A Future Internet Architecture supporting Multipath Communication Networks

Martin Becke, Thomas Dreibholz, Hakim Adhari, Erwin P. Rathgeb
University of Duisburg-Essen, Institute for Experimental Mathematics
Ellernstraße 29, 45326 Essen, Germany
{martin.becke,dreibh,hakim.adhari,rathgeb}@iem.uni-due.de

***Abstract*—The classic layered OSI reference model has reached its limits for the Internet of today. In this paper, we propose a clean-slate conceptual design of a new architecture as a contribution to the ongoing discussion on the Future Internet. We address the shortcomings of the layered model by redesigning the classical model. Our approach differs from the concepts found in prior work, which focus on special parts of the problems (such as the application, the service or the event) by staggering back a couple of steps and trying to see the requirements from a different perspective. Our concept – which is denoted as *Encapsulated Responsibility-Centric Architecture Model* (ERiCA) – focuses on determining the responsibilities by using different planes in addition to a partitioning of the network into different decision domains. With this partitioning, we can reduce the complexity of providing a certain service.**

I. Introduction and Related Work

The history of the Internet is one of the best success stories in the technological areas. It has started with a very few hosts and has grown to the largest network of the world, in terms of architecture as well as in terms of the services supported. Its key idea is the possibility to exchange information among multiple communication partners, irrespectively of geographical distances: the communicating hosts are situated in different networks, possibly being separated into multiple network segments. Usually, the hosts are also confronted with limited resources and with shared media where access time and availability of the medium are not predictable. The goal of a communication between two or more hosts – from an application perspective – is the establishment of transport-specific services, where resources have to be provided as stably and inexpensively as possible.

Nowadays, the rigid layered structure [1] of the Internet protocol stack is becoming an limitation for achieving this goal. In fact, new structures and ways to use this huge network have become the norm; data is stored in data centers, computations are made in clouds and terms such as Software as a Service (SaaS) and on-demand software are becoming more and more usual. All this implies more new issues and difficulties a network designer has to deal with and makes layer violations unavoidable. Therefore, the use of half-layer approaches [2] as well as the use of non-standard API extensions is becoming more and more common.

In this paper, we present a novel clean-slate for the Internet of tomorrow. We would not claim the right to have a new approach which is complete in itself. However, and along this paper, we considered the Future Internet challenges presented by [3] (i.e., mobility, scalability, security, etc.) and demonstrated that our architecture fulfills these criteria. It is here also to be mentioned that clean-slate, as explained by [3], should be viewed as a design process, not as a result in itself. Our goal is to extend the input to the process and to demonstrate an alternative concept. This concept describes a *Encapsulated Responsibility-Centric Architecture Model* (ERiCA-Model),

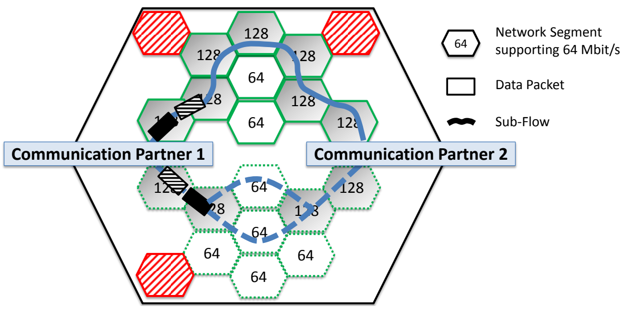

Figure 1. A Cluster of Network Segments

which is based on the idea of recognizing the different actors in the network, their influence as well as their responsibilities. Our approach focuses neither on an application, service nor event driven architecture. Instead, its duty is to determine the responsibilities for the services concerned by establishing a complete transport chain.

