Future Network and MobileSummit 2010 Conference Proceedings
Paul Cunningham and Miriam Cunningham (Eds)
IIMC International Information Management Corporation, 2010
ISBN: 978-1-905824-18-2

Dynamic Protocol Functionality in Cognitive Future Internet Elements

David WAGNER[1], Jens MOEDEKER[1], Thorsten HORSTMANN[1]
[1]*Fraunhofer FOKUS, Birlinghoven, Germany*
Email:{david.wagner,jens.moedeker}@fokus.fraunhofer.de

Abstract: Composing communication protocols as needed is a topic researched for a long time with moving focus. Nevertheless the vast flexibility of these proposals also prevented the real world success: there was no management solution available that could cope with the many degrees of freedom such architectures offer. This paper discusses the Dynamic Protocol Composition (DPC) architecture developed in the Self-NET project which aims at integrating cognition and autonomics in self-managed networks by the Generic Cognitive Cycle consisting of three phases: Monitoring, Decision-Making and Execution. The presented DPC architecture is therefore designed to serve as an execution capability for a powerful and intelligent decision making process. Since the various potential use cases of DPC in autonomous networks differ heavily with respect to situation awareness and decision making, an example is given. Furthermore the architecture and operation of the DPC implementation is presented. First experiments on packet loss compensation show that dynamically applying Automatic Repeat Request (ARQ) using DPC on routers allows to improve quality of service and network performance significantly without constantly adding load to the link concerned.

Keywords: Future Internet, dynamic protocol composition, cognitive network management, autonomic communication, communication protocols

1. Motivation

The Internet as we know it is the result of thoughtful design that was supported by a repeated pattern of implementation and testing in the 1970's [1]. The decision for a narrow waist of oblivious datagram forwarding that avoids any connection state within the intermediate switching nodes paved the way to the Internet's success because it allowed the development of diverse novel applications and services on top of this very basic foundation. Nevertheless the limits of this design are apparent in today's networks: On one hand the simple mechanisms and assumptions of the Internet Protocol are being softened or even given up in many places, just think of middle boxes like firewalls, NAT-routers etc., VPNs, application proxies and so on. On the other hand devices, networks and service have become very heterogeneous, ranging from error-prone wireless sensor networks (WSNs) to multi-gigabit fibre-based storage area networks (SANs). It becomes more and more apparent that today's Internet protocols, namely the transport protocols, fail to offer a service that is acceptably close to the best possible service. We identified two major subjects that would allow overcoming current limitations in a transparent and evolutionary way: First, protocol functionality should be managed in a dynamic way that allows to take into account changing context like available bandwidth or packet loss caused by very different reasons e.g. user mobility, higher load or bad weather changing the characteristics of wireless links. Second, we propose to give up the end-to-end principle and to establish protocol functionality in intermediate nodes in the network. The management of this in-network soft state seems to be feasible in the Future Internet making use of Cognitive Network Management (CNM) that allows making complex and fine-grained decisions in a cognitive manner.

2. Research on Dynamic Composition of Communication Protocols

The idea to compose protocol functionality according to requirements attracted Internet researchers very early in the 1990's and since has been researched for different motivations, with varying focuses and assumptions. In the beginning frameworks like the early x-kernel [2] have been developed which allowed building static protocols according to the actual needs. These protocols had to be developed, compiled and distributed in a static manner and did not allow for runtime changes. With time, the proposed architectures got more dynamic, proposals like DaCaPo [3], Protocol Boosters [4] or DiPS/CUPS [5] allow dynamic runtime reconfiguration of the protocol stack. The focus of the DaCaPo and DiPS/CUPS architectures is still on the end systems whereas the authors of the Protocol Boosters proposal already described the deployment of additional protocol functionality within the network. Nevertheless there were no results published and this idea was not further pursued. Although the authors expected protocol boosters to evolve fast because of their transparency with respect to the end nodes they still assumed homogeneous deployment within an administrative domain.

