This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2008 workshop proceedings.

Distributed Protocol Stacks: A Framework for Balancing Interoperability and Optimization

Dzmitry Kliazovich and Fabrizio Granelli
DISI - University of Trento
Via Sommarive 14, I-38050 Trento, Italy
E-mail: [klezovic, granelli]@disi.unitn.it

Abstract— **This paper discusses architectural details related to a possible evolution in the protocol stack design, enabling to move a set of functions implemented at a particular layer of the protocol stack of one node to another element of the network (node, router, or base station). Communication with the functional block detached in such way is performed using a predefined communication interface. This novel paradigm is refereed to as Distributed Protocol Stacks, and aims at providing a framework for performance optimization of data transfer, improved QoS provisioning, as well as advances in network design, operation, and management.**

Keywords-TCP/IP, distributed protocol stacks, OSI/ISO layered model

I. INTRODUCTION

Current design of network architectures is based on layering principles introduced by ISO/OSI model [1] with the main target to support interoperability and fast deployment of different networking technologies which form current Internet.

The ISO/OSI protocol stack model specifies the set of functions provided by each protocol layer to the immediately higher layer above and the services requested from the lower layer. Any change or replacement of one protocol layer should not affect other layers. These layering concepts formed the basis for the de-facto standard protocol stack on the Internet: the TCP/IP protocol stack [2].

Standardization of layered protocol stacks enabled fast development of interoperable systems and leaded the success of Internet, demonstrating good performance in homogeneous networks which rely on similar physical layer technologies. However, nowadays the networking environment significantly differs from the one the TCP/IP reference model was designed for. Current trend in networking is towards heterogeneous networking [3] with the main driving factor represented by rapid development of large scale wireless networks [4].

In this heterogeneous environment, TCP/IP shows poor performance [5, 6], and the reasons for such performance degradation are different. Several researchers identified the layering structure and lack of cooperation and coordination between layers as the main reason for performance degradation [7, 8]. As a result, many TCP/IP optimization solutions have been proposed which require introduction of a certain degree of awareness and cooperation among the protocol layers [9]. Cross-layering is a relatively new but promising approach for the design of next generation protocol stacks. In order to be widely accepted, it also requires an effort from standardization bodies and corresponding industries for unification of a cross-layer framework and signaling architecture.

However, the cross-layer design is not the only design approach explored on the way of TCP/IP protocol stack adaptation to heterogeneous network environment. Another concept facilitating design is commonly referred to as "agent-based networking". It aims at introduction of active functionalities into the passive network core. Potentially such agents can be implemented at any layer of the protocol stack. However, there is a clear trend for high layer agents. For example, in Explicit Congestion Notification (ECN) scheme [10] a specific agent running in ECN-enabled router is able to inform TCP sender about growing network congestion by marking ECN bit in the packet header. Based on ECN information, the TCP sender adapts outgoing data rate to the glowing network congestion accordingly.

Nevertheless, the vast majority of network agents is implemented at the application layer and is commonly known as proxy agents. The functions implemented by the proxy agents depend on the goals for introducing the proxy. For example, the most widely used type of proxies is Web proxy - which serves as a gateway between a particular Intranet and Wide Area Network (WAN) and may include blocking or other content management functions.

Another group of proxies, for example caching proxies, aims at data transfer performance improvement. An interesting approach is presented in [11]. The authors discuss on the possibility of introducing customized proxies in the network which can perform the functions of content filtering, compression, encryption, remote cashing, and other on a per-client basis.

Furthermore, both cross-layer design and agent-based networking are considered to form an essential basis for novel trends in networking, such as active networking [12] and cognitive networking [13].

In this paper, we combine all three paradigms present in networking design, i.e. layering, cross-layering, and agent-based networking, ensuring their consistence in a new framework, called Distributed Protocol Stacks. Specifically, we propose an improved version of the protocol stack abstraction

operating at higher granularity being able to abstract not only an entire layer as a set of functions , but also part of the functionalities of a layer. Cross-layer design ensures tight cooperation and proper signaling between the abstracted functional blocks. Finally, similar to agent-based networking, the abstracted functional blocks can be detached from the protocol stack and operate in the network.

The remainder of the paper is organized as follows: Section II presents the core of the proposed Distributed Protocols Stack framework; Section III presents a case study which corresponds to detaching of the specific set of functions from the protocol stack responsible for transport layer acknowledgement generation and locating them in-network; Section IV presents performance benefits of the presented case study evaluating quantitative results; Section V opens the discussion on the applicability of the proposed approach; and Section VI concludes the paper with a summary and directions for future work on the topic.

II. DISTRIBUTED PROTOCOL STACKS

The Distributed Protocol Stacks architecture extends traditional layered (ISO/OSI or TCP/IP) protocol stack by allowing abstractions of "atomic" functions from a specific protocol layer and by providing means of detaching the abstracted functional blocks from the protocol stack in order to relocate them within the network ("in-network").

