ICSY
Integrated Communication Systems

Architecture of the Future Internet

Paul Müller, Bernd Reuther, ICSY Lab, University of Kaiserslautern, G

What is ICSY

- The Integrated Communications Systems (ICSY) research group is aiming at the development of services to realize an integrated communication within heterogeneous environments. This is achieved by using service-oriented architectures, Grid technology, and communication middleware within a variety of application scenarios ranging from personal communication (multimedia) to ubiquitous computing.
- Main focus:
  - Service oriented network architecture
  - Lightweight SOA platform

Definition

- Generic definition of the term architecture:
  - The art and science of designing buildings and structures
- In computer science, architecture is the (ANSI/IEEE Std 1471-2000)[1]:
  - fundamental organization of a system
  - relationship of components (to each other and environment)
  - design and evolution principles
- Question for dynamic software systems?
  - how much system specific information (functionality, environment, usage, ...) must or should be considered by an architecture ?
    - some information must be considered
    - too much specific information might cause inflexible systems

[1] http://www.sei.cmu.edu/architecture/definitions.html

Architecture of the **current** Internet

- Fundamental organization
  - Layered Structure
  - One or more parallel protocols per layer
  - Functionality per layer is defined and fixed by a model (OSI or TCP/IP).
  - Location of functionality (end-system or network) motivated by the “end-to-end argument”
- Relationship of components
  - Each layer uses services of lower layers and offers another service to the upper layer
  - Interfaces between layers are not defined, only few common interfaces exist, most prominent:
    - The (Berkley) Socket Interface
    - NDIS (Network Device Interface Specification)
  - Interface between protocols of the same layer are not defined (IP ↔ ARP, IP ↔ Routing-Protocols)

Architecture of the **current** Internet

- Evolution principles
  - Overall
    - It should be possible to redesign a layer and its protocols without having to change the adjacent layers (OSI specification)
      - In IPv6 a new TCP implementation is needed
  - Per protocol
    - Options
    - Version numbers
    - Some bits for “future use”

Problems of the **current** Internet

- Low degree of flexibility
  - Short term: adaptivity and adaptability according to environmental conditions and user requirements
    - There are no (standardized) interfaces for negotiation of capabilities / requirements
    - Most (old) protocols are inflexible and thus hard to adapt (compared to new protocols like SCTP and DCCP)
  - Long term: enhance and exchange functionality
    - Exchange nearly impossible (e.g. IPv4 -> IPv6)
    - Enhancements in narrow bounds is possible
      - Must be backward compatible, which might be hard for complex protocols, e.g. all the TCP flow control enhancements
      - No common mechanisms for capability negotiation, each protocol must implement its own
        e.g. TCP Selective Acknowledgement

Problems of **current** Architecture

- Reasons
  - Architecture of the Internet is also a software architecture.
    It is well know since decades that "tight coupling" hinders maintainability and enhancements of software systems
    - coupling: the amount of information an element must have to use another element[2]
  - There is a lot of tight coupling in the current Internet architecture
    - Presupposition: existence of a specific protocol instance (e.g. TCP can communicate only with another TCP instance).
      In detail this means to presuppose:
      - a set of functionality (behavior)
      - formats of several data structures
    - Presupposition: the service of lower layers, without any negotiation
      - Can not adapt to low qualitative properties, e.g very low bandwidth
      - Can not utilize specific functionality, e.g. QoS / CoS
    - Lack of common interfaces hinders the exchange of (software) elements

