[HeDaKaSt01] *Oliver Heckmann, Vasilios Darlagiannis, Martin Karsten, Ralf Steinmetz;* **A Price Communication Protocol for a Multi-Service Internet;** Advanced Internet Charging and QoS Technology (ICQT), Wien, September 2001. S.

A Price Communication Protocol for a Multi-Service Internet

Oliver Heckmann[1], Vasilios Darlagiannis[1], Martin Karsten[1], and Ralf Steinmetz[1,2]
[1] Industrial Process and System Communications KOM, Darmstadt University of Technology, Germany
[2]German National Research Center for Information Technology, GMD IPSI, Darmstadt, Germany
Email: {Oliver.Heckmann, Vasilios.Darlagiannis, Martin.Karsten, Ralf.Steinmetz}@KOM.tu-darmstadt.de

Abstract[1]*: In this paper we present a very flexible and efficient protocol for distributing price information inside an ISP's management system and to its customers. It designed to work with multiple QoS architectures, for example Intserv and Diffserv.*

This protocol was developed as part of a multi-service Internet service provider's architecture. It supports dynamic pricing and can be used with a number of different transport mechanisms, e.g. embedding tariff messages as policy objects in RSVP messages. As tariffs can get very complex, it is possible but not necessary to send tariffs as Java code. We also discuss the problem a provider's charging and accounting system has with frequent tariff updates and how they can be solved.

The paper also contains clear definitions of tariff and related terms.

Keywords*: Network QoS, Multi-Service Internet, Price Communication, Tariffs*

1. Introduction

Today Internet access itself has become a commodity; as a next step, Internet Service Providers (ISPs) have to be able to differentiate their service offerings for a competitive market situation, based on Internet Quality-of-Service (QoS) technology as well as value-added Internet services.

One important aspect of market managed networking is pricing. In a competitive, market managed, multi-service Internet, a vast variety of different tariffs will exist, many of them might be updated regularly. In this paper a protocol for the flexible and efficient distribution of even the most complex tariffs is presented.

This protocol was developed as part of the M3I[2] project [1]. M3I stands for „Market Managed Multi-Service Internet". The Goal of this project is to design, implement and trial a next generation system which enables market managed resource management, specifically by enabling differential charging mechanisms for multiple levels of service.

The paper is structured as follows: After this introduction we try to give a clear definition of terms we use like *tariff*, *price* and *charge*. Then, we present a small overview of the M3I architecture that uses the price communication protocol. After that the price communication protocol itself is presented. In the fifth section, we show how tariff updates can be handled before presenting the conclusions in the last section.

2. Definitions

The word price is often used in different ways and there can be misunderstandings between *tariff*, *price*, *price coefficient* and *charge*. That is why those terms are defined first:

In the following context the word **service** is used for a low-level network service that an ISP offers to its customers, e.g. the Intserv [2] Guaranteed Service [3] or a service build on the Diffserv [4] AF PHB [5]. The term **session** is used to describe the actual invocation and usage of one clearly specified service.

Users typically pay for sessions so sessions have to be characterized quantitatively. The term **session characterization** is used for a set of well defined quantitative parameters (session characterization parameters) that describes all aspects of a session that are relevant for a tariff. Example parameters are duration, number of bytes or packets sent or received, guaranteed service parameters (rate, peak rate, bucket depth, etc.), number of received ECN marks, etc.

The amount of money the provider charges for a session is the **charge**, it is equal to the customer's costs. A

[1] This work is partially funded by the European Commission under the 5th Framework Programme IST, Project M3I (11429).

charge advice is used for the amount of money that the customer must pay for a (fictional or real) session. The differences between charge and charge advice are subtle but important. The charge advice is the output of the tariff while the charge is the output of the provider's charging and accounting system and is passed on to the billing system. An example will clearify that:

A customer can find out the costs for a one minute call he plans to make by looking at his mobile phone provider's tariff, it is let's say 0,39 DM. That is the charge advice. Once he makes that call his provider's charging and accounting system (CAS) stores that the customer owes the provider 0,39 DM, that is the charge.

