



Future networks – Challenges and opportunities

Towards a Future inter-networking
Architecture

Eurescom Study Programme

Heidelberg, 10/11 December 2007





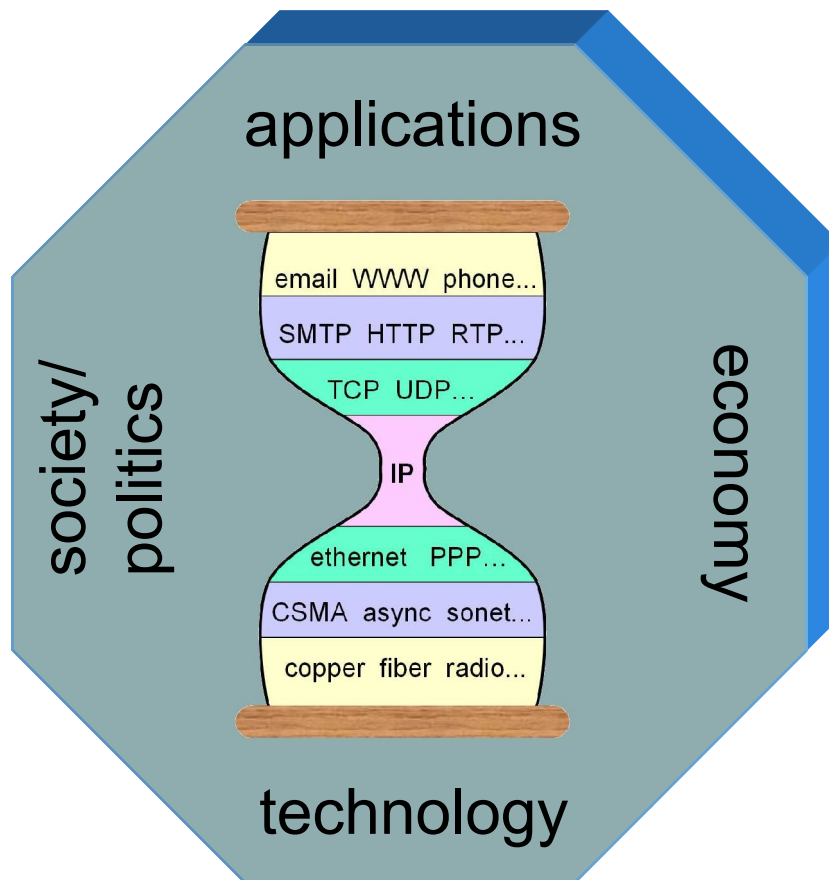
Content

- What we are talking about
- Related Work
- Service Approach for the Future Internet
- Conclusion

1.

What we are talking about

What we are talking about?



- I do not talk about:
 - Applications
 - Technology
 - Economy
 - Society/Politics
- But: Architecture and how it can evolve a new “inter-networking” Paradigm
 - At least as good as today's solution
 - Replace the traditional protocol layering paradigm with a more general model



Architecture

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



Architecture of the current Internet

- Design Goals*:
 - Goal is connectivity
 - Tool is the Internet Protocol
 - Intelligence is end to end
- Design Principles:
 - Layering,
 - Packet switching,
 - A network of collaborating networks,
 - Intelligent end-systems as well as the
 - End-to-end argument

* RFC 1958 „Architectural Principles of the Internet, B. Carpenter, Editor, IAB, June 1996

Architecture of the current Internet

- Fundamental organization
 - Layered Structure
 - One or more parallel protocols per layer
 - Functionality per layer is defined and fixed (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 (Berkeley) Socket Interface
 - NDIS (Network Device Interface Specification)
- Design and evolution principles
 - Overall
 - It should be possible to redesign a layer and its protocols without having to change the adjacent layers
 - But IPv6 requires a new TCP implementation
 - Per layer
 - Use only services of lower layers, i.e. mechanisms of lower layers are transparent
 - But TCP/UDP include IP-pseudo-header in CRC
 - Per protocol
 - Options
 - Version numbers
 - Some bits for “future use”



Basic assumptions ... but

- The end-to-end principle
 - Do not trust the network
→ implement several functionalities in endsystems
 - There are many miss behaving / malicious endsystems (users)
→ today the network can not trust the endsystems
- Layered Architecture
 - Transparency of lower layers → simplifies design
 - Today we have cross layer design – security balconies- ...
- Keep the network simple
 - Stateless IP in the core → robustness
 - today we have additional mechanisms for QoS / MPLS ...
- Convergence is based on IP
 - Widespread and mature technology
→ available within many devices (from PDA to HPC)
 - Widespread technology with low potential for evolution
→ mobility, sensor networks, ...



