Optimization Flow Control in IP Networks:
Distributed Pricing Algorithms and Reality Oriented Simulation

Marek Małowidzki* and Krzysztof Malinowski**

* ** Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warszawa, Poland
** Research and Academic Computer Network (NASK), ul. Bartycka 18, 00-716 Warszawa, Poland

* Military Communication Institute (WIL), 05-130 Zegrze, Poland
malowidz@wil.waw.pl, k.malinowski@ia.pw.edu.pl

Keywords: Optimization flow control, congestion pricing, distributed asynchronous algorithm, IP networks

Abstract
The paper deals with reactive flow control in a communication network where the objective of the control is to maximize the total utility of all sources over their transmission rates. The control mechanism is derived as a price adjustment algorithm, formally to solve the dual problem of the price method. The paper examines the workability of implementation of various proposed price adjustment algorithms in IP-based networks and discusses the feasibility of using prices generated during optimization periods as a base of charging the users. The detailed simulation results are presented.

I. INTRODUCTION
Rapid development of fiber technologies, such as Wavelength Division Multiplexing (WDM), allows to hope that telecommunication networks of tomorrow will be able to offer impressive capacities that will enable meeting ever growing user requirements, especially in the area of advanced real-time services. Some authors predict (see [1]) that over-engineering (providing enough capacity to meet possible peak demands) will eliminate network congestion phenomena. Indeed, the present Internet traffic growth of about 100% per year [2] may be compensated by available technology solutions. Thus the question may be asked if it is reasonable to waste time dealing with reactive flow control issues, where one of the main targets is to provide the means to avoid congestion – perhaps it would be better to wait till tomorrow. Yet, while we do not feel competent to authoritatively answer this question, we believe that at some point the future may not appear so rosy. In other words, we rather concentrate here on the problem and keep away from risky fortune telling.

The paper is devoted to reactive flow control where the objective is to maximize the total utility of all sources over their transmission rates. The control mechanism is derived as a gradient projection algorithm to solve the dual problem. It is important to notice that the control mechanism provides the means to take the network from a congestion state. Besides, the prices generated by the proposed distributed mechanisms may form a base for some kind of a charging scheme.

There are two main goals of the paper. The first one is to analyze the suitability of the approach presented in [3], [4] and [6] to control a data network so as to avoid congestion. We assume here the concrete network type and the concrete protocol stack, that is, an IP network with its Internet Protocol running on best-effort routers (some basic QoS mechanisms may be applied). We are concerned with a wide area network (a carrier network), which provides Internet service for relatively big customers, such as middle or large sized companies, their branches or university campuses. As it will be shown later, this fact has some important consequences for the considered network model.

The second goal is to propose possible implementations of several possible flow adjustment algorithms and to present realistic simulation results.

The paper is organized as follows. Section II presents the network model and the flow optimization problem. Section III describes our simulation model and section IV presents the algorithms tested. In Section V it is described how the simulations have been implemented. Section VI presents the simulation results and Section VII discusses possible charging schemes based on the prices as generated by the distributed algorithms. Section VIII contains final remarks and proposition for future work.

II. NETWORK MODEL AND BASIC FLOW OPTIMIZATION ALGORITHM
The network model is taken from [3] and all the symbols used in this paper are consistent with that work. Let us recapitulate this model here.

A network consisting of a set of unidirectional links is considered. In the basic model each link has capacity \(c_i\), \(i \in L\). The network is shared by a set \(S = \{1,...,S_n\}\) of traffic sources; source \(s\) is defined by a four-tuple \((L(s), U_s(\cdot), x_{s,\min}, x_{s,max})\), \(L(s) \subseteq L\) is a subset of links that source \(s\) uses to transmit information to one or more destinations at the egress points of the network. \(U_s(\cdot)\) is source utility function defined over interval \(I_s = [x_{s,\min}, x_{s,max}] \subseteq \mathbb{R}_+\), with the values in \(R\), where \(x_{s,\min} < x_{s,\max}\), with \(x_{s,\min}\) respectively, minimum and maximum transmission rates that source \(s\) may wish to transmit and
$U_i(x_s)$ for $x_s \in I_s$ is utility attained when source $s$ transmits at rate $x_s$. For each link $l$ let $S(l) = \{ s \in S : l \in L(s) \}$ be the set of sources that use this link. Observe that for $s \in S(l)$, if only if $s \in S(l)$. Let $I = I_1 \times \ldots \times I_{S_m}$.

The basic Flow Optimization Problem can be then formulated [3] with the objective to choose source rates vector $x = (x_s, s \in S)$ so as maximize the sum of source utilities:

$$\max_{x_s \geq 0, 1 \leq l \leq L} \sum_{s \in S(l)} U_i(x_s)$$

subject to $\sum_{s \in S(l)} x_s \leq c_l$,

$$\text{FOP:} \quad \text{subject to } \sum_{s \in S(l)} x_s \leq c_l, \quad l \in L$$

If the feasible set is nonempty (i.e. if $\sum_{s \in S(l)} x_s \leq c_l$) and the performance function is strictly concave – in particular if each $U_i(\cdot)$ is strictly concave over $I_s$, then the unique maximizing solution $\hat{x}_s(p^* x_s)$, called the primal optimal solution, exists.