A **price coefficient** is the partial derivative of charge advice with respect to one session characterization parameter.

If a session is characterized by only one parameter, there is only one price coefficient and this coefficient is what people often intuitively call **price** (money per minute, per kilo etc.). But following Encyclopaedia Britannica [6], a price is „the amount of money that has to be paid to acquire a given product"[2], therefore equal to the charge advice. To avoid confusion the word *price* is used in this work only as a general term.

A **tariff** is a set of rules for calculating the charge advices for sessions of one service. Thus the input of a tariff is a session characterization and the output is a charge advice. One might think that a tariff is the same as the set of price coefficients but this only holds true for the simplest cases (linear tariff algorithms) as the examples below will show.

It is useful to distinguish inside the tariff between the **tariff algorithm** and the **tariff parameters**, (see Figure 1). The algorithm describes how the *session characterization parameters* are combined with the *tariff parameters* in order to obtain the charge advice CA.

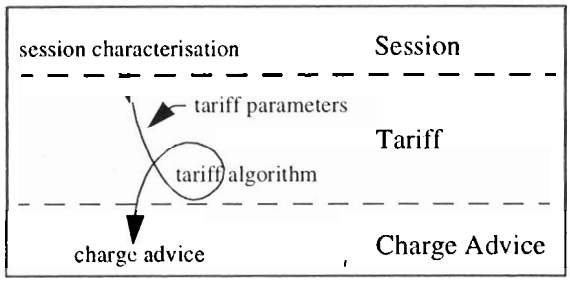

Figure 1: Tariff components

[2] or „the amount of money given or set as consideration for the sale of a specified thing" if looking it up in Merriam Webster [7].

Session characterization parameters: t, u
Tariff parameters: a, b, c
Tariff algorithm: $CA(t, u) = a \cdot t + b \cdot u^c$
Price coefficient of t: $\frac{d}{dt}CA(t, u) = a$
Price coefficient of u: $\frac{d}{du}CA(t, u) = b \cdot c \cdot u^{c-1}$

Table 1: A tariff example

An example is given in Table 1. The example also shows that the tariff parameters are not necessarily the same as the price coefficients. If a tariff algorithm is non-linear, the price coefficients are no longer constant (see the price coefficient for u).

It makes sense to distinguish between the tariff algorithm and the tariff parameters. In a good implementation this has to be done anyway to separate data and methods. It is also efficient, because typically the algorithm will not be changed as often as the tariff parameters.

3. The Architecture

The (simplified) architecture of an ISP is depicted in Figure 3. A tariff can be set manually with the enterprise policy control module (EPC). In a dynamically priced Internet tariff changes are automated and then originate the price calculation module. This module uses several data sources as input for it's decisions, for example the CAS and the mediation module (which aggregates and correlates data from the network infrastructure).

The tariff directory stores all tariffs and forwards them towards the CAS and - if it exists - the bandwidth broker (BB). Note that the CAS and the BB might be distributed systems consisting of a number of boxes.

The customers can be end-users or other providers. They can be informed about the tariffs via a push mechanism (tariff directory sends tariffs and tariff updates) and/or a pull mechanism (end-system requests a certain tariff). If the customer is a provider, it can use the pricing information as a further input source for its own price calculation module.

