IP Flow Accounting Application for Diameter

Racz, P; Stiller, B

Postprint available at:
http://www.zora.uzh.ch

Posted at the Zurich Open Repository and Archive, University of Zurich.
http://www.zora.uzh.ch

Originally published at:
IP Flow Accounting Application for Diameter

Abstract

Flow accounting in IP networks is used by network operators for various purposes, such as network management, traffic management, or traffic analysis. In order to integrate flow accounting into an Authentication, Authorization, and Accounting (AAA) infrastructure, this work designs and evaluates an accounting extension to the Diameter protocol - termed Diameter IP Flow Accounting (IPFA) application - in support of the efficient transfer of IP flow records. The new Diameter IPFA application has been implemented as a prototype and its evaluation shows that it achieves a better performance for the transfer of IP flow records than the traditional Diameter accounting approach.
IP Flow Accounting Application for Diameter

Peter Racz and Burkhard Stiller

Abstract—Flow accounting in IP networks is used by network operators for various purposes, such as network management, traffic management, or traffic analysis. In order to integrate flow accounting into an Authentication, Authorization, and Accounting (AAA) infrastructure, this work designs and evaluates an accounting extension to the Diameter protocol — termed Diameter IP Flow Accounting (IPFA) application — in support of the efficient transfer of IP flow records. The new Diameter IPFA application has been implemented as a prototype and its evaluation shows that it achieves a better performance for the transfer of IP flow records than the traditional Diameter accounting approach.

Index Terms—Flow accounting, diameter, IPFIX.

I. INTRODUCTION

IP flow accounting is highly relevant for network operators, since it can be used for network management, traffic management, or traffic analysis purposes. An IP flow is defined in general as a set of IP packets having the same properties with a fixed maximum inter-packet time. These properties can be determined by values carried in an IP packet, like IP header fields, transport header fields, or by other attributes, like incoming and outgoing interface. Thus, a flow can in general consist of all packets on a network interface but also just a single packet. A commonly applied flow definition uses the properties referred to as five-tuple, including the source and destination IP address, the transport protocol, the source and destination port. IP flow accounting collects accounting data, like number of bytes and packets, per IP flow.

The Diameter protocol [2] is used to provide Authentication, Authorization, and Accounting (AAA) services [8] mainly in network access scenarios and it does not support the transfer of IP flow records. Additionally, the Diameter accounting record format determines a quite high overhead for accounting attributes represented in a few bytes, like in the case of flow accounting, where typically a large number of flow records are generated, containing several short 1 to 4 byte long accounting attributes. Therefore, this work proposes a new extension to the Diameter protocol, called Diameter IP Flow Accounting (IPFA) application, in order to support an efficient transfer of IP flow records and to integrate seamlessly IP flow accounting into a Diameter-based accounting infrastructure. The IPFA application is developed as an alternative solution for IP flow record transfer for Diameter-based infrastructures beside existing flow accounting solutions.

The remainder of this paper is structured as follows. Section II discusses related work. While Section III outlines the architecture of the Diameter flow accounting application, presenting the network components, the newly defined record formats, and the Diameter protocol extension, details of the prototypical implementation are described in Section IV. Furthermore, evaluation results are discussed in detail in Section V. Finally, Section VI summarizes the work and draws conclusions.

II. RELATED WORK

For IP flow accounting several commercial accounting systems exist, such as NetFlow [4] from Cisco Systems or JFlow [7] from Juniper Networks. They support the collection of information on IP flows in routers and the transfer of flow data to a central server for storage and further analysis. The NetFlow protocol defines a flow data export format used to transfer IP flow records. NetFlow version 9 [5], also called flexible NetFlow, extends the set of attributes to be used to define an IP flow, including any IP header fields, transport protocol specific attributes, and routing specific attributes. The NetFlow record format version 9 introduces templates to represent data, that provide efficient record transfer and make records extensible without changing the record format. Templates define semantics of flow records and specify how to interpret values in a flow record. Flow records in turn carry only the values of those fields specified in the template, resulting in a more compact record format with less overhead.


Based on NetFlow version 9 the IETF has developed the IP Flow Information Export (IPFIX) architecture [14], providing a standard representation format [11] and a transport protocol [6] for flow accounting. In IPFIX a flow can be determined based on IP header fields, transport header fields, application header fields, characteristics of the packet itself (e.g., number of MPLS labels), and information derived from the packet treatment (e.g., output interface). The IPFIX protocol is used to transfer flow records between IPFIX devices. It supports flow record templates for compact record transfer.

