := p_b(a(b^*), c_u[b^*] + c_u^*)$
    for all $b \in \{\hat{b} : \hat{b} \in \mathcal{B}_{hot} \wedge u(l,\hat{b}) > 0\}$ do
      $c_u^b := \lfloor q(l) \cdot a(b) \rfloor$
      $p_b^b := p_b(a(b), c_u[b] + c_u^b)$
      while $p_b^* < p_b^b$ do
        $c_u^* := \lfloor q^{dec} \cdot c_u^* \rfloor$
        $p_b^* := p_b(a(b^*), c_u[b^*] + c_u^*)$
      end while
    end for
  else
    $c_u^* := 0$
  end if

Output:  suitable capacity increment $c_u^*$
```

Algorithm 2: Calculation of a suitable link capacity increment (CAPINC).

3.1.2 Simple and Fast Acceleration

The above described mechanism is still computation-intensive, especially because p_b is quite time consuming [3]. Therefore, we use a faster, tunable approach for which the correctness can not be proven in the sense that some budgets can be penalized. If its result shows that some budgets with little offered load have comparatively large blocking probabilities, the algorithms must be tuned more conservatively and run again.
We simply take

$$c_u^{inc} = \max\Big(1, \min_{l \in \mathcal{E}: u(l,b^*)>0} \Big(\lfloor \frac{q(l) \cdot a(b^*)}{h} \rfloor\Big)\Big) \qquad (3)$$

and set initially $h = 2$ in Algorithm 1. Additionally, we take budgets with the least offered load for breaking ties.

3.2 Extension for Resilience Requirements

If a local outage occurs in a network, the traffic must be rerouted. Therefore, sufficient capacity is required on the rerouted path or - in other words - the NAC must limit the admitted traffic such that the capacity suffices. The set $\mathcal{S}$ comprises all considered failure scenarios s which contain the remaining active network topology. To avoid special cases, we include the working scenario in $\mathcal{S}$. For each failure scenario, the routing changes and we describe it by the enhanced routing function $u(s,l,b)$. In the following, we present a simple and an enhanced method to extend the capacity assignment algorithms to resilience requirements.

3.2.1 Simple Resilience Extension

A simple extension of the above algorithm is the capacity assignment $c_u[s,b]$ for all failure scenarios $s \in \mathcal{S}$ with subsequent capacity minimization $c_u[b] = \min_{s \in \mathcal{S}} c_u[s,b]$. This yields obviously safe values for all considered failure scenarios.

```
Input:  (implicitly: topology, routing,
        budgets)
  for all $l \in \mathcal{E}$ do
    $c_u^{inc}[l] := CapInc(l)$
  end for
  $\mathcal{B}_{hot} := \mathcal{B}$
  while $\mathcal{B}_{hot} \neq \emptyset$ do
    choose $b^* \in \mathcal{B}_{hot}$ with largest blocking probability, use smallest offered load for tie breaking
    $c_u^{inc} := \max(1, \min_{l \in \mathcal{E}: u(l,b^*)>0} c_u^{inc}[l])$
    if $\left(\forall l \in \mathcal{E} : c_u^{free}(l) \geq c_u^{inc} \cdot u(l,b^*)\right)$ then
      $c_u[b^*] := c_u[b^*] + c_u^{inc}$
    else
      $\mathcal{B}_{hot} := \mathcal{B}_{hot} \setminus b^*$
    end if
    for all $l \in \mathcal{E}$ do
      if $u(l,b^*) > 0$ then
        $c_u^{inc}[l] := CapInc(l)$
      end if
    end for
  end while
Output:  assigned budget capacities
         $c_u[b], b \in \mathcal{B}$
```

Algorithm 3: Accelerated capacity assignment (ACCCAPASS).

3.2.2 Enhanced Resilience Extension

The following method performs faster and yields more efficient results than the preceding approach. We define failure scenario depending functions $c_u^{free}(s,l) = c_u(l) - \sum_{b\in\mathcal{B}} c_u[b] \cdot u(s,l,b)$, $a_{hot}(s,l) = \sum_{b\in\mathcal{B}_{hot}} a(b) \cdot u(s,l,b)$, and $q(s,l) = \frac{c_u^{free}(s,l)}{a_{hot}(s,l)}$. The adaptation of the above algorithm is done by the reformulation of the condition in Algorithm 1 by $\left(\forall s \in \mathcal{S} \forall l \in \mathcal{E} : c_u^{free}(s,l) \geq c_u^{inc} \cdot u(s,l,b^*)\right)$. For the acceleration purposes, Equation (3) changes to

$$c_u^{inc} = \max\Big(1, \min_{s\in\mathcal{S}, l\in\mathcal{E}: u(s,l,b)>0} \Big(\lfloor \frac{q(s,l)\cdot a(b)}{h} \rfloor\Big)\Big). \quad (4)$$

4 Numerical Results

In this section we illustrate the performance gain by the accelerated algorithm and make the improvement of the enhanced versus the simple resilience extension visible.