II. The ERiCA-Model

A. Basics and Terminology

A *Network Segment* is a network subset that builds a logical or physical connection between two communication partners during a data transmission. Figure 1 shows an example network topology connecting Communication Partner 1 (CP_1) and Communication Partner 2 (CP_2); each hexagon represents a network segment. A *Network Cluster* is the set of network segments that could be utilized to build a combination that allows a data communication between two communication partners. In Figure 1, the network cluster consists of all network segments which could be utilized for connecting the two endpoints; they are represented by hexagons with plain structure. The hexagons with the streaked structure cannot be included in any segment chain between CP_1 and CP_2 and therefore do not belong to this cluster. A possible combination of network segments which can be used for the data transfer is denoted as *Path*. The data packets which are sent via one path are denoted as *Sub-Flow*. The combination of all available sub-flows (in other terms: data which is transferred between CP_1 and CP_2) is denoted as *Flow*. We consider a communication partner representing a service-requesting instance. Within a flow, the logical entity of the data transferred is called *Stream*. Figure 1 shows two streams: here, packets belonging to different streams (Stream 1: plain; Stream 2: streaked) are sent via different paths.

B. Motivation

Common protocol stacks in the Internet of today – like TCP/IP – usually only support a single path per communication direction. Additional extensions may provide a load sharing functionality, e.g. CMT-SCTP or MPTCP. However, these solutions may have serious shortcomings. For example,

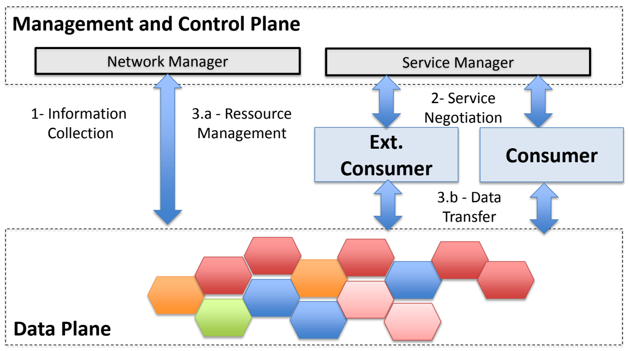

Figure 2. Concept Architecture

the throughput reached with two different, dissimilar segments can be even worse than the throughput reached by just using the best one of them [4].

Let us consider the case of a consumer (here: CP_1) which is about to load data from a data center (here: CP_2), as shown in Figure 1. The consumer has the possibility to connect to two different networks. For the complete flow, 256 Mbit/s are needed. The network is composed of network segments supporting different bandwidth characteristics (gray hexagons support 128 Mbit/s, white hexagons only 64 Mbit/s). Therefore, the transmission would miss the service requirements with a single-path transmission. Even by utilizing multi-path transfer, the required bandwidth could not be guaranteed. In fact, in case of multi-path transfer, the flow could be split into two sub-flows denoted as S_u, and S_l (each using 128 Mbit/s). One possible transmission scenario could be to transmit S_u via the upper part of the cluster and S_l via the lower one.

Concerning S_u, a bandwidth-based routing could allow the network to be able to transfer this data. On the other hand, this is not the case for the lower part of the cluster: there is no path offering a continuously available 128 Mbit/s bandwidth. In this case, it is clear that the limits of the current layer-based Internet architecture have been reached. In fact, in our example, a solution that makes an application-transparent bandwidth aggregation possible – as depicted for the lower sub-flow (dashed line) in Figure 1 – is needed in order to utilize the given network anyway. This scenario is easily extensible to further requirements, such as delay, transmission costs, security restrictions, etc..

C. *Concept Architecture*

In our concept, we differentiate between *Data Traffic*, where service-specific data is transferred, and *Control Traffic* where data is exchanged between the different network components in order to manage the communication itself. That is, as shown in Figure 2, we distinguish between a *Data Plane* as well as a *Management and Control Plane*. The data plane consists of two independent networks; in the best case, they are physically separate networks. Within this architecture, a communication partner, which is willing to access a service, has no direct connection to the accessible medium (here: the data plane). Each access to the resources needed by an application is managed by the management and control plane.