The generic AAA (Authentication, Authorization, and Accounting) Architecture [8] specifies a common framework, forming the basis for user authentication, authorization, and accounting in a multi-domain environment. The architecture includes AAA servers and Service Equipments (SE), communicating via a standard AAA protocol. AAA protocols include
the Remote Authentication Dial In User Service (RADIUS) protocol [13] with its accounting extension [12] and the Diameter base protocol [2] that is considered as the successor of RADIUS. Compared to RADIUS, the Diameter protocol provides a reliable data transfer, fail-over mechanisms, better error handling, and security mechanisms. While the Diameter base protocol defines common functionality, like message format, message transport, capability negotiation, error handling, and security, different extensions — called Diameter applications — provide additional service-specific functionality, like the Network Access Server (NAS) application [3] supporting AAA in network access scenarios. Diameter is a flexible and extensible protocol, enabling the definition of new protocol messages and attributes. Similar to RADIUS, Diameter message attributes are represented by Attribute Value Pairs (AVP), supporting any kinds of attributes in a generic format. However, the AVP-based record format determines a quite high overhead in case of flow accounting, where attributes are typically 1 to 4 byte long, while the AVP header occupies 8 byte and additional padding can have a size of 1 to 3 byte.

Therefore, the Diameter IP Flow Accounting (IPFA) application developed closes a gap of functionality today by providing a new extension for the Diameter protocol in support of an efficient transfer of IP flow records. It provides an alternative flow accounting solution beside IPFIX. It supports a template-based flow record transfer and enables the integration of IP flow accounting into a Diameter-based accounting infrastructure. Additionally, it is fully compatible with the Diameter and IPFIX protocols.

III. IPFA ARCHITECTURE

The IPFA architecture defines network components of the Diameter flow accounting application, new Diameter record formats based on IPFIX, and the new Diameter protocol extension with new commands and AVPs.

A. Network Components

The network components of the Diameter IPFA application are shown in Fig. 1. The Diameter server serves as an accounting server and it includes the IPFA application. It collects IP flow records and stores them in a local database for further processing. Different network devices, e.g., routers, gather flow information by observing the traffic locally and send IP flow records to the server. The Diameter client represents a device that natively supports the Diameter IPFA application, while the IPFIX exporter represents a device supporting IPFIX-based flow record transfer. In order to integrate IPFIX exporters into the Diameter-based accounting system, the architecture includes the Diameter-IPFIX translation agent. This translation agent is capable to translate IPFIX flow records into Diameter IPFA records and it enables the communication between native IPFIX exporters and a Diameter server. Additionally, the translation agent can filter and aggregate flow records before forwarding them to the server.

The IPFIX exporter and the translation agent communicate via the IPFIX protocol and they exchange IPFIX records. The Diameter client and the translation agent communicate with the Diameter server via the Diameter IPFA protocol as specified below, exchanging flow records based on Diameter AVPs.

B. Record Formats

All record formats of the Diameter IPFA application are based on the IPFIX specification in order to provide compatibility. Three different records are defined, including the template record, the option template record, and the flow data record. The template record specifies the structure and semantics of a data record and it contains a sequence of type-length pairs, specifying all attributes of the data record. The option template record defines a template for a data record containing additional information, like the flow key, filtering information, or sampling parameters. The data record is used to transfer flow records and it contains attribute values according to its associated template record. For further details on the IPFIX record format and information model refer to [6], [11].

The Diameter IPFA application defines new AVPs for these three record types. The AVP format for the template record (cf. Fig. 2) consists out of the AVP header, containing the AVP code, flags, and AVP length fields, followed by the AVP value. The AVP value shows a structured format, specifying template records. An AVP can include several template records. The template record is identified by the Template ID field which is numbered from 256 for data records [6].
Fig. 4. Diameter option template record AVP.

Fig. 5. Diameter flow data record AVP.

specifies the number of fields in the template record. A sequence of Field Specifiers defines information elements (Fig. 3), specifying the attributes of a data record. An information element consists of the Information Element ID, specifying the attribute, and the Field Length, specifying the length of the attribute. The information element can be vendor-specific. In this case, the information element includes the optional Enterprise Number and the E bit is set to one. The set of information elements defined by IETF are specified in [11].

