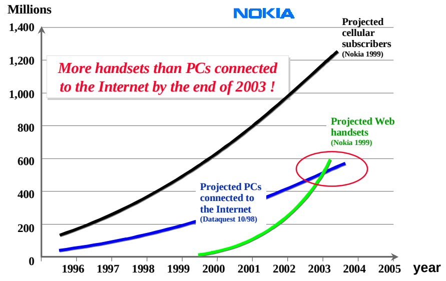
October 28-30, 2002, Dagstuhl Seminars “QoS in Networks and Distributed Systems“

Auditing and Charging in the A^x Architecture

Burkhard Stiller
Information Systems Laboratory IIS, University of Federal Armed Forces Munich
Werner-Heisenberg-Weg 39, D-85577 Neubiberg, Germany
and
Computer Engineering and Networks Laboratory TIK, ETH Zürich
Gloriastrasse 35, CH-8092 Zürich, Switzerland
stiller@informatik.unibw-muenchen.de or stiller@tik.ee.ethz.ch

– Introduction
– Scenario and Problems
– A^x Architecture: Auditing and Charging
– Conclusions

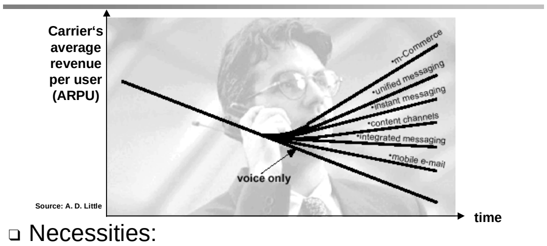
Mobility Growth – Subscribers, Handsets



More handsets than PCs connected to the Internet by the end of 2003 !

Be careful on any projected extrapolations!

Erosion and Opportunities

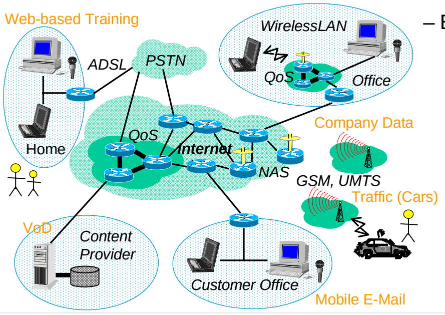


Source: A. D. Little

❑ Necessities:
– Careful definition of services: cost-sensitive.
– Service differentiation: reliability, QoS, pricing.
– Service support: mobility, secure access, auditing.

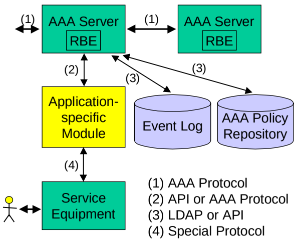
Scenario and Problems

Technology and Application Scenario



– Ensure:
- Mobility:
  – Privacy
  – User identity
- Charging:
  – Payment
  – Auditing
- To charge:
  – Access
  – QoS
  – Content

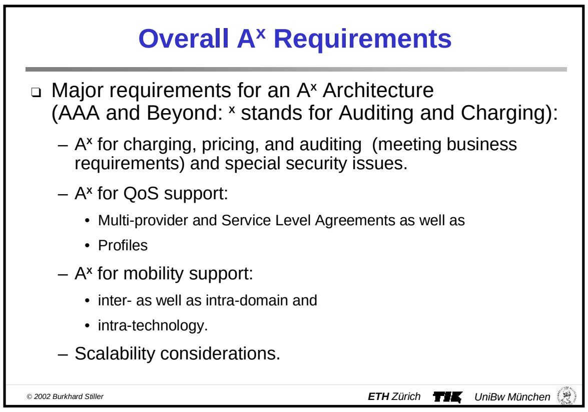
AAA Architecture and Weaknesses



(1) AAA Protocol
(2) API or AAA Protocol
(3) LDAP or API
(4) Special Protocol

AAA	Authentication, Authorization, Accounting
LDAP	Light-weight Directory Access Protocol
RBE	Rule-based Engine

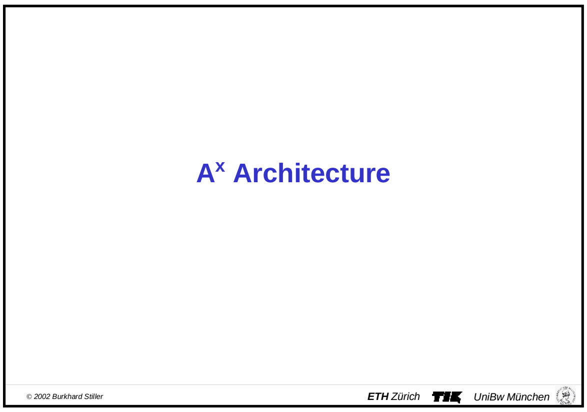
- Policy decision and policy enforcement not separated:
  – AAA Server decides on authorization, but enforces accounting.
- Difficult enhancements:
  – Enforcement located in the AAA server or the Application-specific module.
- AAA applied to transport, but not content, charging, and auditing.
- QoS support not provided.