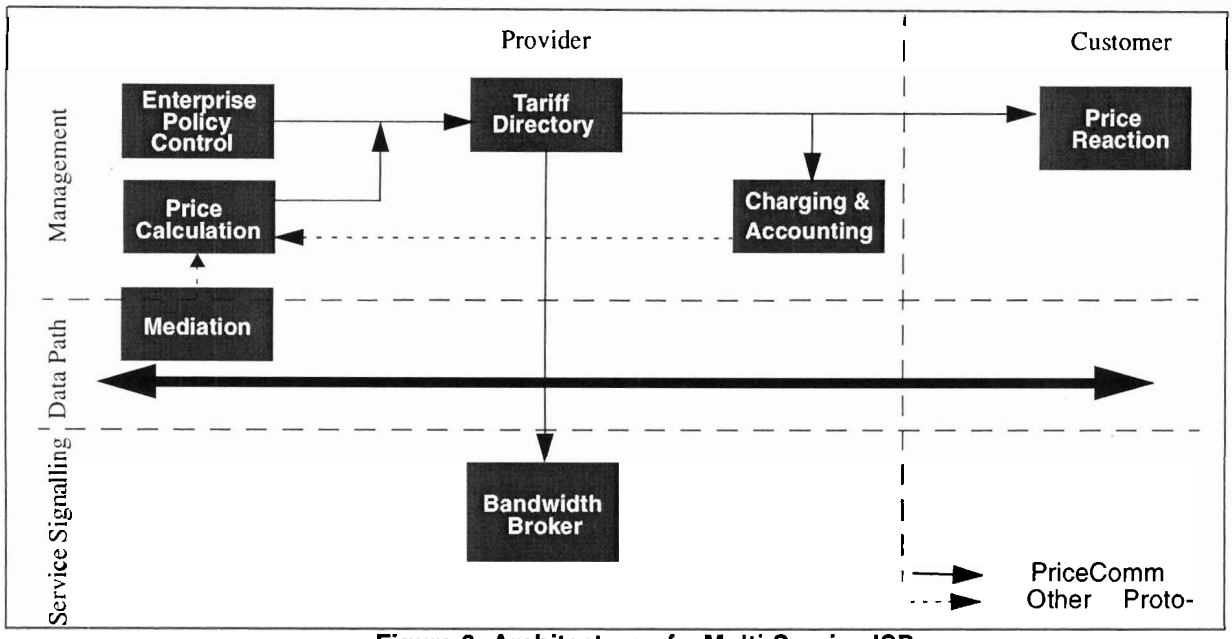

Figure 3: Architecture of a Multi-Service ISP

4. The Price Communication Protocol

We now describe what influenced the design of the price communication protocol and show some of the protocol's aspects in detail.

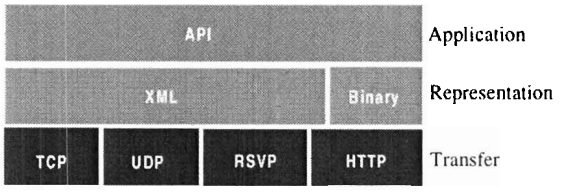

Figure 2: The 3 sublayers

On top of the protocol implementation is an API (application sublayer). Beneath the API is the price communication protocol that distributes the tariffs among the interested systems. The tariffs are encoded in a special way (representation sublayer) and sent using a transport mechanism (transfer sublayer). The sublayers are now discussed separately.

4.1 Transfer Sublayer

The protocol must have a reliable transport mode for tariffs, it is used for example between the price calcluation module and the tariff directory. In a dynamic pricing scenario, it has to transmit regular tariff updates to possibly thousands of customers, so a multicast transport mode is important too. In an Intserv environment tariff messages could also be embedded as policy objects [8] inside RSVP [9] messages.

With those reasons in mind we decided not to build the price communication protocol around one transport protocol (like UDP) but to enable it to use any combination of the following:

- plain TCP connections
- unreliable UDP multicast
- embedded in RSVP messages
- HTTP [10]

Note that using HTTP compares directly to using TCP connections. HTTP is a lot more powerful. It allows reusing existing web infrastructure like caching mechanisms. Also, firewalls can be passed easier if HTTP is used instead of direct TCP connections. TCP is intented only for minimalistic scenarios.

Because different transport mechanisms are possible, a small header has to be added to the tariff message PDU which is described in the next chapter. If plain TCP connections are used, the header contains a length field. It contains a message id in the UDP case, equals a policy object [8] for RSVP or is the content-type header for HTTP.

4.2 Representation Sublayer

4.2.1 Message format. The price communication protocol uses XML messages as PDU. It thus offers all advan-

tages of a text-based protocol (easy to design, debug, understand and expand) and automatically leads to well structured messages. It also allows the reuse of existing code like XML parsers and validators, e.g. [11]. The rules for valid messages are described in an external DTD/XML Schema file and are not hardcoded, which eases the addition or the change of fields and features, a property that is very useful in the design process.