Problems of the current Internet

<http://www.icsy.de>

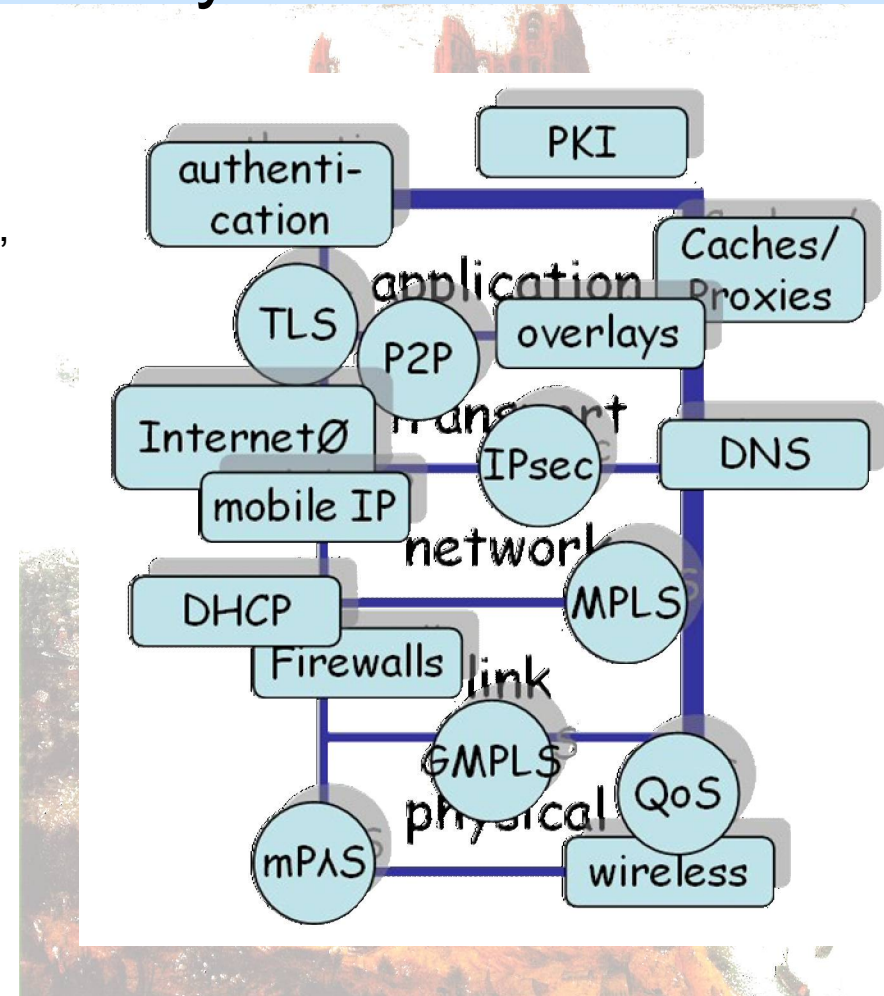
- Low degree of flexibility
 - Short term:
adaptivity and adaptability according to environmental conditions and user requirements
 - no negotiation of capabilities
 - Inflexible protocols
 - Long term:
enhance and exchange functionality
 - Exchange nearly impossible (e.g. IPv4 -> IPv6)
 - Enhancements in narrow bounds is possible
- Typical solution today:
 - **Cross-layer** optimization:
improve adaptivity / adaptability
 - Optimize several protocols
 - **Violates layered structure**
 - Overlay networks:
new mechanisms
 - Rebuild functionality of "lower layers" at "higher layers"
 - Enables (new) functionality for few applications only
 - How many overlays will be required?

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)

- Increasing interdependencies hinder innovation
 - Hard to integrate new mechanisms
 - QoS / CoS
 - Mobility
 - Security / Authentication

- Complexity is still rising ...



2.