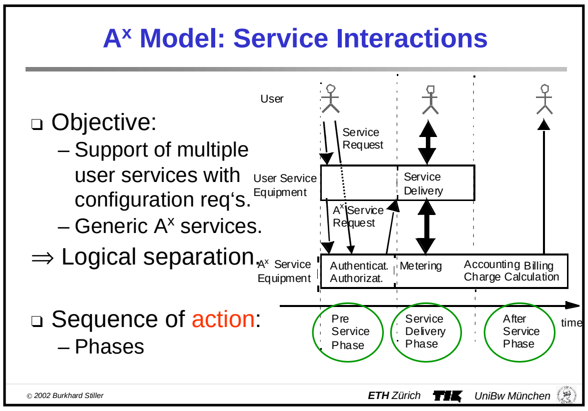
Overall A^x Requirements

❑ Major requirements for an A^x Architecture (AAA and Beyond: ^{x} stands for Auditing and Charging):

- A^x for charging, pricing, and auditing (meeting business requirements) and special security issues.
- A^x for QoS support:
  - Multi-provider and Service Level Agreements as well as
  - Profiles
- A^x for mobility support:
  - inter- as well as intra-domain and
  - intra-technology.
- Scalability considerations.



A^x Architecture



A^x Model: Service Interactions

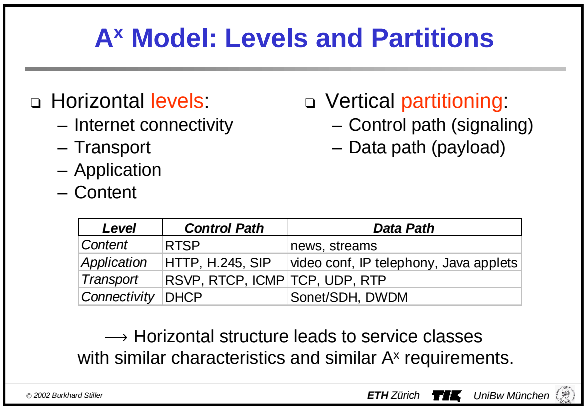
❑ Objective:

- Support of multiple user services with configuration req‘s.
- Generic A^x services.

⇒ Logical separation.

❑ Sequence of action:

- Phases



A^x Model: Levels and Partitions

❑ Horizontal levels:

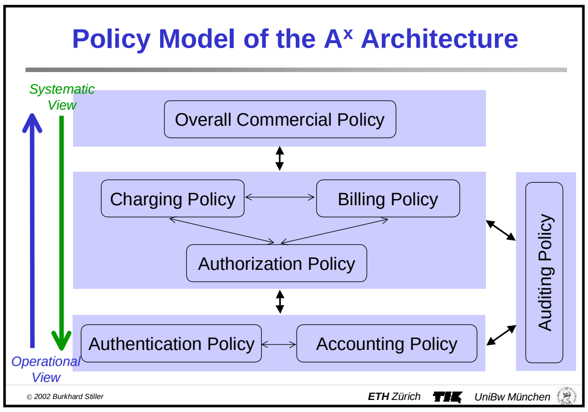
- Internet connectivity
- Transport
- Application
- Content

❑ Vertical partitioning:

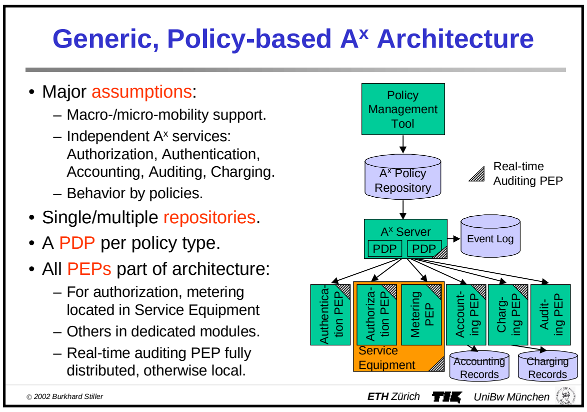
- Control path (signaling)
- Data path (payload)

Level	*Control Path*	*Data Path*
Content	RTSP	news, streams
Application	HTTP, H.245, SIP	video conf, IP telephony, Java applets
Transport	RSVP, RTCP, ICMP	TCP, UDP, RTP
Connectivity	DHCP	Sonet/SDH, DWDM

⟶ Horizontal structure leads to service classes with similar characteristics and similar A^x requirements.