The idea of separating the networks is not new: [5] discusses the consequences, advantages and challenges associated with this design. However, our approach does not focus on the reduction of the router complexity. Instead, the separation of management and data communication allows to find a unique, but not centralized, allocation of responsibilities. Responsibilities are defined as the choice of functions and mechanisms in the bounded system.

1) Management and Control Plane: The Management and Control Plane is a computational entity (e.g., a cloud) whose task consists of gathering, aggregating, grouping and classifying native network characteristics. Example native characteristics are physical network properties (e.g., delay or bandwidth), location-based properties or security-based properties. A similar architecture applying homogeneous services, based on service classes over a meshed network, is described in [6]. However, it only considers bandwidth. Our approach is also able to handle any other network property, too. We denote a set of native characteristics describing the minimum characteristics needed to guarantee a service class as *Quality Class*. Such quality classes build up the base on which the management and control plane works as a mediator, and match them with the application service requirements.

The management and control plane is divided into two major parts: the *Service Manager* and the *Network Manager*. The service manager announces different services and functions, depending on the existing quality classes and their costs. Before establishing a connection, the consumer contacts the service manager by sending a request with its requirements, such as bandwidth, delay upper limits or security restrictions. The network manager, which is in constant touch with the network infrastructure, is aware of the characteristics of the available links. Its task is to manage the available resources, in order to utilize them in an effective way. Depending on the characteristics requested by the consumer, the network manager allocates (if required) the resources needed and releases the remaining resources for future services. Once the service manager has confirmed a service, the consumer is able to start transferring data.

Many other clean-slate designs also focus on the reengineering of the network stack into a new kind of network service-based stack as well. In [2], [7], this is performed by using functional composition. Here, the requirements of the application will be utilized to generate an adapted network stack, as a result of composing functional mechanisms. Our concept could adapts ideas from these approaches, especially for finding a usable description language for the requirements of an application. But, from our point of view, the network should not only be adapted to the application needs. We are convinced that it is sufficient for a network to offer a tuple of existing characteristics that may guarantee a minimal fit to the application requirements intervals. This property may greatly improve the flexibility of the system.

2) Data Plane: It consists of the physical components being responsible for transferring the data. The communication chain could be composed of many different media types (e.g., Ethernet, ADSL, WLAN, LTE, etc.) which all have their individual characteristics.

This gives a short overview about the roles presented in our approach, and how we define the information flow between them. In order to establish a *Communication Chain*, i.e. a set of one or multiple segments connecting two communication partners, decisions (e.g., routing) have to be made, based on gathered information. Clearly, we need a strict organization of where to make and deploy these decisions. In our concept, this is performed by a *Decision Domain* whose functionality will be described in the following.

D. *Decision Domains*

In the Internet of today, the communication partners have – in most of the cases – no influence on and very limited information about the characteristics of the paths used to transfer data. In order to make sure that a service can be established, different strategies are possible. One solution to cope with this difficulty is a complete network separation [7], [8] for the various service classes provided by the different sub-flows which are requested by the services. This approach can be helpful in special cases. However, in the usual case, different consumers have to share the same network in order to achieve cost-efficiency.

From our point of view, there is no need for a strict separation of the services provided by the network. But a way is needed to connect and to route over the usable network segments. Furthermore, additional solutions are needed to aggregate it with other sub-flow characteristics. This is for example the case if many high-bandwidth links are used to bundle performance. In this case, unused resources on neighbor network segments with bandwidth resources larger than the already-used link could be utilized in order to improve the quality (here: throughput) of the existing communication.