The protocol implementation can be based on the W3C Simple Object Access Protocol (SOAP) framework [12]. The SOAP specification describes a XML-based lightweight protocol for exchange of information in a decentralized, distributed environment. It it typically used in combination with HTTP but can also use plain TCP or UDP.

Additionally, we intend to design a binary encoding of a subset of tariff messages, which can then be piggy-backed onto other protocols like RSVP.

4.2.2 How to represent tariffs. Tariff algorithms can be very complex (consider, for example, some current mobile phone tariffs). They can include special discounts (for example 10% discount after the 10th minute) and often depend on the time of the day and the day of the week (weekend...). It is unrealistic to expect that all future multi-service Internet tariffs will fit into a standardized scheme.

The protocol must therefore be flexible. It can include any description of a tariff and it's parameters. One could invent and use a highly flexible and powerful tariff definition language. We chose not to do that and use Java instead. As the distributed tariffs should also be human-readable, a textual description of the algorithm and the parameters can also be included as plain text and/or XML.

Apart from this, it is sensible to assume that there will also be a number of „standardized" tariffs, that is tariffs where the algorithm is known to everyone and therefore no code has ever to be sent around.

4.2.3 PDU. The protocol uses two basic kinds of messages: the tariff message that contains either a complete tariff or a tariff update (see Section 5) and the request message that is used to request a certain tariff.

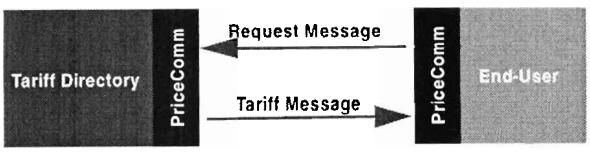

Figure 4: Pull mechanism

Figure 5 displays how a tariff message looks like. The provider must be identified globally. If this is done by using his domain name no separate central coordination authority is needed. A human-readable name can be included optionally (*name* tag) as well as authentication information. Please note that the security and authentication aspects are out of scope of this work.

Then, the tariff that is included in this message is identified. The tariff ID has to be unique only inside the provider space. It should be noted that the tariff ID is not the same as the service ID which is (with an optional human-readable name) also included, because one tariff could be used for more than one service and the other way around.

The tariff algorithm is described inside the *algorithm* tags. It has got a version number and includes optional information about when it starts and/or stops to be valid. It includes one or more descriptions of this algorithm. Different content-types can be used for the different descriptions. Our implementation currently supports Java, plain text, HTML and XML text. The same is done for the tariff parameters within the *parameters* tags.

To avoid that redundant information is sent in the case of very frequent tariff updates, the algorithm - which usually does not change too often - can be left out. Also the parameters can be tagged as incomplete which indicates that only the changed parameters are transmitted. Instead of directly including code or text inside the *description* tags a url that points to that content can be used (this is signalled by setting referenced to "true", see text/plain parameters example in Figure 5).

If a tariff message is a direct answer to a request message it references this request message in the reference tag. This is necessary because the protocol uses an asynchronous request/reply mechanism. A system can send a number of requests directly one after the other while answering tariff messages can be send in a different order and possibly with a different transport mechanism.

The second message type is the request message. It allows requesting either a special tariff or - for the case that users do not yet know, what tariff they are interested in - allows requesting tariffs for one service. Service discovery is outside the scope of the price communication protocol.