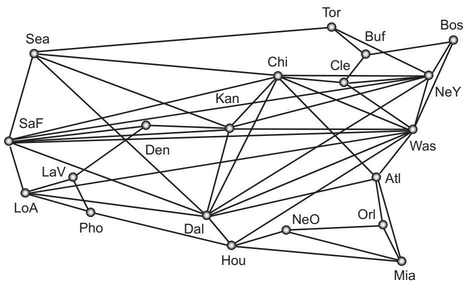

Figure 1: Topology of the test network.

The network under study is given in Figure 1 and the traffic matrix is proportional to the population (cf. Figure 2) of our virtual test network. The offered load is scaled such that it is in a reasonable relationship to the 1 $Gbit/s$ links in the network.

Name(v)	*(v) [10³]*	*Name(v)*	*(v) [10³]*
Atlanta	4112	Los Angeles	9519
Boston	3407	Miami	2253
Buffalo	1170	New Orleans	1338
Chicago	8273	New York	9314
Cleveland	2250	Orlando	1645
Dallas	3519	Phoenix	3252
Denver	2109	San Francisco	1731
Houston	4177	Seattle	2414
Kansas	1776	Toronto	4680
Las Vegas	1536	Washington	4923

Figure 2: Population of the test network.

4.1 Impact of the Acceleration Algorithm

From a rough runtime analysis we expect a linear growth of the runtime regarding the scaling factor for the basic algorithm and about a logarithmic behavior for the accelerated version. We have implemented our algorithms in Java and run them on a Pentium IV 2Ghz standard PC. We scale both the traffic matrix and the link bandwidth by the same factor and measure the runtime of the program. Figure 3 shows the runtime depending on the scaling factor and illustrates as expected the superiority of the accelerated method over the basic algorithm.

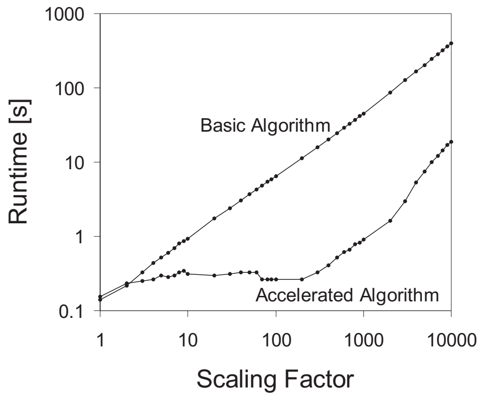

Figure 3: Runtime comparison of the basic and the accelerated algorithm.

4.2 Comparison of the Simple and Enhanced Resilience Extension

If networks are well designed for the offered load, the simple and the enhanced extension for resilience requirements lead almost to the same results. However, networks are static and traffic load changes such that they do not always fit together. In such a case, the enhanced extension method leads to more efficient budget assignments.

We take the network in Figure 4 and consider only a single failure scenario for resilience. We assume that the aggregate flows f_0 and f_1 have the same offered load. For the sake of simplicity, we indicate the budgets by their corresponding aggregate flows. The simple resilience extension calculates $c[s_0, f_0] = c[s_0, f_1] = 5 \ Mbit/s$ for the case s_0 without any failure, and $c[s_1, f_0] = 7.5\,Mbit/s$, $c[s_1, f_1] = 2.5\,Mbit/s$ for the case s_1 that the 5 $Mbit/s$ link fails. Hence, the allowable budget capacities are $c[f_0] = 5\,Mbit/s$ and $c[f_1] = 2.5\,Mbit/s$.

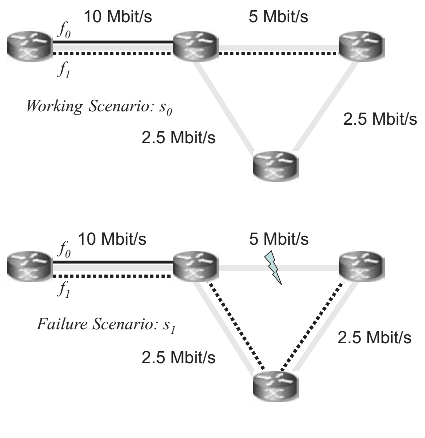

Figure 4: Small networking scenarios.

The enhanced resilience extension raises both budget capacities concurrently until $c[f_1] = 2.5\,Mbit/s$ is fixed due to the failure scenario s_1. Then, the other budget can take advantage of the full remaining capacity of the 10 $Mbit/s$ link and it is finally set to $c[f_0] = 7.5\,Mbit/s$.

This small example illustrates the operation of both algorithms and shows that the enhanced resilience algorithm leads to more efficient results than the simple version. To show that this phenomenon is not a pathological artefact, we validate this finding in the Lab03 network whose links are provisioned with 1 $Gbit/s$. We dimension the budgets with both resilience extension methods under consideration of all possible single link failures and limit their size by a minimum budget blocking probability of 10^{-6}.