We distinguish between two kinds of decision domains:

1) Local Decision Domain (LDD): is a clearly-arranged, autarchic network entity that denotes the smallest responsibility entity in our concept. All information needed to establish the services are available within the entity itself (provided by configurations or policies). The size of a LDD can vary between one network segment with at least one host (e.g., a combination of an ADSL router and a laptop) and complex structures consisting of one or more segments (e.g., a campus network). The information gathered within the network and the communication of the LDD with external networks is performed by an entity called *Edge Device*; it is placed on the borders of a LDD.In our design, and contradiction to [9], no functional dependencies are generated within the network. In fact, possible singular functionalities (e.g., anti-virus functionality) in the network are strongly against our paradigm that a transport medium just provides a homogeneous service class over a heterogeneous, meshed network to support data transport.

2) Distributed Decision Domain (DDD): The information delivered by a LDD is useful for data transfer within the LDD itself. This makes it possible to guarantee services inside the LDD. However, in a typical data transfer, the communication is not limited to a local network only. Instead, communication partners may belong to distinct, and – in most of the cases – distant entities, i.e. to different LDDs. For this purpose, and in order to support the services within multiple LDDs, the management and control plane adopts the functionality of gathering and computing the information belonging to different LDDs, in order to generate a decision base. A set of LDDs is denoted as DDD. This hierarchic architecture makes the management and control plane within a DDD responsible for keeping a balance between the available global resources and the services supported.

Note, that there may be multiple DDDs in most use cases. Let us consider two neighbor DDDs: DDD_1 and DDD_2. A consumer belonging to DDD_1 could be confronted with the situation that a service supported by DDD_2 is needed. For such inter-DDD communication, we designed a strategy where at least best effort quality classes can be established. In this case, DDD_1 has to be registered by DDD_2 as *External Consumer*, as illustrated in Figure 2. That is, in the context of DDD_2, the DDD_1 behaves like a standard consumer (e.g., an application); DDD_1 is able to send service requests to the service manager of DDD_2, and to use the data plane resources of DDD_2.

In other words, dividing a network into different DDDs leads to a new and simplified view on that network. This fulfills the goal of scalability [3], which is one of the most important challenges for the Future Internet. That is, different DDDs may exist for special clusters of similar services. A DDD could e.g. provide a high-level routing decision, which makes it possible to decrease the number of low-level routing entries in a LDD situated in a DDD. Also, our approach handles the economical goal [3]: a DDD is able to allow or to deny access to certain services of a network entity, based on economic criteria. For example, an Internet service provider could provide multiple quality classes at different prices. Depending on the charges, some high-quality classes may only be allowed for premium customers.

III. Functionality

Starting from a single network segment, it is first important to figure out a tuple of basic, unique, non-manipulable physical-based characteristics (e.g., bandwidth, latency, jitter or error rate) which describe this segment. This is performed e.g. by configuration, measurement or calculation and is denoted as *Basic Network Characteristic* (BNC). The set of the BNCs characterizing the segments situated in a LDD are used to build the base information describing the properties of a LDD.

Let us consider a LDD composed of a cluster built by two disjoint paths between two edge devices on the borders. Multiple ways of how to describe the BNC of this LDD are possible. The LDD itself could decide between different description possibilities. For example, the LDD could utilize the available paths for channel bundling (i.e. aggregation to increase bandwidth). Alternatively, the LDD could decide to use one path for redundancy. Here, the LDD changes its properties to decreased bandwidth, but with a reduced error rate.

The LDD expands the property list to provide additional mechanisms and algorithms, such as prioritization of streams in order to improve the performance level of the LDD. It is also possible that no decision making process is desired within the LDD, so the LDD could provide only basic characteristics. Segment bundling as well as repetition can be provided as an additional possible functionality. In the latter case, it is in the responsibility of the DDD to decide how to use these root characteristics. This set of characteristics, as well as the cost of each characteristic, is reported by the edge devices to the DDD. It is further denoted as *LDD Network Characteristics* (LDDNC).

Particularly for security and simplification purposes, an abstraction is needed while gathering information from the edge devices. Here, it is important to minimize the amount of information exchanged between the different entities. The abstraction of the characteristics between the edge devices at the border of the DDD are used to build a database which is denoted as *DDD Network Characteristics* (DDDNC).