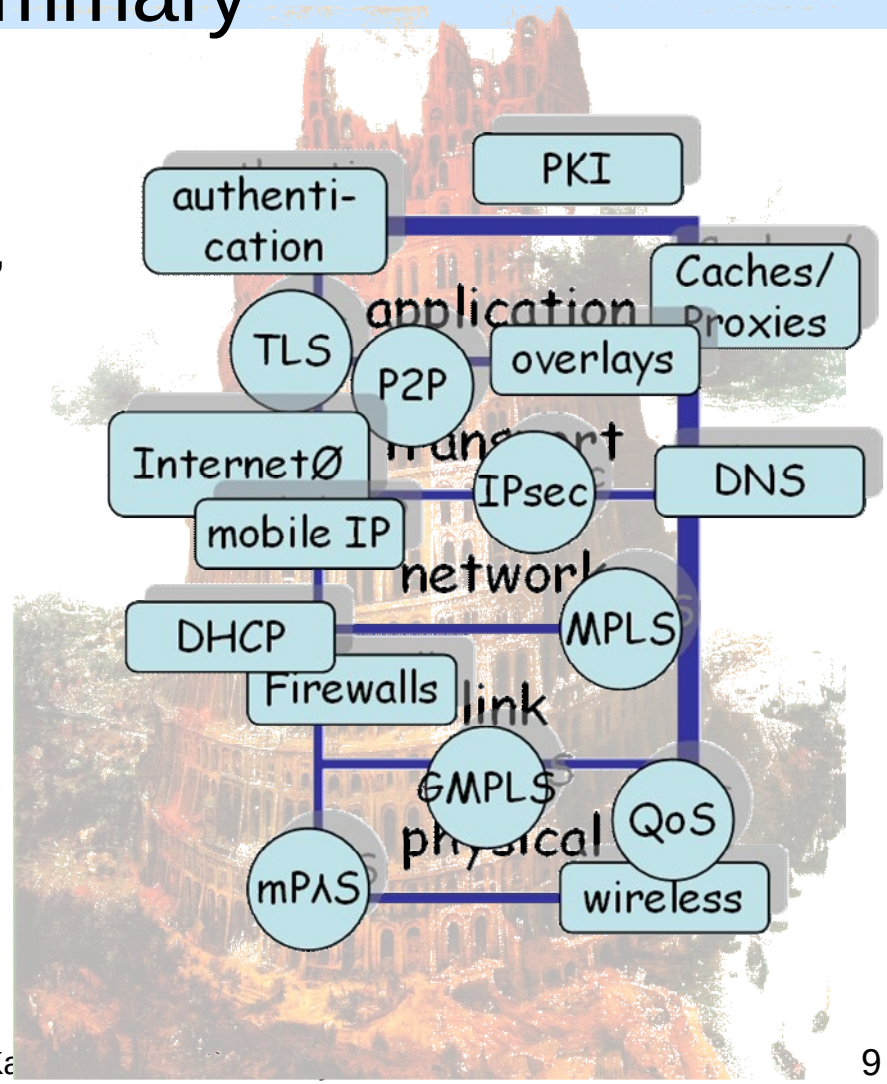
[2] Edward Yourdon and Larry L. Constantine
Structured Design: Fundamentals of a Discipline of Computer Program and System Design, Prentice-Hall, 1979

Problems of **current** Architecture

- Typical solution today:
  - Cross-layer: improve adaptivity / adaptability
    - Optimize several protocols according to some goal
      e.g. performance, manageability, use in wireless networks
    - Violate layered structure
  - Overlay networks: new mechanisms
    - Rebuild functionality of "lower layers“ at
      “higher layers”, e.g. routing
    - Enables (new) functionality for few applications only
      - Many overlay networks already exist
      - Does this scale ?

Architecture of the **current** Internet summary

- Original architecture is violated
  - Middleboxes (NAT, caches/proxies,…)
  - Intermediate layers (TLS, IPsec, MPLS, ...)
  - Specialized network domains (areas with specific QoS or security properties)
- Hinders innovation
  - Hard to integrate new mechanisms
  - QoS / CoS
  - Mobility
  - Security / Authentication
- Complexity is still rising …



Service Approach for the Future Internet

- Basic Idea:
  - A communication system made of loosely coupled services loosely coupled → avoid implicit premises as much as possible
  - Apply SOA principles to communication systems (requires new techniques)
- Explicitly refer to required/offered functionality and data structures
  - Enables change of functionality and data structures and thus provides higher degree of flexibility
- Define explicit interfaces and interaction between elements of the architecture
  - Dependencies to each other