The guiding vision for developing this Dynamic Protocol Composition (DPC) framework is a potential node-by-node deployment of an evolutionary DPC in the Future Internet which will be autonomously managed by a much higher degree than todays networks.

3. Our Concept of Dynamic Protocol Composition

The design of our concept for DPC is based on the idea of most Future Internet nodes being equipped with a cognitive management entity that is able to understand the challenges of the current network situation and the consequences of available execution options. The Network Element Cognitive Manager (NECM) will be able to provide consistent and resilient intelligent network management since it takes situation aware decisions using an ontology that allows reasoning based on the stored formal representation of administrator expert knowledge. Therefore the DPC framework itself should rely on NECMs decision wherever possible. Nevertheless the decision making on this level is resource-intensive, in particular it is expected to cost more time than acceptable for many packet-handling decisions. Therefore the design defines several levels of control that e.g. allow the NECM to proactively define rules defining how to act if certain conditions are met. By this the overall system will be able to achieve a very fine-grained control of the protocol behaviour regarding functionality and time without needing to consult the NECM for each decision.

Hence the functional components, called Functional Protocol Elements (FPEs), handled by this framework are designed as small as reasonable: adding a sequence number, sending an acknowledgement or forwarding a packet are typical examples. This results in a world of micro-protocols that allows for highly efficient execution and minimal overhead in the network. The composed protocols can be tailored exactly to the needs of the service in a particular situation and can be adapted at any time. To implement these composed protocols FPEs may express dependencies to other FPEs. This allows single FPE instances and the respective header fields to be used to achieve several independent and more complex functionalities: A byte counter FPE can be used to achieve flow control at the same time as loss detection, maybe combined with automatic retransmissions.

The DPC design therefore is targeted on an efficient and highly dynamic packet handling which incorporates setting up of so called Functional Couplings like Automatic Repeat Request (ARQ) between nodes. To achieve this goal there is an architecture needed that on one hand allows fast and efficient packet processing but on the other hand allows for fast and transparent adaptation triggered either by the NECM directly, by preconfigured policies or by requests from cooperating nodes.

3.1 Applying DPC to intermediate nodes

The concept of minimal functional elements also facilitates the fine grained extension and adaptation of functionality within the network to the current network situation: With such a framework it is possible e.g. to activate adding Forward Error Correction (FEC) to a (UDP-like) video streaming service on wireless hop and at the same time at the same link to activate local buffering and retransmissions for a flow known to be a reliable file transfer. More general applying communication functionalities, e.g. encryption or reliable transmission, dynamically between arbitrary nodes in the Future Internet allows to tailor the performance of each link to exactly the requirements of each flow and to the capabilities available in the current situation.

Still the application of the principle of DPC to intermediate nodes i.e. routers creates need for setting up and managing state in these nodes in an efficient and resilient manner. These challenges do not apply to current end-to-end protocols of the IP-based protocol family that on purpose do not set-up any state in intermediate nodes in order to protect state information from loss and to be able to mask local changes and failures. We assume that in a first phase in-network DPC functionality will be used to improve network performance and capabilities but end nodes won't rely on it and will still manage the crucial state themselves. This scenario will be facilitated by CNM combined with soft state mechanisms (regularly refreshing signalling). The potential benefit of dynamically providing functionality within the network is promising a better network performance and higher efficiency. Therefore this topic is worth to be investigated more deeply.

4. Cognitive Management of DPC

The management of the proposed fine-grained micro-protocol-based DPC framework is a very complex task that requires an understanding of the current network situation. Therefore we assume that situation awareness and cognitive decision making are the key to successful deployment of DPC mechanisms and designed the architecture to allow efficient cooperation with NECM. The different requirements and resources of the DPC framework and the NECM lead to the design of several levels of control interfaces and respective decision making facilities defining different lines of division of work. They allow NECM to proactively define condition-action-policies for DPC and also to directly configure protocol functionality for arbitrary filters.