The above basic problem, with additively separable objective functions and capacity constraints, is a particular instance of a complex optimization problem which can be solved by dual method using price coordination (e.g. [7], [8], [9]).

The Local (Source) Problems are:

$$\max_{s \in I, t} U_i(x_s) - p^* x_s$$

$$\text{LP}_s: \quad \text{where } p^* = \sum_{l \in L(s)} p_l$$

Each source can, independently from others, solve the above local problem for given price $p^*$; it is important to note that the local utility function $U_i(\cdot)$ may not be known to other users and to the network operator as well. The solution of $\text{LP}_s$ is denoted as $\hat{x}_s(p^*)$ and the associated optimal value of $\text{LP}_s$ objective as $B_s(p^*)$.

The dual problem to FOB, defined through the solutions of $\text{LP}_s$, $s = 1, \ldots, S_m$, is the key to distributed algorithm for adjusting prices $p^*$. Since the Lagrange function of the FOB problem is:

$$Lg(x, p) = \sum_{s \in S} U_i(x_s) + \sum_{s \in S} \sum_{l \in L(s)} p_l c_s - \sum_{s \in S} x_s = \sum_{s \in S} (U_i(x_s) - p^* x_s) + \sum_{l \in L(s)} p_l c_s$$

the link prices $p_l$ (i.e. the Lagrange multipliers associated with the capacity constraints) can be computed through the minimization of the dual function. The Dual Problem is:

$$\min_{p \geq 0, 1 \leq l \leq L_m} \sum_{s \in S(l)} B_s(p^*) + \sum_{l \in L(s)} p_l c_s$$

$$\text{D:} \quad \text{where } p^* = \sum_{l \in L(s)} p_l$$

The basic distributed synchronous link algorithm proposed in [3], which, in fact, is just the basic descent algorithm for dual function minimization with price projection on $\mathbb{R}_+^L$, is as follows:

$$p_{l}(t+1) = [p_{l}(t) + \gamma (\sum_{s \in S(l)} \hat{x}_s(p^* x_{s_i} - c_i))], \quad (5)$$

where $\gamma = \max(y, 0)$. In the above eqn. (5) $p(t)$ denotes the value of the $l$-th link price at iteration instant $t$; the same notation is used for $p^*(t)$. Thus, in the presented synchronous version of the distributed price adjustment algorithm all sources receive, at a given time $t$, prices $p(t)$, compute the respective source prices $p^*(t)$ and then the solutions of $\text{LP}_s$, $s = 1, \ldots, S_m$. The obtained values of source rates $\hat{x}_s(p^*(t))$ are then signaled to the links, where the new values of link prices $p(t+1)$, $l = 1, \ldots, L_m$, are computed according to link algorithm $A_{l\text{ink}}$: the iteration index is advanced by one and so on. This basic algorithm requires full time synchronization; the new values of the source rates and the link prices should be computed only after all information bearing current iteration index (time marker) is signaled through the network.

Convergence of the basic algorithm can be easily established using general results available for coordination by price instruments [8]; in particular under the condition that source utility functions $U_i(\cdot)$ are increasing and strongly concave on $I_s$. The theorem provided in [3] establishes convergence under the assumption that each $U_i(\cdot)$ is twice continuously differentiable on $I_s$, $-U_i''(x_s) \geq 1/\alpha_i$ and $0 < \alpha_i \leq \alpha_m$, where $\alpha_m = \min_{s \in S} |L(s)|$, $\alpha_m = \max_{s \in S} |L(s)|$ and $\alpha_m = \max \alpha_i$. In other words the projection descent algorithm (5) is convergent for sufficiently small stepsize $\gamma$.

Of much interest is the asynchronous distributed algorithm, also proposed and investigated in [3]. This allows both the sources and the link algorithms to use weighted averages of past values of, respectively, link prices and locally optimal source rates – assuming that the differences between current time index $t$ and time markers of those past values are bounded from above. Obviously, the convergence of such asynchronous version of the distributed price coordination strategy may be slower but can again be proved under the above assumptions. The algorithm is described in Section IV.

Malinowski proposed in [6] the Modified Flow Optimization Problem, which relaxes significantly the original model. It assumes that routing may be done through multiple paths and (what is more important from the point of view of this work) that traffic generated by sources may vary due to flow control rules (shaping, policing) during congestion. This algorithm is also described in Section IV.

III. SIMULATION MODEL

In this section we describe our simulation model. The model should be possibly close to the Internet reality. We are concerned with IP traffic in the network layer. The simulation model components are: the source, its broker, the utility function and the communication mechanisms.

A. Traffic Source

It is assumed that a source represents a “big” user, that is, a company, its branch or a university campus. More
technically speaking, a source may group some number of LANs. A private user, who uses a modem to get access to the Internet, is beyond our interest. A typical private user generates little traffic and pays a flat fee for the access. He is offered best-effort quality and in a rare case when he needs a special, better, quality of service, for example for interactive applications, he will use an appropriate service provider.

The assumption about source being a big user has some important consequences. First, its traffic profiles should be relatively “inert” and repeatable (in similar circumstances, e.g. traffic generated within the same period of a typical weekday should possess similar characteristics). Second, what is ever more important, traffic generated by such a source is generally different from an optimum rate that a source has chosen for this time period. The simulation experiments presented in [3], [4] and [6] assume that each source generates traffic at an optimum rate, calculated from the source utility function and for current network prices. In our case, while it is possible to try to predict traffic rates for the next period, it is impossible to know them for sure.