Policy Model of the A^x Architecture



Generic, Policy-based A^x Architecture

- Major assumptions:
  - Macro-/micro-mobility support.
  - Independent A^x services: Authorization, Authentication, Accounting, Auditing, Charging.
  - Behavior by policies.
- Single/multiple repositories.
- A PDP per policy type.
- All PEPs part of architecture:
  - For authorization, metering located in Service Equipment
  - Others in dedicated modules.
  - Real-time auditing PEP fully distributed, otherwise local.

A^x Auditing

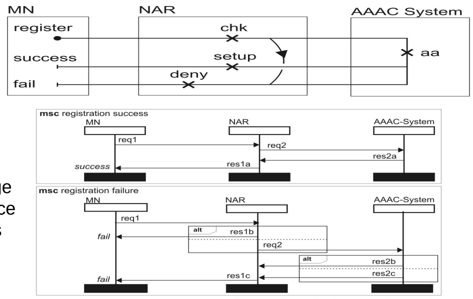
Definition — Auditing

- ❑ Auditing is the process of examining information on a provided service to check, whether the service has been provided correctly or the contractual negotiated parameters have been met.
- ❑ Logging of events and actions is based on information transmitted in messages between A^x entities.
- ❑ A^x support services:
  - – Provider – Provider: A^x, Service Compliance
  - – Provider – User: Mobile Network Access, A^x, Service Compliance
  - – Provider security: Attack, Misuse, Bugs

From Use Case Maps to Message Sequence Charts

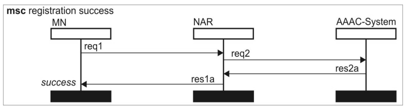
Use Case Maps

Message Sequence Charts



Note: Once a *fail* is reached, the message exchange is aborted

Messages — Example of an A^x Service



ID	Message	Parameters
req1	MN Authentication Request (MNARq)	NAI; Credentials
req2	Access Router Request (ARR.f)	Session Id; Host Information; User Information
res2a	Access Router Answer (ARA)	Result Code; Session Information
res1a	MN Access Response (MNARp)	Keys; Profile Sub-Set; Session Information

Message Formalization — Sample

```
NAR : instance
        in MNARq from MN
                MNARq : message
                Mobile Node Authentication Request, MN
                (
(10)                    <CAAP-Header: CHAP_CODE=2>
(11)                    {Challenge}
(12)                    {NAI}
(13)                    {MIPv6-Mobile-Node-Address}
(14)                    [MIPv6-Home-Agent-Address]
(15)                    [MIP-Binding-Update]*
(16)                    {MN-DH-PV}
(17)                    <MN-MAC>
(18)                    [AVP]*
                )
                endmessage
endinstance
```

Parameter References

Diameter AVPs or similar

Sample Log Entry

- ❑ Action:
  “AAAC.h has granted MN access and send an ARA message back to AAAC.f.”

```
event; logger; from; to;
time[hrs:ms:date]; session-id, result-
code, origin-host, session-timeout
```

… etc. …

Sample:
ara_sent; aaach::123; aaach::123; nar::121;
2000:0020:08162002; ses01, res2000,
fe80::201::fec:a072, 2010:0000:08162002

Main Auditing and Logging Policies

– P1:= A valid request should not be turned down.

– P2:= An invalid request should be rejected.

– P3:= The active entity, taking an action is responsible for logging this action, not the entity experiencing the event triggered.

– L6:= Whenever a log entry is made, the actual time the reported action took place must be logged.

Logging and Auditing Mechanisms

❑ Centralized main log:
  – mySQL or similar

❑ One Auditor per main task:
  – E.g., per process (Registration, Flow Termination)

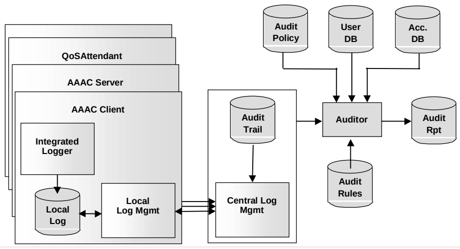
❑ Local DBs store individual log entries.