Related Work



Programs

- Germany
 - **IKT 2020**
- EU 6/7 Frameworkprogram
 - **FP7** Challenge 1: Pervasive and Trusted Network and Service Infrastructures Objective 1.1: The Network of the Future
- USA
 - New Arch Project (30.6.2000 – 31.12.2003; DARPA)
 - Future Generation Internet Architecture
 - **FIND/GENI** NSF projects
 - Clean Slate Project (Stanford)
- Korea
 - **u-IT839** : Future of the Internet for Korea
- Japan
 - **CORE**: Collaborative Overlay Research Environment



Related Work

- The most interesting clean slate approaches are
 - **role-based approach (RBA)** conducted in the NewArch project;
 - and the **SILO approach** introduced by Dutta et. al.;
 - and more general the **Clean-Slate project at Stanford**.
- The **RBA** represents a non-layered architecture organizing communication in functional units referred to as “roles”.
 - Roles are not hierarchically organized, and thus may interact in many different ways.
 - The main motivation for RBA was to address the **frequent layer violations** that occur in the current Internet architecture
- The **SILO** approach also introduces a non-layered design
 - It is based on silos of services assembled on demand and specific to an application and network environment.
 - The overall goal of the SILO architecture is to facilitate “**cross-layer**” interactions

3.

Service Approach for the Future Internet



The Internet a complex SWS

- The In Physical connectivity: Links
- Point-to-point connectivity: NIC, switches
 - **Distributed** hardware, protocols - local management
- End-to-end connectivity: Routers
 - Forwarding, addressing, routing
 - **Distributed** hardware, protocols, software, management by Internet Service Providers (ISPs)
- Process-to-process connectivity: TCP, UDP
 - De-/multiplexing, reliability, congestion control, ...
- Applications: Web, P2P, ...
 - Users
 - **Distributed**, independent, autonomous, ...



The Internet a complex SWS

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



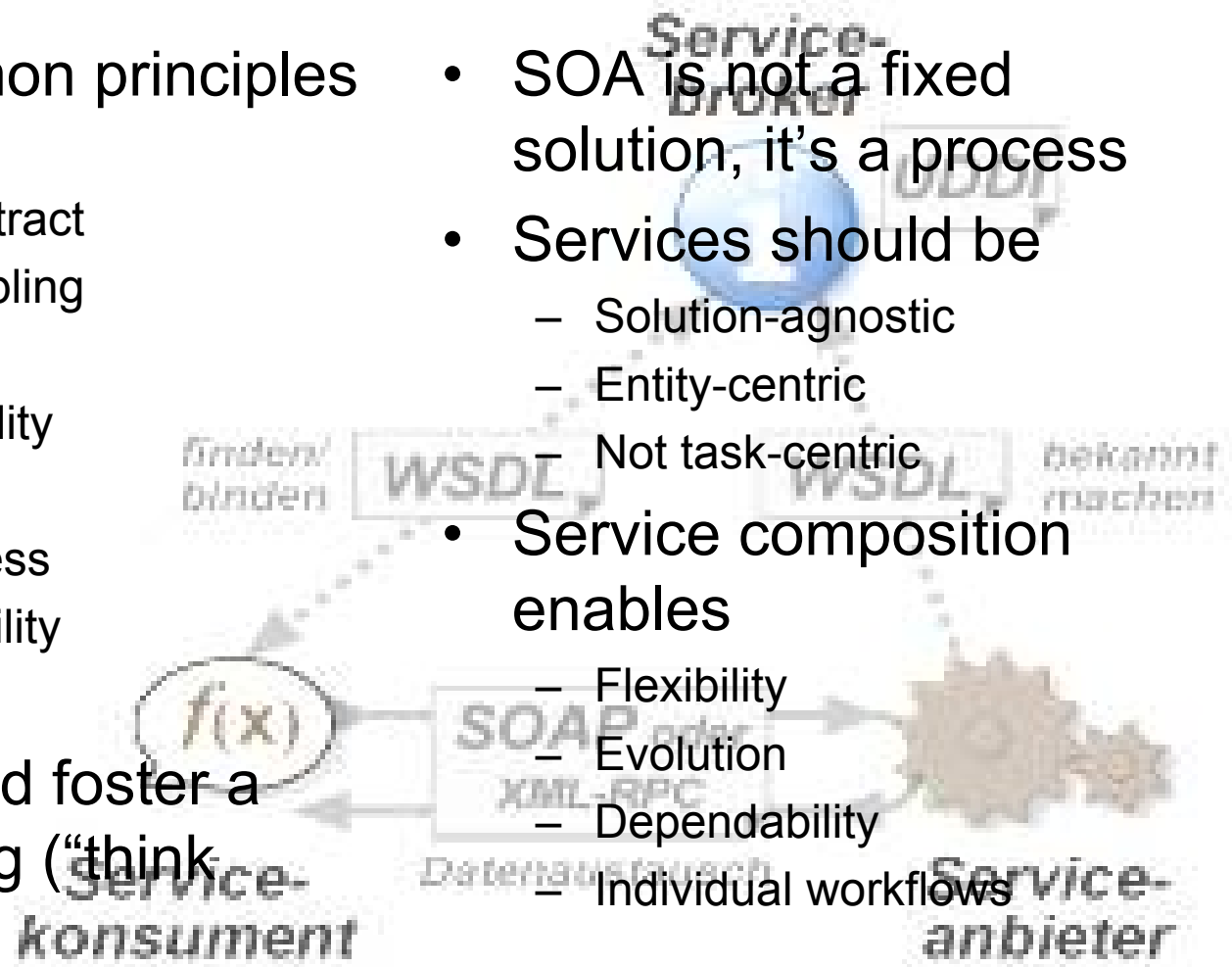
The term “Service”