The complete process shall be pointed out for a basic use case where NECM as the decision making entity, a monitoring component and the DPC framework as execution module run the cognitive M-D-E cycle on a wireless router serving several point-to-point links. The monitoring component providing also filtering and simple correlations will have thresholds defined e.g. for packet loss. In case of violation of this threshold on one interface it will alarm the NECM providing all relevant state information, in particular packet loss, load in each direction, modulation and signal to noise ratio (SNR) for all interfaces. The NECM will use its ontology to characterise the current network situation: The packet loss on the wireless link occurs due to bad physical conditions as indicated by a low SNR. Overload or aggressive queuing strategies are not the reason. The expert knowledge stored in the ontology is also the basis for deriving potential actions and rating them: In this example changing to a lower modulation is not possible since the load on that link is too high. Rerouting the traffic in the mesh topology would be possible but has a low priority since it will increase the load on other links. Since the next hop is known to support DPC, additional options exist. Adding FEC requires a certain amount of bandwidth and would improve reliability but achieving acceptable reliability can't be guaranteed so the rating is not high. Activating ARQ will provide the desired reliability but requires bandwidth for both directions on that link. The rating of this decision is highest so DPC will configured to request the establishment of ARQ with the next hop router for all packets routed on this link. After the well-known time needed for this mechanism to take effect, NECM queries the monitoring component to check for success of the action taken.

So for the CNM DPC provides additional execution options. In order to correctly identify the situations where changing functionalities in the network by the means of DPC is the best option NECM has to be provided with the appropriate expert knowledge stored in an ontology. The quality of the ontology is crucial since it decides on the performance of the overall system. The ontology allows the knowledge base to be transparently updated and extended at any time, conceivably by a supervised or even unsupervised learning process.

5. The DPC architecture and implementation

The DPC implementation developed in the Self-NET project [6] unified framework that replaces the Linux IPv6 protocol stack but runs as a user space application. It is based on the C++ open source Simple and Extensible Network Framework (SENF [7]) and uses packet sockets. For inter-node signalling IPv6 extension headers are used which allow the DPC signalling to be transparent for legacy IPv6 nodes: the highest-order two bits of the used option type are set to zero, by this advising nodes that don't recognise this option to skip over it and continue processing.

The main objects of this framework besides packets are:

Functional Protocol Elements (FPEs) A FPE implements network protocol functionality. Some FPEs require the cooperation with other FPEs so these define dependencies in their Self-Description. FPEs that don't have any dependencies are called basic FPEs and only these FPEs can be instantiated in Composed Protocol Chains (CPCs).

Packet Filters (PFs) A PF is defined by a set of packet attributes which partitions all packets into matches and not-matched, e.g. a destination IPv6 network as used in routing rules. More complex examples use more attributes like source address, traffic class, flow label, transport protocol, port address etc. Two different filters are either disjoint, have a partial overlap or one includes the other.

Composed Protocol Chains (CPCs) A CPC consists of a PF and a (ordered) chain of FPEs that defines what this nodes does with a packet that matches the PF. CPCs can be configured to be linked by a common signalling system. A CPC defines a unidirectional protocol composition, this allows DPC functionality to be tailored to the actual needs which are often asymmetric: Two nodes will usually have different CPCs active if they are configured to collaboratively provide a certain functionality, e.g. when providing ARQ (shown in figure 3).

5.1 Components and operation

An overview on the modules of the implemented DPC architecture and their interoperation is given in figure 1. The main components and their roles are introduced in the following:

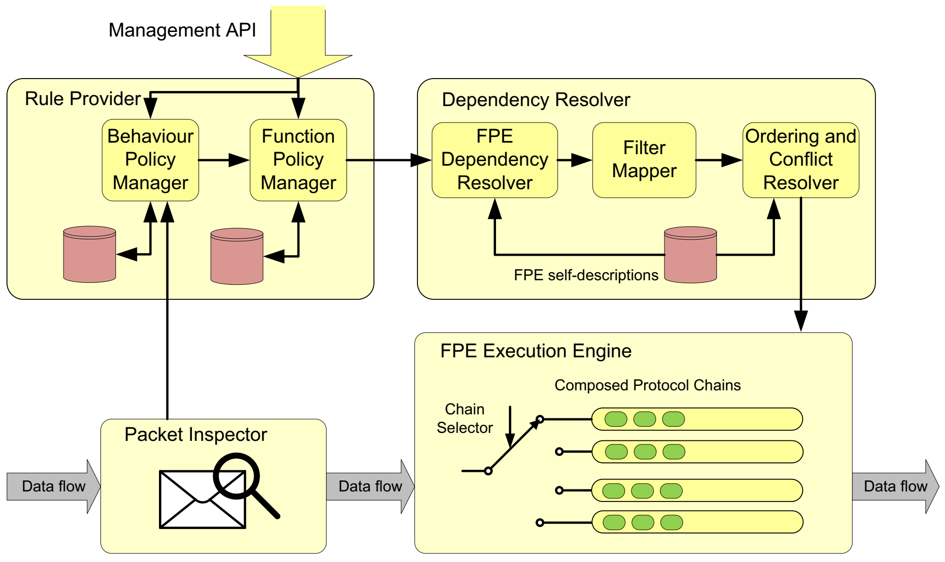

Figure 1: Dynamic Protocol Composition Architecture

The Packet Inspector (PI) This module checks an incoming packet for so called Active Information Elementss (AIEs) which trigger the reconfiguration / activation / deactivation of a specific FPE, e.g. a packet could contain an activate CRC-check flag.

The Rule Provider (RP) This module manages policies for the application of FPEs for certain filter and policies for reacting on modification requests from other nodes received via AIEs. The FPE policies include rules as known from routers or middle boxes like routing rules or firewall rules. More important these rules may define additional functionality, e.g. FPEs for adding or checking checksums, adding sequence numbers, triggering or sending acknowledgements, time based retransmission, buffering, etc.

The Dependency Resolver (DR) This module composes the set of CPCs that is defined by the rule sets provided by the RP by executing three steps: First, the dependencies of the FPEs are resolved using the Self-Description of the FPEs. Second, policies with overlapping PFs are resolved and merged to a set of policies with non-overlapping PFs but larger sets of FPEs. In the third step the FPE are ordered, again using their Self-Description.

The FPE Execution Engine (FEE) This module is responsible for the execution of the configured CPCs. For each incoming packet it selects the best matching filter, just like a router selects the longest matching prefix, and applies the FPEs in the order given by DR.

6. Initial trials

In a first step we implemented the basic components RP, DR, PI and FEE and six FPEs that in combination allow two hosts to set up an ARQ functionality for the link between them. At this point of time the implementation does not include the CNM so the focus is on the evaluation of the DPC framework and its implementation. Nevertheless the selected execution corresponds to the scenario described in section 4. and shows a link that is affected by packet loss (10%, Gaussian distribution) which is after some time transparently and seamlessly compensated by a hop-by-hop ARQ mechanism dynamically activated using the DPC framework. The experimentation