B. Broker

To make the source model more concrete, we introduce here the concept of a source broker. Such broker represents the source in contacts with the carrier network; the broker knows the source utility function and works as the source’s accountant. Moreover, brokers are trying to predict the transmission rates of their sources for the subsequent periods. There are two possible methods to achieve that. The first one would use the history logs with the past source activities; similar periods from the past could be taken as a base for prediction. Additionally, some kind of artificial intelligence tool (e.g. an artificial neural network) could be used. The second approach would involve current traffic analysis with approximation and extrapolation. That is, a broker would approximate the source transmission profile using a smooth function and then use extrapolation to assess this profile for the next period. The second approach would require that the traffic profile fluctuate relatively slowly (of course, smoothed profiles are concerned here), when compared with the price adjustment period. We believe this is valid due to big source characteristics, although further analysis of traffic profiles would be needed.

A source with its broker is presented in Fig. 1. The broker controls the source’s output router, which performs shaping on the outgoing traffic to make it consistent with the desired source rate \( x_s \), computed by the broker for a given period. Typically, the broker would be implemented as an application, running somewhere within the user own network, but typically “close” to its output router.

Fig. 1: The source model

Further in the paper, if no danger of misunderstanding appears, the terms “source,” “user” and “broker” may be used interchangeably and, unless otherwise stated, will mean the same thing.

C. Utility function

All the algorithms proposed in [3], [4] and [6] assume that the utility functions are increasing, strictly concave and twice differentiable on their domains \([x_{s\min}, x_{s\max}]\). This assumption is required to prove the convergence. One may ask if such assumption is realistic. Let us analyze then a typical utility function.

Below some minimal transmission rate the service that the user is interested in will not be able to work at all so the utility will be zero. On the other hand, there exists capacity large enough to make the user unable to consume it. That is, above some threshold, the utility function should remain constant. This is rather simple and obvious, but what shape this function should take between those extreme values?

Typical utility functions will be simply non-decreasing, as those shown in Fig. 2. The most intuitive utility function seems to be linear, and it even does not have to be continuous (see Fig. 2a). Most authors assume utility functions as having sigmoid shapes (Fig. 2b). However, utility function also might exhibit a step pattern, as shown in Fig. 2c. This may happen if the user would like to use several services with different quality of service requirements (in this case, QoS factor is the capacity).

Fig. 2: Utility functions
We can approximate linear function with a strictly concave one but we could equally well approximate it with a convex function! Additionally, for the sigmoid case, there is a rate value above which such a function is strictly concave. This rate is \( x_{s,\text{min}} \) of our model. The solution for the step function also exists: the source may be modeled as several sources with sigmoid utility functions that sum up to the original step function.

To summarize: the utility functions may be treated as strictly concave within \([x_{s,\text{min}}, x_{s,\text{max}}]\) range, assuming that \( x_{s,\text{min}} \) is correctly chosen. The minimum rate, \( x_{s,\text{min}} \), should be guaranteed in a Service Level Agreement between the user and the network operator. Moreover, the user should pay a flat fee for traffic within \([0, x_{s,\text{min}}]\) range and pay extra only for traffic exceeding this value. In simulation experiments \( x_{s,\text{min}} \) is usually set to 0 but it is worth noticing that providing nonzero guaranteed values in IP network is not a trivial task.

As our source is unable to generate traffic of a given rate “on demand,” one more question arises: should the utility functions be constant over time or may they vary? The maximum rate, \( x_{s,\text{max}} \), changes from period to period. Should this change affect the utility function? We believe so and propose a way to modify the utility functions later in the paper.

D. Link-source communication

The links - strictly speaking routers with interfaces connected to given links - must in some way inform the sources about the prices calculated for the next optimization period. This is necessary for the sources to compute the optimal transmission rates. The question arises about how could the communication be implemented to impose only a limited, reasonable, overhead traffic?

The simplest naive implementation, where every link sends a message to every source, is impractical and cannot be used in real networks. First, the number of sources using a given link may be very high. How could a link know about all served sources? It would have to examine packets or it would have to be told about this fact; both ways are very inconvenient. Sending all the messages to all sources would cause packet storms, which could cause serious bottlenecks. Second, this approach would require sources to know everything about their routes; how many messages should sources wait for?

Fortunately, there are some more realistic solutions. From equation (4) it can be seen that in fact the source needs to know only the total price for the whole path to the destination. In [5] the authors propose the so-called one bit marking scheme, using one bit in the IP packet header, to carry information about the price. While this approach looks promising, it may be difficult to use in our scenario setup. Due to the fact that we are aiming at using network prices as a base for charging the users, we need the exact and secure information about prices. Thus, we propose here a more traditional approach. At the end of a subsequent optimization period, every ingress router in the carrier network generates a price message for every connected source and all its destinations. The messages travel towards their common destinations and the links (routers) along the route update prices carried in the price messages. The egress routers (connected with destinations) send these messages back to the sources. It is assumed that the input routers are able to send pricing messages along all routes being used by a given source; this may require a complex path configuration and is not a trivial task.