Service Approach for the Future Internet

- Avoid complex protocols
  - There is no need to bundle functionality that might be used independent of each other
  - Protocol decomposition to micro protocol is not new, e.g.
    - Dynamic Network Architecture (O'Malley & Perterson)
    - Dynamic Configuration of Light-Weight Protocols (Plagemann, Plattner, Vogt, Walter)
    - Componentized Transport Protocols (Condie et al.)
    - …
- The service approach is more general
  - Replacing implicit assumptions by explicit references does not reduce functionality

Service Approach for the Future Internet

- Avoid to presuppose where some functionality is placed (end-system, network or network domain)
  - The end-to-end argument postulates that some functionality can only be implemented in end systems.
    But is the location of a functionality a principle that never changes?
    - Saltzer, Reed, and Clark mention an alternative to end-to-end implementation: The goal would be to reduce the probability of each of the individual threats to an acceptably small value.
      This was considered to be too uneconomical (1984) → is this true today and in future ?
    - Moors[3] argues that the end-to-end argument is mainly derived from trust and not from technical issues → what is acceptable depends on requirements!
    - Typically reliability should be provided end-to-end, but interceptions of TCP connections by proxies are reality today.
      Who cares about the reduced reliability?
  - The architecture should not presuppose where some functionality is located, because this may change (but an application may do so).
  - In consequence: a layered structure is no longer appropriate

[3] Tim Moors, A Critical Review of End-to-end Arguments in System Design, Proc. 2002 IEEE International Conference on Communications, pp. 1214-1219, April 2002

Advantage

- For users
  - Adaptivity / Adaptability to environment
    - optimized performance
  - Adaptivity / Adaptability to requirements
    - optimized qualitative properties (i.e. QoS)
  - Request services instead of mechanisms
    - Easy to use, because much less technical know-how required
  - Extendable set of mechanisms
    - Large toolbox of services available
- For providers
  - Extendable set of mechanisms
    - Add functionality needed locally (e.g. for traffic engineering, accounting, management, …)
    - Easy to deploy new services
  - Reduced dependencies between mechanisms
    - Improved robustness

Architectural Issue: Flexibility

- The future internet should fulfill a lot of different requirements and should be applicable in a lot of different environments
- We claim: in the long run no fixed set of mechanisms/protocols will fulfill all requirements and are appropriate in all environments
- Consequence: the future internet must be flexible according to the mechanisms used
  - Short-Term: adapt to current requirements and environment
  - Long-Term: evolve with ongoing technological developments
- Generic concepts for flexible handling of mechanisms (i.e. “how to put things together”) are a major challenge for the design of the future Internet architecture

Architectural Issue: Scalability

- The future Internet should be accessible for everybody, everywhere, every time and at every scale
- From this follow scalability demands according to
  - Dimension
    - Efficient and easy to use in small networks
    - Suitable for world-wide networks with many hosts
    - Ubiquitous computing / sensor networks (InternetØ)
  - Capabilities of links
    - Low and high capacity links
    - Different characteristics of error rates and delay
  - Capabilities/resources of nodes
    - Efficient in miniature devices with few resources as well as for large HPC systems

Architectural Issue: Application neutrality

- The future Internet should support all kinds of applications
  - Do not presuppose who will use the network and how
  - Note: the current Internet was originally developed for data exchange between computers only
- Applications must be independent of communication mechanisms
  - Make all mechanisms used for communication transparent for applications (loose coupling between application and communication system)
  - For example, today a common application using UDP can not utilize DCCP instead without re-writing some code of the application. Such dependencies hinder the spreading of new improved mechanisms/protocols