The AVP format for the option template record (cf. Fig. 4.) is similar to the template record AVP. The AVP value has a structured format and it contains one or more option template records. The Template ID includes the template identifier. The Field Count contains the number of fields in the option template record, including also the number of scope fields. The Scope Field Count specifies the number of scope fields. Both normal fields and scope fields have the same format as shown in Fig. 3. The AVP also contains an optional padding field.

The AVP format for the flow data record is shown in Fig. 5. The AVP value has a structured format and it contains one or more flow data records. The Template ID specifies the template to be used to interpret the flow data record. The template is transferred in the template record or in the option template record as specified above. The Length field specifies the length of the data set that includes one or more data records. The sequence of Field Values contains the attribute values of flow records. The length of a field value is not necessarily 32 bits as represented in Fig. 5, but they are encoded according to their data types specified in [11]. The AVP also contains an optional padding field.

C. Diameter Commands

Diameter accounting messages derived from [2] are used to transfer flow records specified by the Diameter IPFA application. The Diameter Accounting-Request message has been extended to support the transfer of the newly defined AVPs and it is shown in Fig. 6. The notation used in the specification is according to [2].

The Accounting-Request (ACR) command is used to transfer accounting data between Diameter peers. The command is extended with the IPFIX-Version-Number, the IPFIX-Export-Time, the IPFIX-Sequence-Number, the IPFIX-Source-ID,
### TABLE I

**DIAMETER FLOW ACCOUNTING AVPs.**

<table>
<thead>
<tr>
<th>AVP name</th>
<th>AVP type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accounting-Record-Type</td>
<td>Enumerated</td>
<td>It defines the type of the accounting record [2]. The Diameter IPFA application defines the new record type FLOW_RECORD, which is used to indicate flow accounting records.</td>
</tr>
<tr>
<td>IPFIX-Version-Number</td>
<td>Unsigned32</td>
<td>It contains the version field of the IPFIX packet that defines the version of the record format [6].</td>
</tr>
<tr>
<td>IPFIX-Export-Time</td>
<td>Unsigned32</td>
<td>It contains the time at which the IPFIX exporter sent the record as defined in [6].</td>
</tr>
<tr>
<td>IPFIX-Sequence-Number</td>
<td>Unsigned32</td>
<td>It is an incremental sequence number of data records as specified in [6].</td>
</tr>
<tr>
<td>IPFIX-Source-ID</td>
<td>Unsigned32</td>
<td>It contains the identifier of the IPFIX observation domain as specified in [6].</td>
</tr>
<tr>
<td>IPFIX-Template-Record</td>
<td>OctetString</td>
<td>It contains template records according to the format specified above.</td>
</tr>
<tr>
<td>IPFIX-Option-Template-Record</td>
<td>OctetString</td>
<td>It contains option template records according to the format specified above.</td>
</tr>
<tr>
<td>IPFIX-Data-Record</td>
<td>OctetString</td>
<td>It contains data records according to the format specified above.</td>
</tr>
</tbody>
</table>

### D. Diameter AVPs

New AVPs defined by the Diameter IPFA application are summarized in Table I. The Accounting-Record-Type AVP is extended with a new value specifying flow accounting records. The IPFIX-Version-Number, the IPFIX-Export-Time, the IPFIX-Sequence-Number, and the IPFIX-Source-ID AVPs carry IPFIX header fields and are used by the translation agent when converting IPFIX messages to Diameter messages. Between a Diameter client and server these AVPs are not used. The IPFIX-Template-Record, the IPFIX-Option-Template-Record, and the IPFIX-Data-Record AVPs include template, option template, and data records, respectively.

### IV. IMPLEMENTATION

The Diameter IPFA application has been implemented as a prototype in C++, providing all extended accounting messages and additional AVPs, as specified above. Fig. 8 shows the implementation architecture. The prototype is based on the Open Diameter library version 1.0.7-h [10] that provides the implementation of the Diameter base protocol. It supports Diameter message exchange, message routing, and the management of Diameter connections. The Diameter Client and the Diameter Server implement the functionality of the Diameter IPFA application and enable the exchange of flow records.

The Open Diameter API provides client and server session classes in order to implement application specific client and server functionality. The prototype uses the AAA_ServerAcctSession and AAA_ClientAcctSubSession classes to implement the flow accounting functionality and derives its session classes from them.