One final remark about the synchronization. It is assumed here that the routers in the carrier network are synchronized, with an accuracy interval that is much shorter than the price adjustment period, which – supposed that this period is in the order of tens of seconds – is feasible. The computation periods of the sources would be slightly shifted in time, due to time required for the pricing messages to reach the source.

E. Source-link communication

The basic price algorithm (5) required also communication in the reverse direction, that is, from the sources to links. This could be achieved in a way described above. However, this would add even bigger communication overhead. Moreover, in our source model there is one more reason for this information transfer to be impractical: the brokers do not know the rate in advance! (They try to predict the rate and later apply shaping, but the real rate could be much smaller or too big to buffer.) Thus, the only choice is to resort to statistics available in routers such as traffic incoming on a given interface or an output queue length.

Low proposed later in [3] a way to estimate source rates using only local information with a proof that the optimality is still maintained. This is further discussed in the following subsections.

F. Routing

We assume that routing between any source-destination pair involves a single fixed path. Despite that some algorithms allow traffic splitting among two [4] or even more [6] paths, the present Internet routing practically uses single paths only; moreover, collecting and signaling prices in multiple path routing cases seems to be even more complex.

IV. TESTED ALGORITHMS

In this section present several distributed price adjustment algorithms and discuss their implementations, that is, the interpretation of the parameters (e.g. \( c_i \)) and variables \( x_s \) in the context of an IP network.

The algorithms may be divided into two main categories: traffic-based methods, which make use of traffic generated by the sources or flowing through the network, or queue-based schemes, that compute prices on the basis of the routers queues. The advantages and the possible flaws of both approaches are discussed.

A. Algorithm I: traffic-based, basic variant

The first obvious idea is to treat \( c_l \) from equation (5) as the link capacity and \( \sum_{s \in S(l)} x_s(p(s,t)) \) as the average incoming traffic for link \( l \) over an optimization period \( t \) (the links
cannot make use of the traffic generated by sources, for the reasons explained earlier).

Note that the sum of the source rates in equation (5) and the average incoming traffic are not the same thing. The difference appears during the congestion periods, when some packets may get lost and when average incoming traffic may decrease from link to link. This can have some subtle consequences on the operation of the traffic-based price adjustment algorithm. It will be interesting to compare the performance of this approach, based on traffic flowing through the links, with theoretical source-rate approach, when the links know traffic rates generated by the sources (where \( \hat{x}_s(p^s(t)) \) represent these rates). Provided the results do not differ significantly, this will prove that one can simplify the implementation greatly without significant performance degradation.

**B. Algorithm II: traffic-based with history (asynchronous)**

In [3] Low and Lapsley propose to use the aggregate past source rates at link \( l \) (15)-(19) in [3]) as follows:

\[
p_{l}(t+1) = [p_{l}(t) - \gamma \dot{P}_{l}(t)]\]

\[\dot{A}_{l}(t) = c_{l} - \hat{x}_{l}(t)\]  \hspace{1cm} (6)

\[\hat{x}_{l}(t) = \sum_{s \in S(l)} \hat{x}_{ls}(t)\]  \hspace{1cm} (7)

\[\dot{x}_{ls}(t) = \sum_{l \neq \epsilon} a_{ls}(t', t) x_{s}(t'), \quad s \in S(l)\]  \hspace{1cm} (8)

where

\[
\sum_{l \neq \epsilon} a_{ls}(t', t) = 1, \quad a_{ls}(t', t) \geq 0, \quad \forall l, s \text{ with } s \in S(l)\]  \hspace{1cm} (9)

The proposed scheme is very general – there may be many possible policies of how to choose the values of \( a_{ls} \) parameters. For our purposes it is important to note that there is no clear reason to vary the values of those parameters between the sources, so we can assume that \( a_{ls}(t', t) = a_{s}(t', t) \). Thus, (7) can be written as

\[
\hat{x}_{l}(t) = \sum_{l \neq \epsilon} a_{ls}(t', t) \sum_{s \in S(l)} x_{s}(t'), \quad s \in S(l)\]  \hspace{1cm} (10)

Note that this algorithm is equivalent to the previous traffic-based approach with the *latest data only* when with \( a_{s}(t', t) = 1 \) for \( t = t' \) and \( a_{s}(t', t) = 0 \) otherwise. In our simulations we arbitrarily assume that the older the data the less important they are, so we decrease the value of \( a_{s} \) twice for every earlier period (i.e. \( a_{s}(t'-1,t) = 0.5a_{s}(t', t) \)).

Since this algorithm is a more general form of that given by eqn. (5), we will be using in our simulations the simpler name traffic-based for both cases.

**C. Algorithm III: traffic-based with headroom**

In [6] it is assumed that the total link capacity, \( c_{l} \), is divided between the steady-state flow capacity \( c_{l} \) and the headroom \( h_{l} = c_{l} - c_{l} \), where \( c_{l} \leq c_{l} \) for all links \( l \). The headroom allows the constraint

\[
\sum_{s \in S(l)} x_{l}(t) \leq c_{l}, \quad l \in L\]  \hspace{1cm} (11)

to be temporarily violated, and such situation indicate a pre-congestion state.