❑ Main log entries with embedded SQL code

❑ White-Box logging:
  – Different logging levels implemented in A^x entities
  – Dynamically control of logging levels

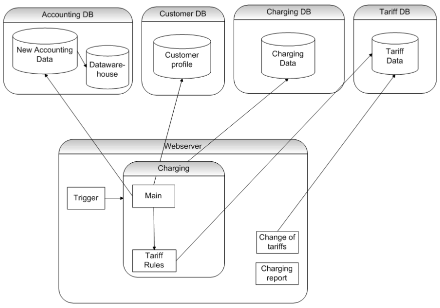
Auditing Framework

❑ Addressing service level guarantees and violation conditions.

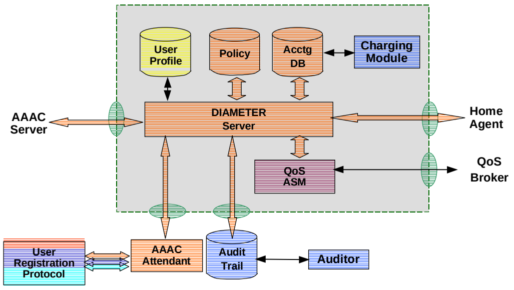


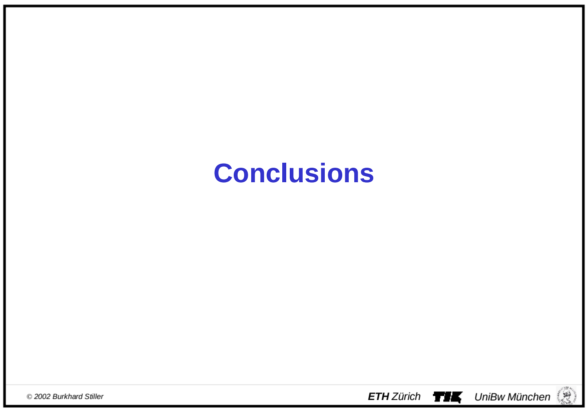
A^x Charging

Charging Databases

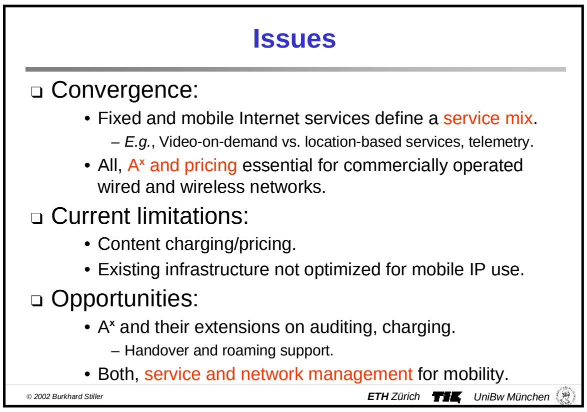


A^x Instantiation — IST MobyDick Project



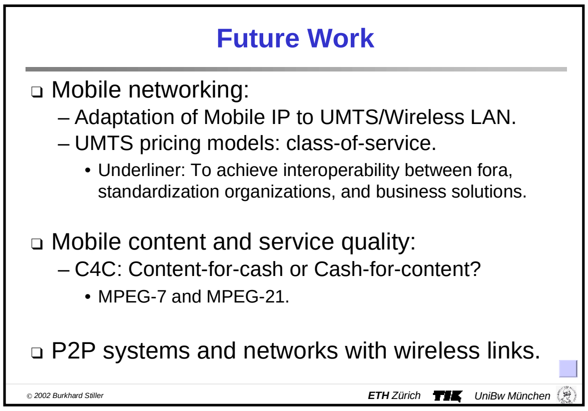


Conclusions



Issues

- ❑ Convergence:
  - Fixed and mobile Internet services define a service mix.
    - *E.g.*, Video-on-demand vs. location-based services, telemetry.
  - All, A^x and pricing essential for commercially operated wired and wireless networks.
- ❑ Current limitations:
  - Content charging/pricing.
  - Existing infrastructure not optimized for mobile IP use.
- ❑ Opportunities:
  - A^x and their extensions on auditing, charging.
    - Handover and roaming support.
  - Both, service and network management for mobility.



Future Work

- ❑ Mobile networking:
  - Adaptation of Mobile IP to UMTS/Wireless LAN.
  - UMTS pricing models: class-of-service.
    - Underliner: To achieve interoperability between fora, standardization organizations, and business solutions.
- ❑ Mobile content and service quality:
  - C4C: Content-for-cash or Cash-for-content?
    - MPEG-7 and MPEG-21.
- ❑ P2P systems and networks with wireless links.

Thank you for your attention.

Many thanks to J. Gerke, Hasan, D. Hausheer,
P. Kurtansky, J. Mischke, C. Rensing, M. Vinje and
all partners of the MobyDick project, in particular FhG FOKUS.