The budget sizes are significantly larger if they are calculated by the enhanced resilience extension instead by the simple one. Figures 5 and 6 present a distribution of the absolute and the relative capacity gain by the enhanced resilience extension compared to simple one. More than half of the budgets remains unaffected and does not profit from the enhanced resilience extension. The additional budget capacity of the increased budgets differs considerably and the distribution for the absolute and the relative gain is different because the traffic matrix is heterogeneous. The average absolute gain is about 2 $Mbit/s$ per budget and the average relative gain is about 6.6% per budget. Figure 7 shows the difference of the respective logarithmic blocking probabilities. The budget blocking probabilities obtained with the enhanced resilience extension are up to 4 orders of magnitude smaller than those obtained with the simple resilience extension, and on average this advantage is 0.47 orders of magnitude. Hence, the benefit of the enhanced resilience extension is also clearly visible in large networks.

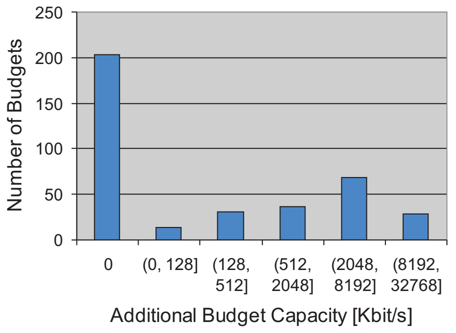

Figure 5: The enhance resilience extension assigns larger budget capacities than the simple resilience extension.

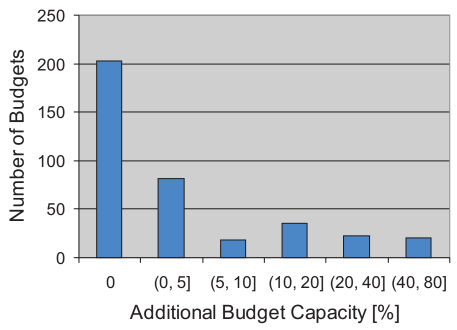

Figure 6: The enhance resilience extension assigns larger budget capacities than the simple resilience extension.

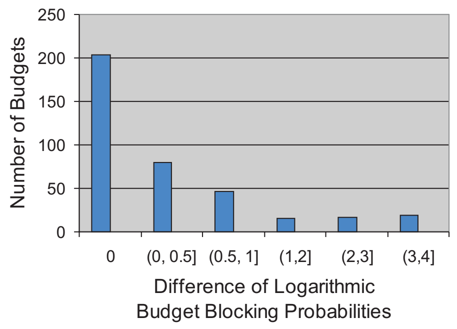

Figure 7: The enhance resilience extension obtains smaller budget blocking probabilities than the simple resilience extension.

5 Conclusion

Network resilience against network failures together with Quality of Service (QoS) can be achieved by traffic rerouting in case of local outages combined with smart capacity planing. The border-to-border (b2b) budgets (BBB) for Network Admission Control (NAC) is best suitable for that purpose [1]. The capacity of the BBBs is the base for admission control purposes and must reflect the physical network capacity to guarantee QoS [2]. In this work we presented an accelerated capacity assignment algorithm for BBBs and suggested a simple and an enhanced resilience extension to consider rerouting in failure scenarios.
The numerical results showed that the acceleration of the assignment algorithm reduces the computation time significantly, especially for networks with large links and heavy traffic load, such that it makes our proposal feasible for application in real networks. Furthermore, we gave a small example which illustrates the advantage of the enhanced resilience extension over the simple one. In large networks, the enhanced algorithm effects that 50% more traffic can be admitted without violating any QoS requirements.

Acknowledgment

The authors would like to thank Prof. Tran-Gia for the stimulating environment which was a prerequisite for that work.

References

[1] Michael Menth, Stefan Kopf, and Joachim Charzinski, "Network Admission Control for Fault-Tolerant QoS Provisioning," in *High-Speed Networks for Multimedia Communication (HSNMC04)*, Toulouse, France, June 2004.

[2] Michael Menth, Sebastian Gehrsitz, and Jens Milbrandt, "Fair Assignment of Efficient Network Admission Control Budgets," in 18^{th} *International Teletraffic Congress*, Berlin, Germany, Sept. 2003, pp. 1121–1130.

[3] COST-279 Management Committee and Michael Menth (Eds.), *COST-279 Midterm Report: Analysis and Design of Advanced Multiservice Networks Supporting Mobility, Multimedia, and Internetworking*, ARACNE, Roma, 1st edition, Jan. 2004.

[4] Vern Paxson and Sally Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226–244, June 1995.

[5] James Roberts, Ugo Mocci, and Jorma Virtamo, *Broadband Network Teletraffic - Final Report of Action COST 242*, Springer, Berlin, Heidelberg, 1996.