Apart from the tariff or service identification the request message contains a unique *reference* tag so the answering tariff message can refer to it.

```
<?xml version="1.0" encoding="UTF-8" ?>
<tariffMessage>

  <provider>
    bt.com
    <name> British Telecom </name>
    <authentication> ... </authentication>
  </provider>

  <tariff>
    <ID> 131 </ID>
  </tariff>

  <service>
    <ID> 40 </ID>
    <name> IntServ GS <name>
  </service>

  <algorithm>
    <version> 3 </version>
    <valid>
      <from> <date> 20.01.00 </date>
             <time zone="MEZ"> 00:00 </time>
       </from>
    </valid>

    <description content="binary/java"
                 referenced="false">
      <class name="com.bt.intserv.gs.t3"
           encoding="base64"
             compression="ZIP">
        ...
      </class>
      ...
    </description>
    <description content="text/plain"
                 referenced="false">
      <text>
      ...
      </text>
    </description>

  </algorithm>


  <parameters>
    <version> 305 </version>
    <valid>
      <from>  20000</from>
    </valid>

    <description content="binary"
                 referenced="false">
    ...
    </description>

    <description content="text/plain"
                 referenced="true">
        http://www.bt.com/...
    </description>
  </parameters>

  <reference>
    <sender> 162.154.20.41 </sender>
    <num> 12 </num>
  </reference>

  <authentication> ... </authentication>

</tariffMessage>
```

Figure 5: Tariff Message

4.3 Application Sublayer

Our implementation of the price communication protocol offers an easy-to-use interface and takes care of the complicated steps like requesting a tariff, extracting the code from the message, as well as installing and instantiating it.

5. Tariff Updates

As mentioned above, the protocol supports tariff updates. We now discuss shortly what happens when a completely new tariff shall be used for the first time. We also discuss how the parameters and/or the algorithm of an already existing tariff can be updated. Both points can cause some problems with the provider's CAS that have to be solved.

5.1 Installing a new tariff

When a new tariff is put into use for the first time, the CAS has to be informed of how the parameters of the session characterization of that tariff look like and how it can get this data from the mediation module.

An efficient way of doing that is to include some code with the tariff code in the tariff message; the code is installed and started inside the CAS and performs the operations mentioned above.

The protocol supports this by defining an extra tag that allows to mark classes to be for internal use only. They are not passed on to customers. Thus, a new tariff can be installed without having to stop, reconfigure and restart the CAS. This makes setting up new services a lot easier.

5.2 Updating a tariff

As we showed above, a tariff message can be optimized for size. This is intended for dynamic pricing scenarios in which tariffs might change very often, typically depending on the congestion level of the network. In these cases the parameters change often while the algorithm will probably remain constant over a long time and does not have to be included in every message.

5.3 Updates and Accounting

A problem occurs if a tariff is changed while a session is active. Does the session end and does a new session with the same parameters start immediately? If a tariff is non-linear, for example by offering a discount for session longer than 10 minutes, the charge for one long session is different to the total charge of two smaller sessions.

A provider can apply many different policies. If the session continues when a tariff is updated, he could charge by the last (or the first or the cheapest) tariff only. If a session stops when the tariff changes he can charge the old session by the old and the new session by the new tariff. But this can be seen unfair by the end-user if he looses the discount mentioned above.

If the accounting system knows exactly how the tariffs work, the policies mentioned above can be implemented by the accounting system. But for changing the policy or changing the tariff algorithm the CAS has to be reconfigured or even reprogrammed. As the data collected by the CAS is highly important for his businees, the ISP will normally try to avoid this situation. It can be avoided if the system is designed in a way that the accounting module only needs little insight into the inner workings of a tariff algorithm. The following solution allows the accounting module to have no deeper insight into the tariffs by adding extra functionality to the tariffs, which can - as shown above - easily be installed and updated without the CAS having to stop:

First, if a tariff changes while a session is active the CAS continues accounting that session, it justs stores additionally the status of the session at the moment of the tariff change. With this information the session can be split apart later by the tariff code if necessary.

In order to calculate the charge (after the session has finally ended) the accounting system passes the complete session characterization plus the old tariff(s) to the new tariff. The new tariff knows which of the policies above to use and how to take into account discounts and other non-linearities. If necessary, it will recursively call the older tariffs.

Note that we assume that the CAS keeps track of the old tariff algorithms and stores them as long as there are sessions active that started when that tariff was valid.

This way it is also very easy for a provider to change his policy without reconfiguring its CAS and pass this information on to the end-users. Additionally, its CAS needs no further insight into the tariffs which makes changes in tariffs a lot easier to realize.

6. Related Work

We now describe two other related protocols and compare them with the price communication protocol presented in this paper.