A consumer requests a service provided from the service manager in the management and control plane. The service manager sends a request to the network manager, which – depending on the requested service of the communication and based on the data provided in the database – starts searching for a suitable quality class. Here, the network manager performs a lookup for the minimum quality class needed to guarantee the service requested by the application. Quality classes are pre-defined formal definitions which describe network properties that could be relevant to ensure special services or common service types. The limits specified by a quality class could be a result of past experiences or customer requirements. The quality class necessary for supporting a high-performance grid computing data transfer could e.g. request a down and upstream bandwidth higher than 12 Gbit/s, as well as a delay less than 10 ms.

The Service requested by the communication partner triggers the service manager to start the *Normalize, Cluster and Aggregate Process* (NCA). Based on the DDDNC, the network manager starts searching for possible communication tactics between the considered partners and to fill in the *Possible Connection List* (PCL). Here, it is important to notice that the focus of this process is not to find the best path, e.g., based on minimum hops. Instead, the goal is to define as many different possibilities as possible to establish the service. Based on this selection of connection possibilities, the best match from the PCL for a selected quality class has to be identified. Once the matching process is completed successfully, a *Flow-ID* is generated and associated with the match found. This is not a one-time procedure; whenever possible, the PCL will be rechecked for a better connection through the network. This could be done for example during idle time.

In our concept, the service manager acknowledges the application service request by sending a flow-ID for each subflow requested by the service description. At the same time, it informs the network manager about the allocation of this quality class. Depending on the requirements requested, the network manager is able to allocate the network resources, if an additional reservation is needed. This procedure implicitly assures a certain reliability and availability, which is also one of the major challenges [3] for the Future Internet. The corresponding flow-ID may be mapped to each packet in a specific header. The idea of using such a header can also be found in prior work on Future Internet designs, such as [10] introducing a role-specific header. However, in contrast to the static workflow described in [10], we use an identifier which is linked to a kind of more dynamic workflow. This transforms the communication to a dynamic process which allows characteristics to be varied during the data transmission, for example after network failures. During a data transmission, the edge device extracts the flow-ID from a packet, contacts the management and control plane which decides the next LDD to route the packet to. This is similar to flow routing performed by ATM or MPLS with an unique flow-ID.

Inside the LDD, a packet is forwarded – based on LDD-specific policies – to the edge device that is connected to the next LDD. Once a packet reaches a new domain, the same procedure is repeated again. In case of failures, the network manager unit in the management and control plane will be notified by the last edge device. In case of a technical problem for a provided and promised PCL item, the network manager could switch to another path to ensure resilience, even if higher costs are caused. In addition, the network manager is able to change its transmission tactic – in order to increase efficiency or to optimize costs – at any time of the transmission. This happens in background, transparently for the consumer and for the service manager.

One of the obvious use cases related with this dynamical aspect is the support of mobility [3]. In our architecture, the changes of the characteristics of a communication element could be handled differently. The first alternative is to handle changes by the decision domain itself. For a mobile device roaming between two WLAN networks with the same characteristics in the same LDD, the LDD is responsible for reestablishing the connection. However, if mobility events cannot be managed by the decision domain itself, e.g. when one communication partner moves to another LDD, the changes have to be propagated by the edge devices to the network managers, which have to rethink about a new strategy (such as forwarding the traffic via another decision domain).

Furthermore, and related to the flow-ID, improved security – which is another Future Internet requirement [3] – has to be handled. Our approach uses this ID as a kind of a label requested in a service-oriented way. The consumer itself has no possibility to interact with the network organization instances (here: the network manager). This can be compared with a parcel service, where a consumer only knows where the collection point is (in this case, it is the service manager), but he is not able to gather information about the transporter itself (here: the data plane or the network manager). Furthermore, one critical point in the context of security is the avoidance of denial of service attacks. In fact, many possibilities exist, such as restricting the number of request by policy.