The Diameter Client generates pre-defined flow records and sends them to the Diameter Server. The Diameter Client implements the A4CFlowAcctClientSubSession and A4CFlowAcctRecordCollector classes. The A4CFlowAcctClientSubSession is inherited from the AAA_ClientAcctSubSession of the Open Diameter library and it implements the call-back functions specified by the Open Diameter API. The A4CFlowAcctRecordCollector is inherited from the AAA_ClientAcctRecCollector and it implements the flow record generation. The A4CFlowAcctClientSubSession template class is instantiated with the A4CFlowAcctRecordCollector class.

The Diameter Server receives flow records from the client and stores them in its database. The A4CFlowAcctServerSession is inherited from the AAA_ServerAcctSession and it receives incoming accounting requests. For every new accounting session the Open Diameter library creates a new instance of this class via the AAA_ServerSessionAllocator. The A4CFlowAcctRecordStorage is inherited from the AAA_ServerAcctRecStorage of the Open Diameter library and it parses incoming flow record and stores them in the database. The A4CDatabase represents the interface to the MySQL database of the server.

In order to be able to evaluate and compare the Diameter IPFA application with the basic Diameter NAS accounting,
the prototype also implements record transfer according to
the Diameter network access server application [3]. Thus, the
client can send flow records either based on record templates,
as specified by the Diameter IPFA application, or based on
separate AVPs for each attribute in the flow record according
to the Diameter accounting approach.

V. EVALUATION

The Diameter IPFA application provides functional en-
hancements to the Diameter protocol. It supports an efficient
IP flow record transfer by using record templates. These
templates enable a compact record transfer, since the semantic
meaning of accounting attributes is transferred only once,
while subsequent data records carry only the values of ac-
counting attributes specified. This ensures a record transfer
with low communication overhead. Additionally, the Diameter
IPFA application enables the transfer of several IP flow records
within a single Diameter message, while the classical Diam-
eter NAS accounting transfers every accounting record in a
separate Diameter message, resulting in higher communication
overhead. Finally, the Diameter IPFA application is compatible
with the IPFIX protocol. It specifies Diameter AVPs for IPFIX
header fields and a Diameter-IPFIX translation agent in order
to be able to translate and transfer IPFIX messages via the
Diameter protocol.

Furthermore, the Diameter IPFA application has been eval-
uated performance-wise based on its prototypical implementa-
tion. The respective test-bed consists of two nodes, a Diam-
eter Client and a Diameter Server. Both nodes are a Dell PowerEdge 850 with a Pentium 4 3.6 GHz CPU and
1 GB RAM, running Debian GNU/Linux 3.1 (sarge) with
kernel 2.6.16. They are interconnected via a gigabit Ethernet
switch. The Diameter Server uses a MySQL server version
4.0 as its database. Since the IPFA implementation has been
developed as a prototype and proof of concept, it has not been
optimized for performance. Furthermore, the Open Diameter
implementation utilized is still under general development,
which means that those results achieved cannot be applied to
derive performance characteristics for a real world network
deployment. However, the aim of the evaluation presented
here is to compare IPFA to the traditional NAS accounting
approach and to determine clearly the potential gain based on
the same implementation environment and the test-bed setup.

The performance evaluation includes performance measure-
ments with respect to the record transfer time and record
processing time and compares the new Diameter IPFA and
the traditional NAS accounting approach. In case of the
flow accounting approach, records are transmitted using the
record format specified above, while for the NAS account-

<table>
<thead>
<tr>
<th>Record type</th>
<th>Number of attributes</th>
<th>List of attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type 1</td>
<td>6</td>
<td>source/destination AS, number of bytes uplink/downlink, number of packets uplink/downlink</td>
</tr>
<tr>
<td>Type 2</td>
<td>9</td>
<td>source/destination IPv4 address, protocol, source/destination port, number of bytes uplink/downlink, number of packets uplink/downlink</td>
</tr>
<tr>
<td>Type 3</td>
<td>17</td>
<td>source/destination IPv4 address, protocol, source/destination port, source/destination TOS, source/destination interface, min/max packet size, min/max TTL, number of bytes uplink/downlink, number of packets uplink/downlink</td>
</tr>
</tbody>
</table>

Fig. 9. Record transfer time for the IPFA application.

These experiments varied the number of attributes in a flow
record and the number of flow records transmitted at the
same time both for the Diameter IPFA and NAS accounting
approach. A set of three different flow records have been
defined with different number of flow attributes, as shown in
Table II.