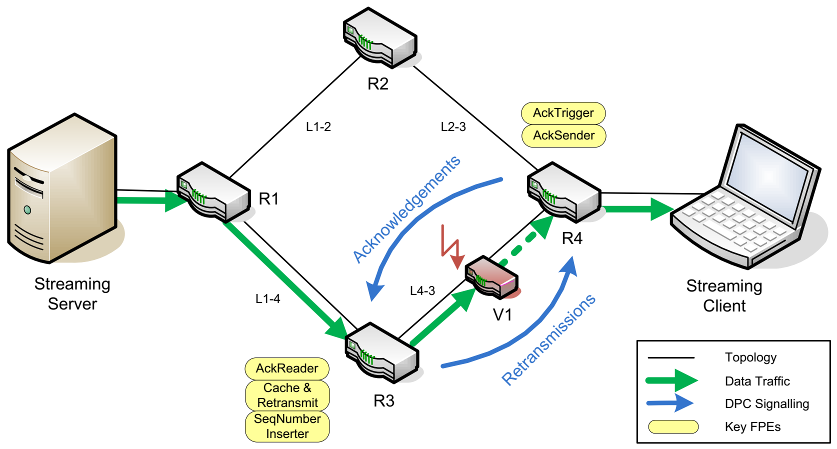

Figure 2: Experimentation network setup

setup as depicted in figure 2 consists of seven Linux based PCs connected with Fast Ethernet. Two of them play the role of a streaming server and a streaming client and use a standard Linux IPv6 protocol stack. Four others, R1-R4, are equipped with the Fraunhofer DPC implementation and shall represent the routers of a wireless mesh network that serves the client with an Internet connection. The seventh computer V1 uses a standard Linux IPv6 protocol stack and allows to induce packet loss in a defined manner at the link between R4 and R3 with the Linux kernel network emulation feature Netem [8].

In the experiment we start with no loss and no DPC functionality except forwarding active on all routers. In a first step we induce ten per cent packet loss at V1. After some seconds we manually activate ARQ for transmissions from R3 to R4. This is implemented by two CPCs on R4 and two CPCs on R3, one sending and one receiving chain for this interface on each router. Please find the precise composition in figure 3. The packets to be forwarded to R4 are extended by a Hop-by-Hop extension header carrying a Self-NET DPC sequence number option. It will be acknowledged by a packet carrying only a Self-NET DPC acknowledgement option.

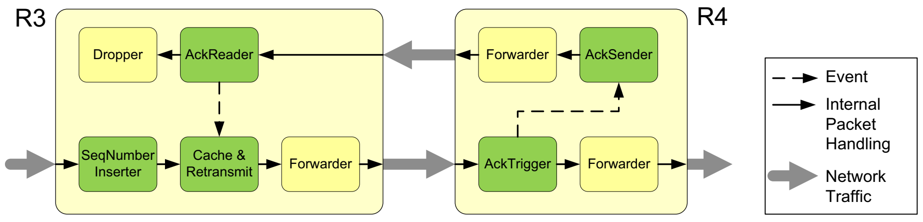

Figure 3: CPC configuration when ARQ is enabled

6.1 Initial results

Our first measurements are depicted in figures 4 and 5. A first result is that our user space implementation creates no significant delay and is reasonable fast, we can see roughly 1 ms delay over four hops including three nodes where DPC replaces the IPv6 stack. Figure 4 depicts the packet loss in the last second for each given point, in fact after activation of ARQ not one packet is lost. The increase of packet loss induced at 8.1 seconds and the improvement after 11.4 seconds can be seen. While before activation of DPC-based ARQ in figure 5 only one main cluster of delay can be recognised at ca. 1 ms, after point 19.5 two, maybe three, additional clusters can be recognised, representing packets that reach their destination after the first, second or third retransmission.