The Modified Local (Source) Problems are defined as

\[
\max_{x_{ls}} U_{s}(x_{s}) - x_{s} p_{s}(t)\]  \hspace{1cm} (12)

\[p_{s}(t+1) = [p_{s}(t) + \gamma(\sum_{s \in S(l)} x_{s}(t) - c_{l})] \]  \hspace{1cm} (13)

where \( p_{s}(t) = \sum_{l \in L_{s}} p_{l}(t) \frac{x_{l}(t-1)}{\hat{x}_{l}(p^s(t-1))} \)

and the Modified Algorithm with Feedback for the links is

\[p_{s}(t+1) = [p_{s}(t) + \gamma(\sum_{s \in S(l)} x_{s}(t) - c_{l})] \]  \hspace{1cm} (13)

In our model, \( \hat{x}_{s} \) would be a *total traffic* outgoing from broker \( s \) while \( x_{s}(t) \) denote traffic *forwarded* on the link \( l \) during period \( t \).

It is worth to make a few observations related to MFOP. First, in the case of multiple paths routing, the routers must track the packets coming from the individual sources and to maintain appropriate statistics. Second, the price modifications in one step are limited, which may slow down the process of price adjustment but also limits price variations between the successive time instants. Besides, the algorithm allows routing to vary and routers to split traffic in whatever way they find it convenient; however, collecting statistics related to a given source would be a real challenge. Finally, the MFOP is a step towards pricing for traffic delivered to destination, although it is probably still quite away from a practical model.

**D. Remarks about traffic-based approaches**

Unfortunately, all traffic-based algorithms have one important flaw that is not immediately visible from the equations. Recall that we are concerned with the user traffic in the *network (IP) layer*. We know link capacities in the network. However, these are the capacities in the physical layer - so it tells how fast bits are propagated “over the wire.” How to compute the capacities in the network layer, so that \( \hat{\gamma} \) and \( c_{l} \) could be expressed in fully understandable terms? The answer is that this is not outright possible. The following factors affect the capacity in the network layer:

- Data link layer overhead, which includes data link layer protocols headers, data link layer fragmentation as well as bit or byte stuffing. This decreases available capacity. The interesting thing is that the exact overhead depends on the traffic pattern and even on particular data carried in the IP packets,
- Data link layer compression (e.g. [11], [12]), which increases available capacity.

It may seem astonishing at first that thanks to data compression the available capacity in the IP layer may in fact be greater than the raw physical layer capacity \( c_{l} \). It may also be much smaller. Anyhow, we can regard \( c_{l} \) as
some estimation of the IP layer capacity, but we should keep in mind that this is a rough estimation.

E. Algorithm III: queue with threshold
There is a possibility to resort to output-queue-based interpretation for \( \hat{x}_l \) and \( c_l \). In this approach, \( \hat{x}_l \) is an average output queue length for link \( l \) over a given period and \( c_l \) is set to some fraction of the queue capacity, the threshold that means a "congestion" state.

F. Algorithm IV: queue length
Algorithm A2 proposed in [3] also removes the need for explicit communication of the source rates to the links and is based on a queue length. The price is directly proportional to aggregate backlog \( b_l(t) \) at link \( l \) at time \( t \) (the aggregate length of link \( l \)’s output queue at the end of a given period). The algorithm assumes (eq. (16)) that the queues are unbounded and that the capacity of a given link \( l \) is distributed "fairly" between traffic originating at different sources, i.e. if a queue length is increased (decreased), no queue fraction coming from an individual source is strictly decreased (increased). It should be understood that the changes in the queue length should affect all sources.

The price \( p^2(t) \) is expressed as:
\[
p^2_l(t) = \eta_l(t), \gamma > 0
\]
where
\[
b_l(t+1) = [b_l(t) + \hat{x}_l(t) - c_l]^{+}
\]
it is assumed that there exist
\[
\sum_{n \in S(l)} \theta_n(t) = 1
\]
such that
\[
\eta_n(t) = \theta_n(t) \eta_l(t)
\]
where
\[
\eta_l(t) = b_l(t+1) - b_l(t)
\]
\[
\eta_l(t) = b_l(t+1) - b_l(t)
\]
In eq. (15) it is assumed that each time period is of a unit length. The above approach differs significantly from all other algorithms in that it does not use the average values but rather an absolute value of the queue length at the end of a given period.

G. Remarks about the queue-based approaches
Both the traffic-based and the finite queue algorithms are to some degree equivalent. The first approach allows for larger price changes prices during link congestion periods (due to limited queue sizes). In the second approach one is able to estimate better the available link capacity (the property of price variations limitation may, in fact, an advantage, see conclusions about pricing in section VIII).

V SIMULATION
The simulator used for this project was OPNET Modeler 7.0.B for Windows 2000 [13]. OPNET is a packet network simulator that allows, among others, designing, verifying and simulation of IP networks. While OPNET had all the components necessary to prepare and run simulations of wide-area IP networks, its models had to be modified to support the algorithms tested here.

A. Router model in the OPNET simulator
The router model in OPNET is based on the classical router architecture with multiple interface cards and memory-based switching fabric. While this is not the most modern architecture, there are still many such routers working in the Internet. All the components needed for simulation of the algorithms tested in the paper were prepared by modifying the router model.

B. Source
The source is composed of two independent elements: a traffic generator and a broker; both are modeled using the modified OPNET router model (see Fig. 3). The generator emits traffic, according to some pattern, and thus models the source LANs and their traffic. The generator is connected with the broker’s router, so the whole traffic sent by the generator flows through this router. The broker is allowed to affect the traffic in any way it sees fit. Then the traffic is forwarded to the input router in the carrier network.