Additionally to the size of the flow record, the number of
flow records transmitted at the same time is a parameter in the
evaluation, since in case of flow accounting usually several
flow records are exported at the same time in a batch. In
the experiments the Diameter Client transmits 1, 25, 50, or
100 flow records at the same time. Note, that in case of the
Diameter IPFA application these flow records are transmitted
in a single Diameter message, while in case of the NAS
accounting in separate messages.

Finally, the set of experiments includes all possible com-
binations with the three different record types (cf. Table II),
the four different flow record numbers (i.e. 1, 25, 50, 100),
and the two different accounting approaches (i.e. flow and
NAS accounting). Results are averaged over 1000 runs. The Diameter Client exports every 5 seconds a set of flow records.

A. Record Transfer Time

The record transfer time is shown in Fig. 9 and Fig. 10 for
the three different record types in the function of the number
of records exported at the same time in case of the Diameter
IPFA application and the NAS application, respectively. This
time is measured on the Diameter Client from the time, when
the client starts creating a flow record, until it receives the

Authorized licensed use limited to: MAIN LIBRARY UNIVERSITY OF ZURICH. Downloaded on September 9, 2009 at 05:34 from IEEE Xplore. Restrictions apply.
In order to be able to better compare the IPFA and NAS accounting approaches, Fig. 11 shows the flow record transfer time of IPFA relative to the time of NAS accounting, i.e., $T_{IPFA}/T_{NAS}$. The Diameter IPFA application performs in general better than the traditional Diameter NAS accounting approach. For shorter records (i.e. record type 1) the performance difference is larger, while for longer flow records (i.e. record type 3) the difference decreases. In case there is only a single flow record transmitted at the same time, there is no significant difference between the flow and NAS accounting approaches, i.e. the diagram shows about 100%. However, when several flow records are exported at the same time, the new Diameter IPFA application performs 20% better for longer flow records and 40% better for smaller ones.

In case when a single flow record is exported at the same time, the performance difference is very small. However, when multiple records are exported at the same time, the processing time in case of IPFA is shorter, which is around 90% to 95% of that of the NAS accounting.

### B. Record Processing Time

Additionally to the record transfer time, the processing time of flow records on the Diameter Server has been analyzed. The processing time is measured on the server from the time, when the server receives an Accounting-Request message, until the server processes the request, stores the records in its database, and sends back an Accounting-Answer message. Note, that in case of the Diameter IPFA application a single Accounting-Request message can contain several flow records, while in the case of NAS accounting each record is transferred in a separate message. Therefore, in case of NAS accounting always the time required for a single record is measured and the time required for a set of records is derived from these measurements as an approximation by multiplying the result for a single record with the number of records exported at the same time. The processing time linearly increases with the number of records exported at the same time and it is larger for longer flow records (i.e. record type 3) for both the flow and NAS accounting approaches.

Fig. 12 compares the two approaches and shows the processing time of the Diameter IPFA application relative to the processing time of the NAS accounting, i.e. $T_{IPFA}/T_{NAS}$. In case when a single flow record is exported at the same time, the performance difference is very small. However, when multiple records are exported at the same time, the processing time in case of IPFA is shorter, which is around 90% to 95% of that of the NAS accounting.

### C. Discussion

The IPFA application determines an alternative solution for IP flow accounting and provides an efficient way of embedding IPFIX records into the Diameter protocol. For the deployment of the IPFA application the following possibilities exist:

1) In a pure IPFA-based deployment, flow exporters (e.g., routers) send flow records to the accounting server via the Diameter protocol using the IPFA record format. In this case there is no need for any translation agents, since all network components use the Diameter protocol. If there is already a Diameter server deployed in the network, this server needs a software update to support the IPFA application.
2) The IPFA application can also be deployed in coexistence with IPFIX. In this case translation agents are required that convert from Diameter to IPFIX protocol formats and vice versa. A common Diameter-based accounting server can be deployed in the network and IPFA-capable devices will send flow records to the server via the Diameter protocol, while IPFIX-capable devices will send flow records via the translation agent to the server. This configuration also allows for a seamless migration to IPFA.

If the IPFA application is deployed for high-speed routers, the flow record transfer performance is highly relevant. To avoid the loss of flow records, the capacity of the network as well as the performance of the Diameter client and server have to be dimensioned appropriately. That is, the network has to have sufficient capacity to transfer flow records, and Diameter client(s) do(es) not overload the server on average. Additionally, lossless flow accounting is guaranteed by applying the reliable transfer of the Transmission Control Protocol (TCP) or the Stream Control Transmission Protocol (SCTP) (the transport protocols selected for Diameter).