- A unit of work done by a service provider to achieve desired end results for a service consumer.
 - The results of a service are usually the change of state for the consumer and/or the provider
- Benefit: Loose coupling between the participating software agents, enabled by:
 - A small set of **simple and ubiquitous interfaces**.
 - **Descriptive** messages constrained by an extensible schema delivered through the interfaces. None, or only minimal, system behavior is prescribed by messages.
 - **Extensibility**
 - **Service discovery**
- Focus on exposing business functions, not technology



Lessons learned from SoA

- Eight common principles
 - Reusability
 - Formal contract
 - Loose Coupling
 - Abstraction
 - Composability
 - Autonomy
 - Statelessness
 - Discoverability
- Demand and foster a new thinking (“think SOA”)
- SOA is not a fixed solution, it’s a process
- Services should be
 - Solution-agnostic
 - Entity-centric
 - Not task-centric
- Service composition enables
 - Flexibility
 - Evolution
 - Dependability
 - Individual workflows



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
 - Authentication
 - **802.1x**, Radius, TACACS, Kerberos
 - 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*

Used by / required
for TCP/IP



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
- **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*

Service Approach for the Future Internet

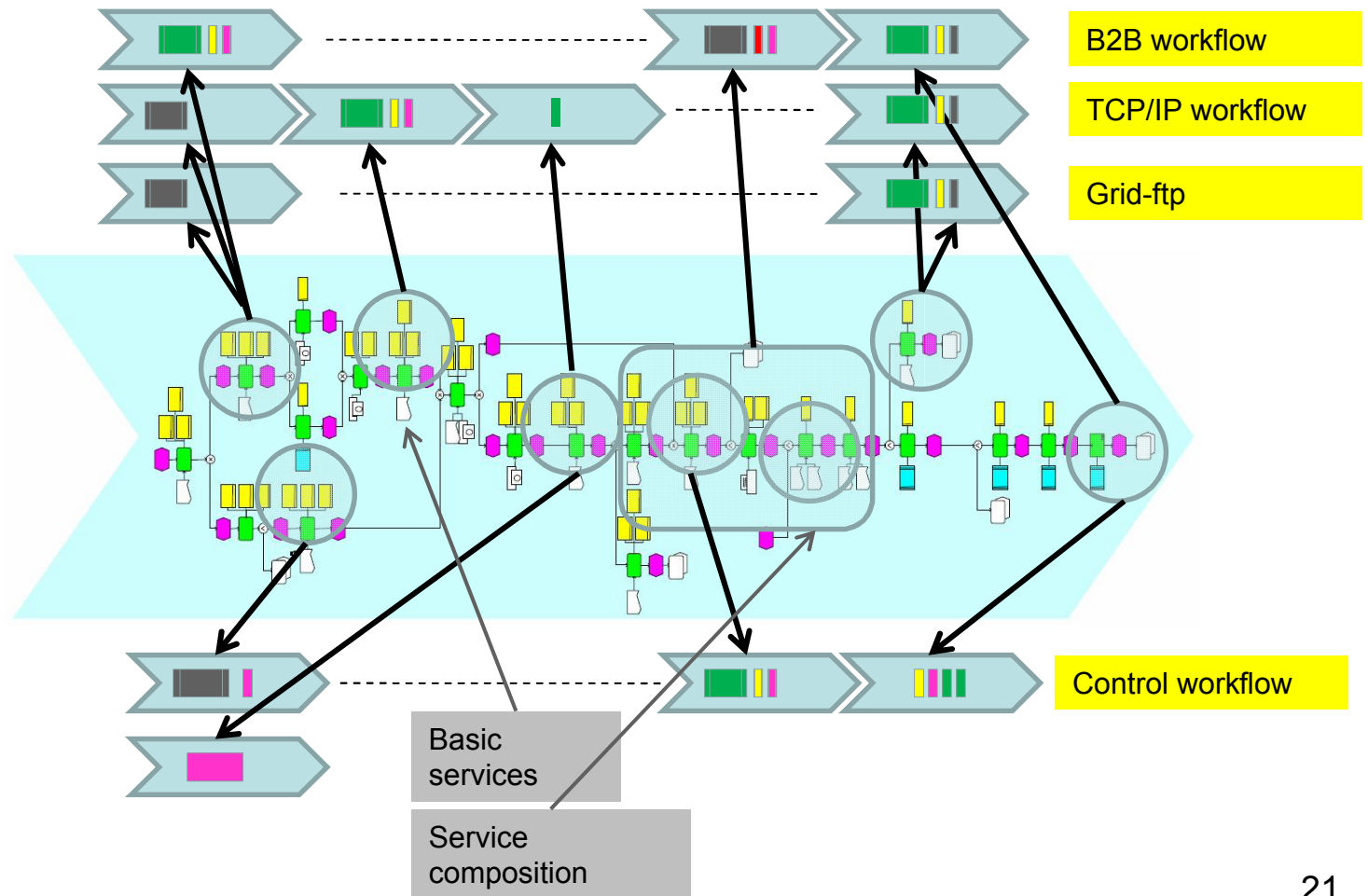
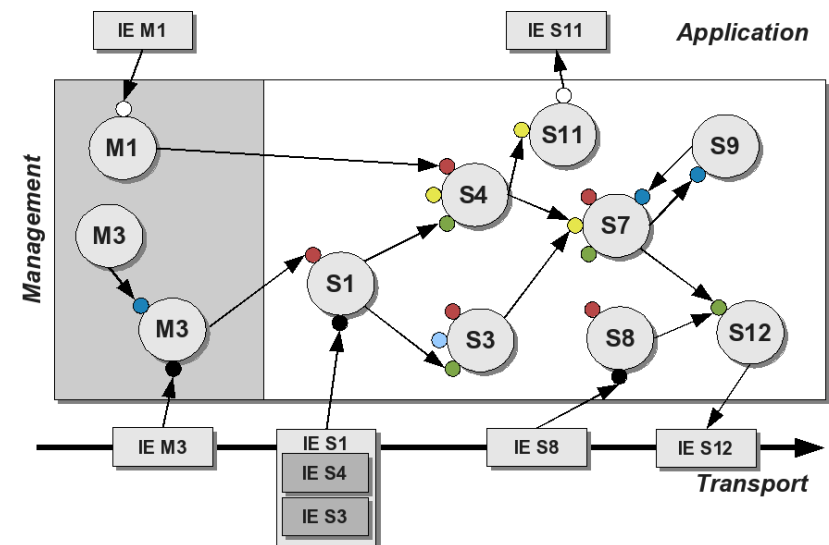


Illustration of Services Concept

- 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)
- Services have well defined interfaces
 - Enable exchange of service implementations
- Services are fine granular
 - Similar to micro protocols
- Similar to “role based architecture” approach 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.



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 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



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



Layered vs. Service-Oriented

- Traditional layered architecture
 - Modularity
 - Several functionalities per protocol unit
 - One header per protocol with interrelated data
 - Processing order:
 - Sequential per layer
 - Explicit/Implicit:
 - Explicitly name protocol
 - Implicitly associate data structure and processing rules
 - Explicitly add options
- Service-oriented architecture
 - Modularity
 - Functional spec of a communication building block.
 - One header (metadata) per functionality
 - Processing order:
 - Arbitrary graphs
 - Explicit/Implicit:
 - Explicitly name functionality and data structure
 - Explicitly add optional data

4.

Conclusion and Summary



Advantage/Impact on:

- 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
 - Ease of access to relevant information
 - Easy to introduce **new applications** and **services** with new features
 - Security, mobility, quality of service



Advantage/Impact on:

- Economy, Society, Politics:
 - New interfaces between providers (network/service)
 - New value-chain and new roles for providers
 - Open interfaces may enable new ecosystems of business Alliances
 - Information society
 - ...