6.1 Open Settlement Protocol

The Open Settlement Protocol (OSP, [13]) is an ETSI TIPHON[3] specification that describes a set of protocols to permit the exchange of inter-domain pricing, authorization and settlement information between Internet telephony operators. The pricing part overlaps with the price communication protocol presented in this paper, yet, there are a lot of differences.

Both protocols are XML-based and both allow a binary format. While OSP uses HTTP only as underlying protocol, the price communication protocol allows different transport mechanisms.

OSP can be used to exchange *prices* (OSP terminology) - expressed in the terminology of this work, OSP is restricted to very simple tariffs with one price coefficient (like the per-minute prices for phone calls). As OSP is more intended as settlement protocol between providers, it offers an explicit mechanism to accept or decline prices (pricing confirmation). It offers no mechanism to request a special tariff or a tariff for a given service.

Summarizing, OSP offers broader capabilities but is far less powerful in the area of price communication and it is specialized for the IP telephony use case.

6.2 Internet Open Trading Protocol

The IETF Internet Open Trading Protocol (IOTP) specification [14] describes a framework for payment protocols and uses XML over HTTP. This work is complementary to the price communication protocol as it is mostly concerned with the settlement, while the price communication protocol determines how the individual charges the bill consists of are calculated.

7. Conclusions & Outlook

In this paper we presented a flexible protocol that allows to use a number of different transport mechanisms like UDP multicast, HTTP and RSVP to distribute tariffs between the ISPs management systems and to the customers. The protocol makes no special assumptions about the QoS architecture used (Intserv, Diffserv etc.).

To give ISPs freedom, tariffs can be distributed as Java code, thus every imaginable tariff can be realised. Dynamic pricing is feasible as the protocol supports a push mechanism and small-sized messages. Introducing a new tariff and updating existing ones cause problems in the charging and accounting system of an ISP. We discussed those problems and showed how they can be solved.

[3] European Telecommunication Standards Institute (ETSI) Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON)

Most parts of the protocol and the middleware using it are implemented and used within the M3I project. They will be available as open source after the end of the project.

We would like to thank the following people that contributed in the design process of the price communication protocol: Burkhard Stiller, Jan Gerke, Bob Briscoe, Clazien Wezeman, Huw Oliver, Hans Daanen and the rest of the M3I team.

8. References

[1] M3I Project, http://www.m3i.org/.

[2] **Braden R., Clark D., Shenker S.**, "Integrated Services in the Internet Architecture: an Overview", RFC 1633, June 1994.

[3] Partridge, C. and R. Guerin, "Specification of Guaranteed Quality of Service", RFC 2212, September 1997.

[4] S. Blake, D. L. Black, M. A. Carlson, E. Davies, Z. Wang, and W. Weiss, "RFC 2475 - An Architecture for Differentiated Services". Experimental RFC, December 1998.

[5] J. Heinanen, F. Baker, W. Weiss and J. Wroclawski, "RFC 2597 - Assured Forwarding PHB Group". Request for Comments, June 1999.

[6] Encyclopaedia Britannica, http://www.britannica.com/.

[7] Merriam Webster, http://www.m-w.com/.

[8] S. Herzog, "RFC 2750 - RSVP Extensions for Policy Control". Standards Track, January 2000.

[9] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. RFC 2205 - Resource ReSerVation Protocol (RSVP) – version 1 functional specification. Standards Track RFC, September 1997.

[10] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, "RFC 2068 - Hypertext Transfer Protocol -- HTTP/1.1". Standards Track, January 1997.

[11] Apache XML Project, http://xml.apache.org/.

[12] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, Dave Winer, "Simple Object Access Protocol (SOAP) 1.1", W3C Note 08, May 2000.

[13] European Telecommunications Standards Institute (ETSI), "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); Open Settlement Protocol (OSP) for Inter-Domain pricing, authorization, and usage exchange", ETSI TS 101 321 V2.1.1, August 2000.

[14] Burdett, D., "Internet Open Trading Protocol - IOTP Version 1.0", RFC 2801, April 2000.