C. Source’s traffic generator
The traffic generator sends IP packets according to the prepared traffic profile (an average rate expressed in bytes per second). Packet sizes belong to the [28,1500] byte range (minimum and maximum IP packet lengths in Ethernet); additionally, their size distribution is taken from traffic observed in LANs (where, for example, the packets of 40 and 1500 bytes definitely prevail and make up about two-thirds of the total traffic).

This is a simple generator model and no feedback is involved. The result is that this makes a connectionless source model (UDP) rather than a connection-oriented (TCP) one; the traffic of the latter type (e.g. HTTP or FTP) is poorly emulated. In fact, TCP control mechanisms should affect the TCP part of the source traffic, especially during the congestion periods, and this is not modeled here.
D. Broker

The broker has two main tasks:
- To predict the maximum rate \( x \) that the generator might use during the next period. This will serve as \( x_{s,\text{max}} \) for the next period and will be used to modify the source utility function for this period.
- To shape and policy the traffic flowing through the broker router to make it conformant to the optimum rate \( \hat{x}_s \) calculated from the source utility function.

The first task is quite complex and involves a few actions. Since it is assumed that the traffic rate fluctuates relatively slowly when compared with the pricing period, the pattern of generated traffic may be approximated and then extrapolated to compute \( x_{s,\text{max}} \) for the next period. Since the traffic pattern is usually fluctuating, it is exponentially smoothed first. The broker stores the past values of the traffic rates and at the end of each period, when the new \( \hat{x}_s \) is to be computed, it approximates the function of the generated traffic using a linear or a square function. The calculated function is used to compute \( x_{s,\text{max}} \). Then the predicted \( x_{s,\text{max}} \) is used to modify the source utility function. We have chosen to modify the utility function by "stretching" or "squeezing" its strictly concave part along the \( x \)-axis. Finally, the modified utility function is used to calculate \( \hat{x}_s \), i.e. the source policy for the next period. The whole process is depicted in Fig. 4.

![Broker's computational tasks](image)

**Fig. 4:** Broker’s computational tasks.

The stretching or squeezing the utility function is one of the simplest ways for its modification. One can imagine policies, which dictate that the range of the utility values should also change. For example, a company could value more traffic generated during common daily operation hours and decrease its utility for evening or night activities.

E. Link to source communication

Link to source communication, necessary for notification of the sources about the prices for the next period, is realized implicitly, that is, using global variables. This considerably simplifies the whole implementation, while it does not affect simulation results in any important way.

VI SIMULATION RESULTS

A. Comparison criteria

Most simulation results presented in [3] and [6] were obtained under the assumption that the source willingness to generate traffic changed relatively slowly, which allowed the considered algorithm to converge. By source willingness we mean here that it generates the traffic according to a stationary utility function and to prices received from the network. However, our approach to simulation is different, as we have to take into consideration another nature of our sources.

While it is still crucial to verify the convergence of an algorithm, and thus its correctness, this is only one factor of the assessment of the overall algorithm performance. Recall that our source model assumes that the traffic generated by the source is variable, closer to real traffic generated in the Internet, and may vary from one computation period to another. The situations when the network arrives to steady state, and the prices stabilize, may be rare. In other words, there may be not enough time for a given algorithm to reach an equilibrium point. In the varying conditions, the algorithm with worse convergence may in fact perform better, if it converges faster in its initial phase (that is, when it is able to approach faster the optimal point, even if this approximation is rough).

B. Simulated network

The simulated network is presented in Fig. 5. The network is purposely simple to make the analysis easier. Source 1 transmits traffic to destination sink 1, source 2 and source 3 – to sink 2 and source 4 – to sink 3.

![Simulated network](image)

**Fig. 5:** The simulated network

C. Convergence verification

The first part of simulation was aimed at convergence verification, thus traffic profiles were simple – sources 1 to 3 transmitted constant traffic, 120kB/s, 80kB/s and 40kB/s, respectively. Source 4 originally transmitted 10kB/s and increased its traffic to 100kB/s at time equal to 30s. The utility functions of the sources were set to \( U_s (x) = a_1 \sin(\pi / 2 \cdot x / x_{s,\text{max}}) \), with \( a_1 = a_2 = a_3 = 10^6 \) and \( a_4 = 2.0 \times 10^6 \). (We have chosen \( U_s(s) \) so that it models user satisfaction as depending on the \( x / x_{s,\text{max}} \) quotient rather than just on \( x \).) The step size \( \gamma \) was set to \( 5 \times 10^{-6} \). Prices were computed and reported to brokers every 1s, with some small random delay. The traffic predictor used ten 0.25s averages and linear approximation to predict \( x_{s,\text{max}} \) for the next period of one second length. Quadratic approximation proved to be too "flexible". The results are presented in Fig. 6, Fig. 7 and Table 1.
Source-rate-based, traffic-based and queue-length-based: These algorithms performed best and achieved stable points after a few steps (Fig. 6 and Fig. 7 show the results for the traffic-based approach with history length \( t-t_0 \) equal to 3). The source-based and traffic-based approaches provided almost identical results. For the traffic-based scheme, extending the history yielded bigger inertia and slightly smoother profiles after the algorithm had converged although due to the way the \( a_t \) parameters were computed, extending the history length above 3 could not change much.