Advantage/Impact on:

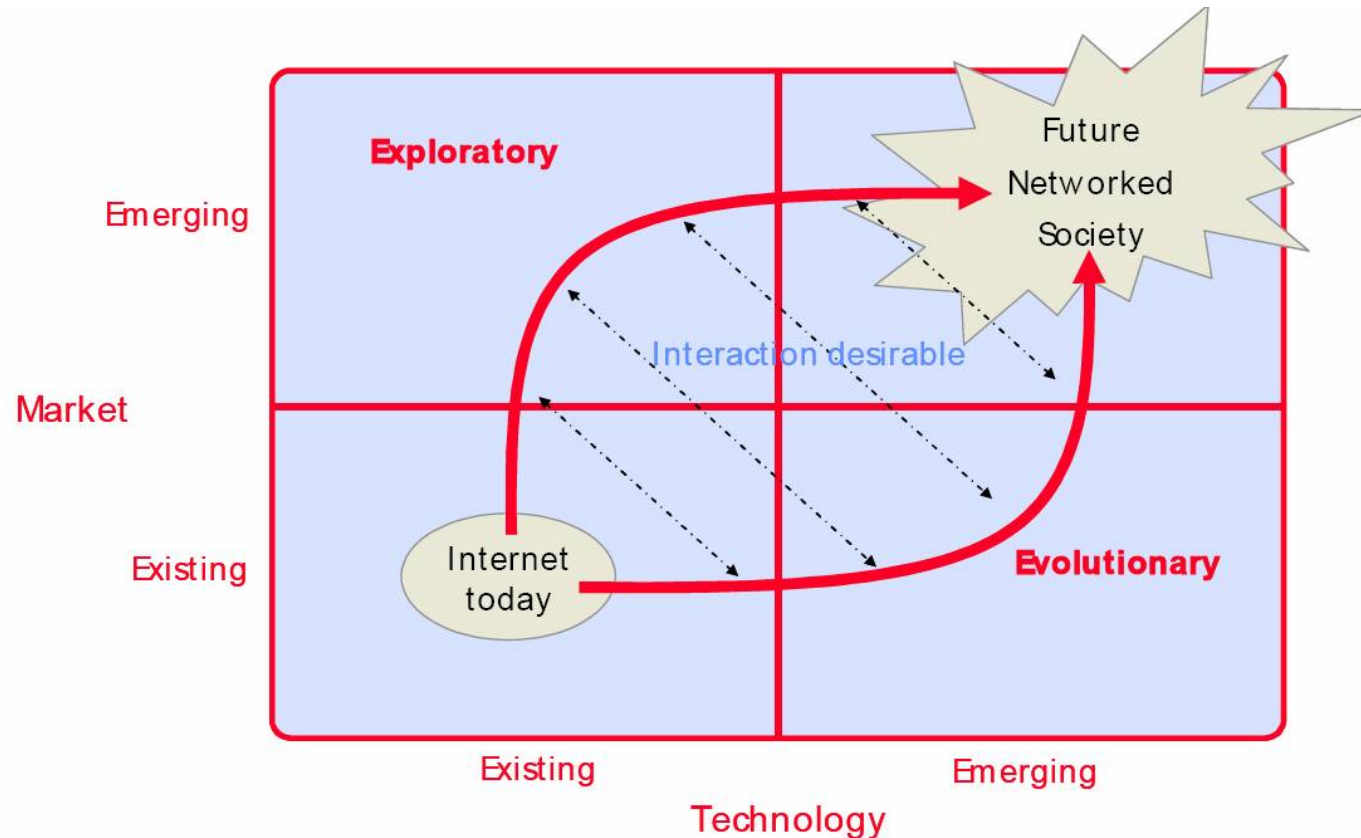
Operators/Providers

- Extendable set of mechanisms
 - Add functionality needed locally (e.g. for traffic engineering, accounting, management, ...)
 - **Easy to deploy new services**
- Technical impact
 - Architecture, network structure, management
 - Control plane (scalable, controllable, debugable, ...)
 - Reduced dependencies between mechanisms
 - Improved robustness
- New value chains
 - New interface between operators and service providers
 - Adopt appropriate solutions with technical impact
 - New services and applications
 - **Early deployment**
 - **Ease of deployment**
 - New business models



Summary

- The EIFFEL (Evolved Internet Future for European Leadership) **view**





Questions?