IV. Conclusion and Future Work

In this paper, we have proposed the architecture of a new, clean-slate approach for the Internet of tomorrow, which we denote as Encapsulated Responsibility-Centric Architecture Model (ERiCA). This concept makes it possible to recognize the different actors in the network and their influence as well as to manage the different responsibilities by establishing a complete transport chain. We have also reasoned that the proposed architecture fulfills the challenges required by the Future Internet (i.e. security, mobility, scalability, etc.) as defined by [3].

It is here to be noticed that we are in the early stage of this work. As ongoing and future work, we are going to realize a proof-of-concept implementation, in order to demonstrate the advantages of our concept and also to find out possible shortcomings. In addition to it, multiple points have to be analyzed and optimized, such as the clustering and aggregating steps in the NCA process or the efficiency of the decision-making process. In addition to it, important milestones of our future work are the analysis of the scalability of the approach as well as the examination of our concept with a larger number of services with more fine-granular requirements.

References

[1] V. G. Cerf and R. E. Kahn, “A Protocol for Packet Network Intercommunication,” *IEEE Transactions on Communications*, vol. 22, no. 5, pp. 637–648, May 1974, ISSN 0090-6778.

[2] R. Dutta, G. N. Rouskas, I. Baldine, A. Bragg, and D. Stevenson, “The SILO Architecture for Services Integration, controL, and Optimization for the Future Internet,” in *Proceedings of the IEEE International Conference on Communications*, Glasgow/United Kingdom, June 2007, pp. 1899–1904, ISBN 1-4244-0353-7.

[3] A. Feldmann, “Internet Clean-Slate Design: What and Why?” *ACM SIGCOMM Computer Communication Review*, vol. 37, pp. 59–64, July 2007, ISSN 0146-4833.

[4] T. Dreibholz, M. Becke, E. P. Rathgeb, and M. Tüxen, “On the Use of Concurrent Multipath Transfer over Asymmetric Paths,” in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, Miami, Florida/U.S.A., Dec. 2010, ISBN 978-1-4244-5637-6.

[5] J. Rexford, A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, G. Xie, J. Zhan, and H. Zhang, “Network-Wide Decision Making: Toward A Wafer-Thin Control Plane,” in *Proceedings of 3rd ACM Workshop on Hot Topics in Networks (HotNets-III)*, San Diego, California/U.S.A., Nov. 2004.

[6] M. Albrecht, M. Köster, P. Martini, and M. Frank, “End-to-end QoS Management for Delay-Sensitive Scalable Multimedia Streams over DiffServ,” in *Proceedings of the 25th Annual IEEE Conference on Local Computer Networks (LCN)*, Tampa, Florida/U.S.A., Nov. 2000, pp. 314–323, ISBN 0-7695-0912-6.

[7] L. Völker, D. Martin, C. Werle, M. Zitterbart, and I. Khayat, “An Architecture for Concurrent Future Networks,” in *Proceedings of the 2nd GI/ITG KuVS Workshop on The Future Internet*, Karlsruhe/Germany, Nov. 2008.

[8] S. Shenker, “Fundamental Design Issues for the Future Internet,” *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1176–1188, Sept. 1995, ISSN 0733-8716.

[9] F. Liers, T. Volkert, and A. Mitschele-Thiel, “Forwarding on Gates: A Clean-Slate Future Internet Approach within the G-Lab Project,” in *Proceedings of the 9th Joint EuroFGI and ITG Workshop on Visions of Future Network Generations (EuroView)*, Würzburg/Germany, July 2009.

[10] R. Braden, T. Faber, and M. Handley, “From Protocol Stack to Protocol Heap – Role-Based Architecture,” *ACM SIGCOMM Computer Communication Review*, vol. 33, pp. 17–22, Jan. 2003, ISSN 0146-4833.