![Fig. 6: Link and path prices](image)

Fig. 6: Link and path prices

![Fig. 7: Destination data rates](image)

Fig. 7: Destination data rates

<table>
<thead>
<tr>
<th></th>
<th>source</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Source-rate-based</td>
<td>4.59</td>
<td>3.62</td>
</tr>
<tr>
<td>Traffic-based</td>
<td>4.61</td>
<td>3.62</td>
</tr>
<tr>
<td>Traffic-headroom(2)</td>
<td>3.11</td>
<td>3.09</td>
</tr>
<tr>
<td>Queue-length</td>
<td>4.87</td>
<td>3.71</td>
</tr>
<tr>
<td>Queue-threshold(0)</td>
<td>2.39</td>
<td>2.87</td>
</tr>
</tbody>
</table>

Table 1: Traffic delivered to destination [MBytes]

Queue-with-threshold: The simulations shown that this scheme converged only when the threshold was set to 0 (and even in this case, a small but continuous price growth was observed). For nonzero thresholds, even small, sinusoidal oscillations and very slow convergence was observed. The explanation for this behavior is the inertia of average queue length values – even if a source modifies its rate the effects may be visible only after some time. Destination data rates (not shown) revealed that this approach significantly favors the sources with higher utility values – practically all traffic generated by source 4 had been delivered to its destination.

Traffic-based with headroom: Performed well for small queues and small headroom values. The equation (12) contains one subtle problem: due to packet buffering, there is no immediate time relationship between traffic outgoing from a source and traffic from this source transmitted through a given link. In particular, large buffers may cause traffic forwarded, \( x_t(t) \), to be much larger than traffic outgoing, \( \hat{x}_t \), and this may cause “wild” price peaks. To avoid it one may replace – in case of single path routing – eqn. (12) with basic path pricing equation used in (2).

D. “Real” Internet traffic profiles

For the purpose of the second test, artificial traffic profiles were prepared. (We could not use real profiles due to long simulation times and lack of data of required accuracy – at least one-second averages.) This required a few steps. First, appropriate step profiles were created with average traffic rates constant within subsequent one-minute periods. Then, the steps were exponentially smoothed. Next, sinusoidal fluctuations were added. Finally, real profiles taken from LAN traffic were imposed. Fig. 8 shows the final profiles: the real one for source 4 and the smoothed profiles for all remaining sources. Fig. 9 on the next page depicts traffic incoming on two most important links (between router 1 and router 3).

![Fig. 8: Source traffic profiles](image)

Fig. 8: Source traffic profiles

This time we set: \( a_1 = a_2 = 1.5*10^6 \) and \( a_2 = a_3 = 10^6 \). The stepsize \( \gamma \) was set to 5*10^5. Prices were computed and reported to brokers every 5 s. (which seems to be somewhat unrealistic). The traffic predictor used ten 0.5 s. averages and linear approximation to predict \( x_{t+\text{max}} \), for the next period. The results are presented in Fig. 10 – Fig. 15 and Table 2.
Fig. 9: Traffic incoming at two most important links

Fig. 10: Broker 4's extrapolation and received traffic

The extrapolation is good, but obviously not perfect. The exponential smoothing with $\alpha$ parameter set to 0.1 added significant delay. As previously, the approximation with higher-order polynomials did not work well. Yet, more experiments with real-world traffic are needed.

Fig. 11: Prices generated by the traffic-based approach

Fig. 12: Prices from the queue-length-based approach

Fig. 13: Destination data rates (traffic-based)

Fig. 14: Destination data rates (queue-length)

<table>
<thead>
<tr>
<th>source</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source-rate</td>
<td>12.65</td>
<td>19.28</td>
<td>12.30</td>
<td>20.74</td>
<td>65.17</td>
</tr>
<tr>
<td>Traffic-based</td>
<td>12.79</td>
<td>19.15</td>
<td>12.22</td>
<td>20.73</td>
<td>64.89</td>
</tr>
<tr>
<td>Queue-length</td>
<td>12.93</td>
<td>19.71</td>
<td>12.52</td>
<td>21.02</td>
<td>66.18</td>
</tr>
</tbody>
</table>

Table 2: Traffic delivered to destination [MBytes]

It may be observed that both traffic-based and queue-length-based approaches yielded almost the same destination rate values (the difference is of the order of a
few percent and could be caused by a few factors), although the prices generated by both schemes differ significantly. Since the queue-length approach uses temporary queue length values, the prices vary considerably and this is highly undesirable in general, because this causes sources’ rates to fluctuate significantly, too – see the example in Fig. 15.

![Graph showing traffic forwarded by broker 2](image)

**Fig. 15: Traffic forwarded by broker 2**

Results from the source-rate-based approach were, as previously, very similar to the traffic-based approach. Price profiles were similar, although lower in value and not so smooth (probably due to lack of history). Traffic profiles were almost identical.

**E. Final remarks**

**Convergence and step length (\(\gamma\)) selection.** The distributed nature of the system is undoubtedly its advantage, but it also causes serious problems in practice with appropriate step size selection. Unlike centralized schemes (e.g., neural networks learning with back-propagation), it does not enable dynamic step adjustment, and this is a disadvantage.