Examples of Services 1

- Error handling
  - Error detection
    - CRC, Hash, ...
  - Error notification
  - Error recovery Retransmission or FEC
- Identifier
  - Identify an endpoint
  - Identify a location
  - Identify a process
  - Identify a user
- Multiplexing
- Fragmentation/Assembly
  - Fragmentation
  - Segmentation
  - Blocking
- Connection / flow setup
  - Implicit
  - 3-way handshake (e.g. TCP)
  - 4-way handshake (e.g. SCTP)
  - Dynamic label distribution (e.g. MPLS/GMPLS)
  - Halfclose
- Address resolution
  - ARP, DNS
- Routing / forwarding
  - Use local routing tables, DHT
- Flow Control
  - with respect to destination
  - Congestion control, i.e. with respect to network
  - Rate control
- QoS / CoS
  - Classes & Aggregation
    - DiffServ
  - Signaling
    - RSVP
- Path management
  - Path-switching
  - Path monitoring
    - keep-alive, heartbeat
  - MTU Discovery
- Multicast
- AAA
  - Authorization
  - Accounting
- Encryption
  - Key Exchange
    - Diffie-Hellman, RSA
  - Cipher-Algorithm
    - DES, 3DES, AES
- Real Time support
  - Content identification
  - Source identification
  - Start / Stop marker
  - Time + Sequence number
- Communication patterns
  - Request / Reply
  - Message Passing
  - Message Queuing
  - Publish / Subscribe
  - Media Streaming
  - File transfer

1. *This list is not intended to be complete*
2. *The protocols mentioned are not services by themselves, they are only examples for mechanisms*

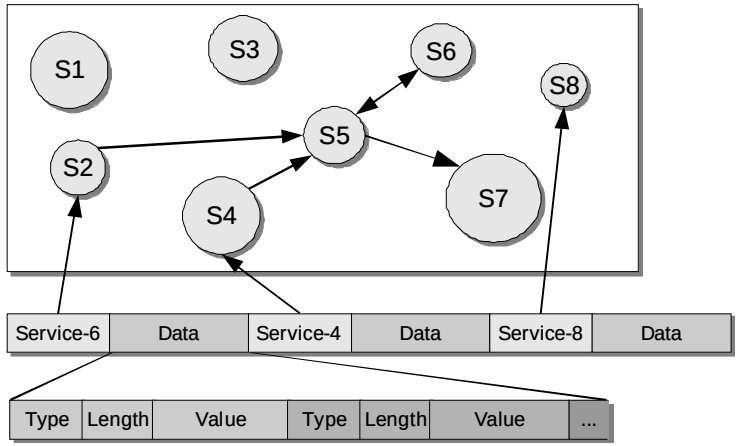
Integrated Communication Systems

Examples of Services 2

- Loop detection and elimination
  - STP, MSTP, RSTP
- Trunking
- Virtual path / tunneling
  - MPLS/GMPLS
- VLAN tagging
- Load balancing
- Routing / determine topology
  - IS-IS, OSPF, IGRP, RIP
  - BGP
- Authentication
  - 802.1x, Radius, TACACS, Kerberos
- Monitor infrastructure
  - Load
  - Error rates
  - Signal strength (wireless)
  - Availability
- Traffic engineering
- Network Management
  - Get / Set (e.g. SNMP)
  - XML based (e.g. netconf)
  - Provide configuration data
    - DHCP, TFTP
- Capability negotiation
- Network admission control
  - by user
  - by device / device configuration
- Network protection
  - Firewalls
  - Intrusion Detection
- Resilience
  - Path/Node failure
- Self-organization and self-management techniques

1. *This list is not intended to be complete*
2. *The protocols mentioned are not services by themselves, they are only examples for mechanisms*

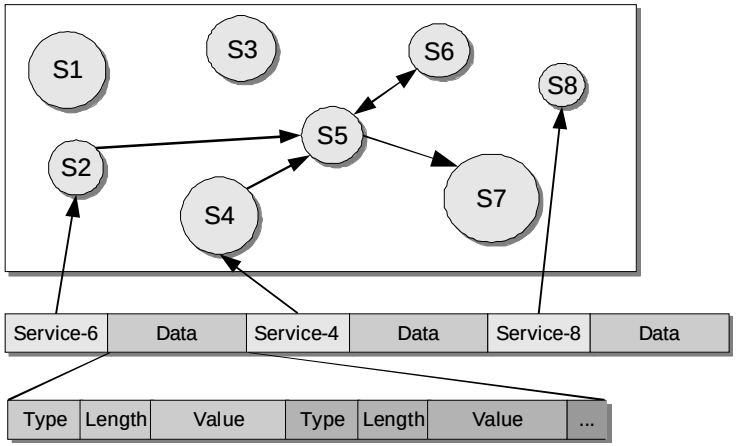
Illustration of Services Concept 1



- Metadata increase flexibility
  - Explicit references to services
    - Simplifies provision of new services
  - Explicit descriptions of data types
    - Simplifies extensions of mechanisms (add optional data)
    - Enables alternative mechanisms (add alternative data)
  - Similar to “role based architecture” approach[4] but:
    - Roles can be exchanged/replaced, but it is not possible to extend roles (e.g. add optional data)
    - Separates application payload from protocol header, i.e. one role can not contain other sub-roles.