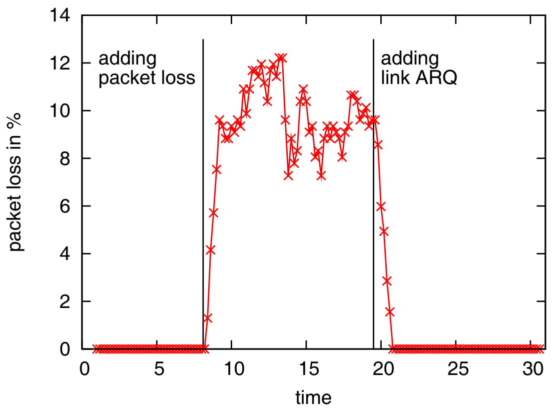

Figure 4: Packet loss at receiver

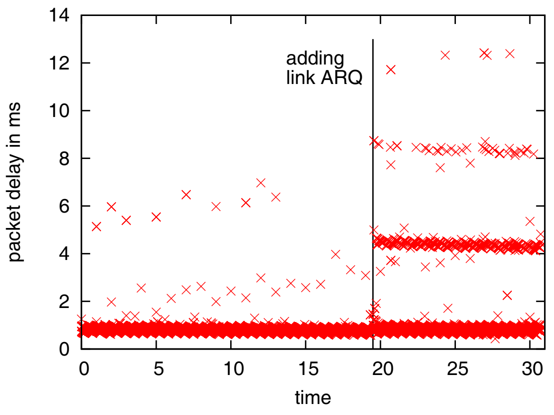

Figure 5: Packet delay at receiver

6.2 Assessment of results

As expected from work in this specific area the application of a hop-by-hop ARQ significantly improved the link reliability while inducing higher jitter. Though jitter and network load are lower than for end-to-end ARQ as used by Transmission Control Protocol (TCP) since this mechanism causes acknowledgements and retransmissions to be forwarded along the complete end-to-end path. Additionally the proposed architecture allows removing that functionality when the conditions improve and by avoids constant load on the network. More general the results show that certain properties of network performance can be significantly improved by the application of the DPC approach between routers. More specifically the implementation developed is validated to be able to provide protocol functionality within the network that improves the network performance for unaware end-nodes. It is also shown that the implementation allows dynamically changing the provided functionality and adapting it to the requirements without interruptions.

The experiments demonstrate that it is possible and may be effective to implement communication protocol functionality within the network. Additionally they show, that these mechanism can be applied and may dynamically changed transparently to the end nodes by a generic architecture that is designed to be very detailed and fine-grained managed by an external entity.

7. Conclusion and Outlook

The concept of applying DPC principles to intermediate nodes proves to be advantageous and the first experiments show that DPC allows to greatly improve quality of service in a seamless and transparent way. There are two main areas that require deeper research: First, a very important topic for further research is the definition of a control and management architecture for the integration with a CNM that bridges the gap between a fast reacting execution and resource-intensive and time-consuming decision-making. Second, the formal storage of decision relevant knowledge in ontologies shall be extended. Here, the approach to focus on selected use cases and to research their aspects in detail has been productive and since the set of potential parameters is overwhelming, this approach should be continued for further research.

In general the combination of DPC mechanisms with its many degrees of freedom with CNM that allows to autonomously make situation-aware decisions based on knowledge is very promising and opens up many new opportunities to increase the efficiency of operating communication networks and by this save scarce resources like spectrum and energy.

Acknowledgment The work presented in this paper was supported by the European Commission Seventh Framework Programme through the Self-NET Project [6].

References

[1] D. Clark, “The design philosophy of the darpa internet protocols,” in *SIGCOMM ’88: Symposium proceedings on Communications architectures and protocols*, (New York, NY, USA), pp. 106–114, ACM, 1988.

[2] N. C. Hutchinson and L. L. Peterson, “The x-kernel: An architecture for implementing network protocols,” *IEEE Trans. Softw. Eng.*, vol. 17, no. 1, pp. 64–76, 1991.

[3] M. Vogt *et al.*, “A run-time environment for da capo,” in *Proceedings of International Networking Conference INET’93* (B. Leiner, ed.), pp. BFC–1–BFC–9, August 1993.

[4] D. Feldmeier *et al.*, “Protocol boosters,” *Selected Areas in Communications, IEEE Journal on*, vol. 16, pp. 437–444, April 1998.

[5] N. Janssens *et al.*, “Towards hot-swappable system software: The dips/cups component framework,” April 2002.

[6] “Self-NET (Self-Management of Cognitive Future InterNET Elements).” EU FP7 project INFSO-ICT-224344. `https://www.ict-selfnet.eu`.

[7] “The simple and extensible network framework.” `http://senf.berlios.de`.

[8] “Netem homepage.” `http://www.linuxfoundation.org/collaborate/workgroups/networking/netem`.