**Complexity.** We were aiming at assuring high level of reality in our simulations. Except for rather simple source model (lack of feedback), we think that most components were reasonably realistic. Unfortunately, the resulting model is very complex and many even low-level parameters can affect the results. Thus, we are probably still far away from the full appreciation of the impact of some important factors (e.g., proper optimization period length selection, reliable prediction, large-scale source model with feedback) and thus more research is still needed.

**VII PRICING AND CHARGING SCHEMES**

It is now time to uncover the reasons due to which we insist on making a relationship between the prices generated by the algorithms and the charging schemes according to which the users should pay for the excess traffic. We generally do not believe that the users, without such clear relationship, would like to obey the rules of the game. In other words, the financial consequences must exist to keep the system working properly.

Let us make a few points about a possible charging scheme (in the remainder of the section we will use the term ‘price’ for the coordination instruments considered so far and the term ‘fee’ for the user payments). Since the prices in a non-congested network will be equal to 0, they cannot be the only basis for calculating a fee. It is possible that during long periods a congestion will not occur, but the network operator cannot forward traffic for free. This fact suggests that a flat fee for network access should be introduced and the network users should pay additionally for traffic during the congestion periods (see also section IIIIC and the consideration about the utility functions).

However, a fundamental question still arises: will this charging scheme work in a real world, where the players are not always honest. Is this scheme designed well, without any weak points, and which would not allow a player to dishonestly increase its profits at the expense of the other players? The answer is, obviously, no. To make things worse, there are quite few possibilities to cheat, and such possibilities exist for the both sides.

Let us consider the network operator. How could he cheat to increase own revenues? There are two options:

1. The operator could pretend that the link capacities are lower than in reality (this is equivalent to a situation where the operator inflates the prices calculated on the basis of traffic and the bounds). The users may then decrease generated traffic, but still the values of \(p's\), will be higher than they should be. Note one interesting point: this situation has even one more advantage from the point of view of the network operator, he is able to earn more and also keep some unused headroom, making him able to compensate for higher traffic demands in future (and thus to earn even more),
2. The network operator can affect routing to choose worse paths, thus causing artificial congestion.

As it can be easily seen, there is no way for the network users to defend themselves against such practices, apart from changing the carrier network they are using – which may not be outright possible.

Consider now the network users. They can exploit one feature of the above charging scheme to decrease the cost of sending their traffic. The prices for the next period are calculated on the basis of traffic generated during the current period. So a wise user could restrict from sending traffic during one period to wait for a cheaper period and then “push” his packets. To defend himself from such actions, the network operator should generally avoid fast changes of the price values from period to period. Actually, the nature of a source in a model concerned here (a big, relatively inert source) makes such activities less feasible.

**VIII FINAL REMARKS**

Finally, we would like to formulate two key postulates. The first one is that routing should be taken into consideration. Note that in the model concerned here, one can have a feeling that in fact the users are penalized for how the network works. The network routing remains static and in fact the only network operator activity is to affect traffic generated by users by means of the prices. We would like to change this pattern; the network should behave dynamically, that is to foresee a possible congestion and appropriately
modify its routing to accommodate as much traffic as possible. The second postulate concerns the way the users should be charged. We have no doubts that if the users are to be charged on the basis of generated traffic, then the only acceptable policy is charging for traffic delivered to destination. This is fair to both sides; moreover, this motivates the network operator to operate better. While we realize that it would be very difficult to formulate a model able to meet these two requirements, we feel that this is the correct direction of the future research.

ACKNOWLEDGMENT
The authors would like to acknowledge the financial support provided by the Foundation for Polish Science (FNP), the financial support and network traffic data provided by the Research and Academic Computer Network (NASK) and the access to OPNET simulator provided by the Military Communications Institute (WIL). The authors would like to thank Professor W. Findeisen for his helpful comments on the pricing schemes, Dr J. Gajda for access to APREG3 approximating procedure and Mr. J. Sobczyk for consulting on the operation of the IP mechanisms.

REFERENCES

Biographies
Marek Malowidzki. Obtained M.Sc.Eng in 1996 from the Faculty of Electronics of the Warsaw University of Technology (WUT). Currently employed as a project manager in Military Communication Institute, Zegrze, Poland. He is also working towards his PhD at WUT. His main interests include object-oriented and component technologies, network management and the Internet.

Krzysztof Malinowski. Born Dec. 13, 1948, in Poland. Obtained MEng., PhD and DSc degrees, respectively, in 1971, 1974 and 1978, all from the Faculty of Electronics of the Warsaw University of Technology (WUT). In 1989 obtained title of Professor of Technical Sciences and was appointed extraordinary professor at WUT. In 1994 appointed ordinary professor at WUT. From 1984 till 1996 served as director of the Institute of Automatic Control, from 1996 till 1999 as the Dean of the Faculty of Electronics and Information Technology of WUT. Currently head of Control System Division of the Institute of Automatic Control and Computer Engineering, and director of Center for Control and Information-Decision Technology of WUT. From January 2000 employed also as a research professor at the Research and Academic Network (NASK) in Warsaw, Poland. He is the member of the Polish Academy of Sciences and the member of the Warsaw Scientific Society. Research interests: hierarchical optimization and control, control and simulation of large-scale systems, design of decision rules under uncertainty. Author or co-author of three books and over 120 journal and conference papers.