In case of high-speed networks, flow accounting can apply sampling to reduce the amount of data to be processed and transferred. The IPFA application supports the use of sampling, since it specifies the record format only, and any sort of sampling can be applied in turn in routers.

A Diameter server in a network is typically used for traditional AAA purposes according to the NAS application, where authentication and authorization time and the rate, at which authentication and authorization are performed, determine the performance of the server. Here, the goal is to reach a short response time, a user has to wait until access to the requested service is granted. If the IPFA application is deployed in addition to the NAS application, the operator will need a Diameter server with higher performance in general in order to maintain the short response time for the NAS application and to receive flow records via the IPFA application. However, the load can be distributed among several servers, and a distributed server infrastructure can be built in the following ways:

1) The IPFA and NAS application can run in parallel on the same server, while a single server will be responsible only for requests originating from a certain part of the network. This reduces the load on a single server.

2) The IPFA and NAS applications can be deployed on separate servers, so that separate servers are responsible for the flow accounting and for the user authentication and authorization. This can be achieved by the application-based message routing of the Diameter protocol. In that case IPFA messages would be routed to a different server than NAS messages.

Both approaches can guarantee a short response time for the user authentication and authorization, and both approaches process flow records at the same time.

Due to the prototypical nature of the current IPFA implementation as well as the Diameter implementation, those results of the performance evaluation addressed above indicate the feasibility of the approach developed. However, for the full performance of the IPFA application in a real world network deployment, future performance evaluations and the comparison of IPFA to a pure IPFIX flow record transfer shall be performed.

VI. SUMMARY AND CONCLUSIONS

The Diameter IPFA application developed and presented in this work extends the Diameter protocol to facilitate an efficient transfer of IP flow records. It specifies in full new record types supporting a template-based IP flow record format. The Diameter IPFA application is compatible with the IPFIX protocol, enabling interoperability with IPFIX-capable devices. Furthermore, it allows for a seamless integration of IP flow accounting into a Diameter-based accounting environment and makes the interconnection of IPFIX-capable devices to a Diameter-based accounting server possible.

All evaluation results show that the Diameter IPFA application performs in general better than the classical Diameter accounting approach for the transfer of IP flow records with respect to the overhead of accounting records, record transfer time, and local processing time on the Diameter server. This, in turn, enables an accounting server to serve a larger number of devices exporting IP flow records: Furthermore, it reduces the load on network components, which will be beneficial for the network operator applying IPFA within his network settings. Driven by the IPFIX compatibility a common usage for Diameter-based accounting set-ups and IPFIX-based applications become efficiently feasible.

VII. ACKNOWLEDGMENT

This paper was supported in part by the EU-funded NoE project EMANICS (No. 26854) on "The Management of Internet Technologies and Complex Services". Furthermore, the authors would like to thank the anonymous reviewers for the comments on improving the final version of this work.

REFERENCES

Peter Racz received his M. Sc. Degree in Computer Science from Budapest University of Technology and Economics in 2000 and his Doctoral Degree in Computer Science from University of Zürich in 2008. Since 2005 he is with the Communication Systems Group at University of Zürich. Previously, he worked as an engineer in the Department of Mobile Networks at Siemens AG, Germany (2000-2002) and was a research assistant at University of Federal Armed Forces, Munich, Germany (2003-2004). His areas of expertise include AAA architectures, accounting for IP-based services, QoS and mobility provisioning in IP networks, mobile telecommunication systems, and security architectures. He has been involved in several European and industry-funded research projects.

Prof. Dr. Burkhard Stiller chairs the Communication Systems Group CSG, Department of Informatics IFI at the University of Zürich UZH since 2004. He holds a Computer Science Diplom (1990) and a Ph.D. degree (1994) of the University of Karlsruhe, Germany. During his research locations of the Computer Laboratory, University of Cambridge, U.K. (1994/95), the Computer Engineering and Networks Laboratory, ETH Zürich, Switzerland (1995-2004), and the University of Federal Armed Forces, Munich, Germany (2002-2004) his main research interests cover, including current CSG topics, charging and accounting of Internet services, economic management, systems with a fully decentralized control (P2P), telecommunication economics, and biometric management systems.