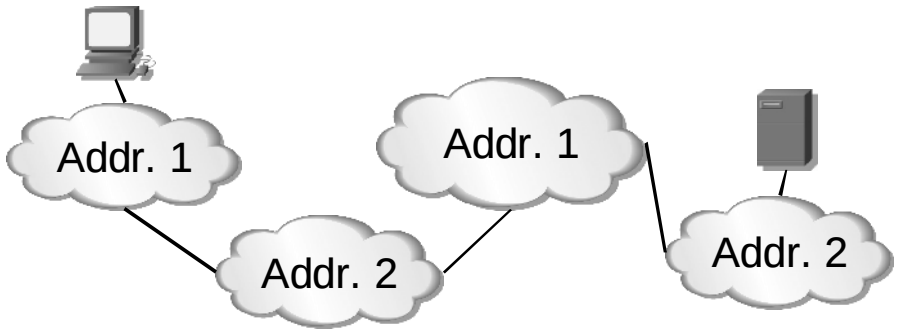
[4] R. Braden, T. Faber, M. Handley, "From protocol stack to protocol heap: role-based architecture", ACM SIGCOMM Computer Communication Review archive Volume 33, Issue 1, January 2003

Illustration of Services Concept 2



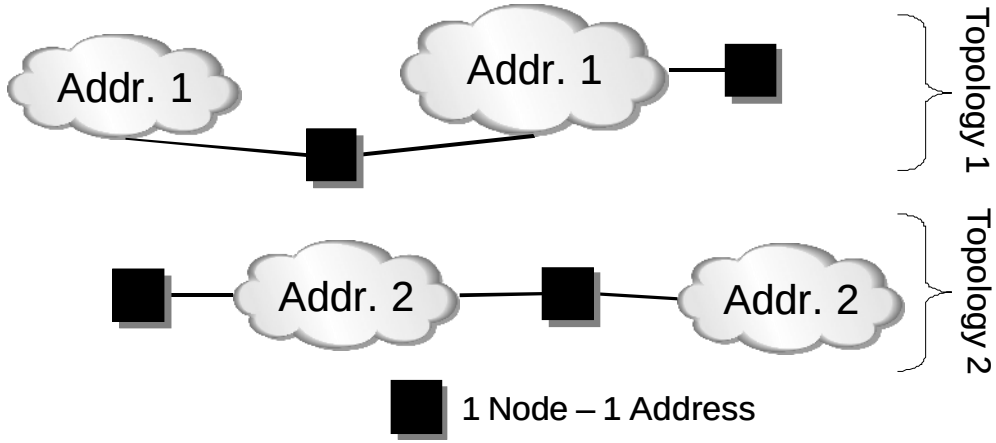
- Dependencies are defined locally and by requestor
- Some services may run in parallel (S2, S4) or in Sequence (S2, S5, S7)
- A service may contain another service
- Services may be stateful (e.g. S2 may map a flow label to an internal flow identifier and/or provide state information for other services)
- Services may negotiate with each other (S5, S6) for example to determine available resources
- Some services have interfaces to applications and/or hardware

Illustration of Services Concept 3



- Problem: usage of different addresses
  - May be necessary during migration
- Solution today: tunnels
  - Requires support of network infrastructure

Illustration of Services Concept 4



- A network domain using another type of address is just a black box
- Two addresses enable end-to-end connectivity without explicitly defined tunnels

Summary

- Tight coupling in the current Internet hinders adaptivity as well as evolution
- Goal: an architecture of loosely coupled elements (services) suitable for a future internet

Convergence

Should not be based on protocols

but on

Architecture