Fig. 1 presents the details of the Distributed Protocol Stack architecture. The design process is composed of the following procedures: abstraction, detachment, connection, and execution.

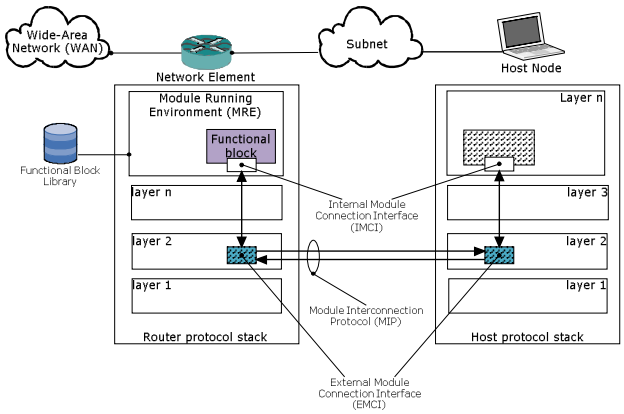

Figure 1. The Distributed Protocol Stack architecture

Abstraction. Before a specific function or a set of functions of the protocol stack can be distributed over the network, they should be abstracted and detached from the protocol stack of the host node.

Identification of the functions to be abstracted depends on the optimization goal and is performed on a case-by-case basis. However, as a general recommendation an abstraction should be performed with non-time critical functions, functions which work on packet basis and do not require continuous access to the internal to the kernel structures. Ideally, abstracted functions should fit into a single functional block which operates at a packet flow basis and requires minimum or no input from the host protocol stack. The output of the abstracted functional block should be applied to the packet flow (for example controlling a single bit in a packet header), trying to avoid the requirement for direct communication with the host protocol stack.

Examples of protocol stack functional blocks that could be easily abstracted include TCP ACK generation module, header compression, IP security related functionalities, congestion related packet drop notification, advertise window adjustment in TCP, and many other.

Detachment. Once the identified set of functions is abstracted as a standalone functional block within a protocol layer, it can be detached and moved into the network. This procedure requires a certain level of cooperation from network elements (routers, switches, or gateways). In particular, network element can be considered "friendly" to the proposed Distributed Protocol Stack if they provide an environment able to support execution of the detached functional blocks – the Module Running Environment (MRE) - as an extension of their protocol stack (see Fig. 1).

MRE provides universal ways for registration and execution of different functional blocks. For example, it may provide a set of standard API functions which can be used by the host node to first transfer the abstracted functional block realized in the set of instructions understood by MRE (module description script language), and then register and run the transferred module.

Alternatively, avoiding the need for module transfer and registration procedures, functional blocks could be chosen from functional block library implemented at the network element. Execution of such blocks at the network element could be controlled by the host node or configured by network operator.

In this paper, for sake of simplicity and aiming at providing a realistic scenario, we assume an infrastructure wireless network scenario where the mobile node detaches some of its protocol stack functions onto the base station. The base station is considered to be friendly in terms of MRE implementation, while other network nodes do not necessarily provide MRE functionality. MRE is considered to be implemented with functional blocks already installed and running at the base station - avoiding host node driven transmission, installation and configuration of the abstracted functional blocks.

Communication between the detached functional block with the host protocol stack is performed using a Module Connection Interface (MCI), which is designed to provide communication between the detached functional block and the host protocol stack.

CMI is composed of two components:

- *Internal Module Connection Interface* (IMCI) connects the detached functional block with MRE at the base station side, while at the host node it provides communication interface with the protocol layer the functional block has been detached.

- *External Module Connection Interface* (EMCI) component provides communication between the detached functional block and the host protocol stack across the network with the use of External Module Communication Protocol (EMCP).

The main idea behind CMI separation into internal and external parts is designed for the purpose of module communication overhead reduction. In particular, EMCI components could be implemented at the lower layers or the protocol stack, leading to fewer header overhead and faster processing. The communication with the IMCI located at the protocol layer where the detachment was performed is performed locally within the protocol stack and it thus does not consume network resources.

Execution of the detached functional block can be triggered by the host node using MRE module installation primitives, or can be configured and running by the base station - without requiring interaction with the host node. In the later case, the base station is responsible for notifying its clients with the information related to the list of functional blocks available. Nevertheless, it should also consider the case of operating with clients which are unaware of functional blocks running at the base station or clients which do not support such operation. Operation of the detached functional blocks should be performed in the transparent way, causing no communication performance degradation.

The design of distributed protocol stack solutions should be driven by cost / benefits analysis, where cost is related to (possibly) reduced interoperability and increased communication overhead between the detached functional block and the host protocol stack, while benefits can be measured in terms of protocol stack performance, enablers for novel user applications, or be driven from the network operator perspective.

In the next section, we present a case study targeting data transfer optimization in infrastructure network scenario where benefits are related to bandwidth and delay improvements.

III. A CASE STUDY

This section provides a discussion on one of the possible applications of the proposed general framework for distributing protocol stacks, providing design and implementation details on a concrete example.

To this aim, the part of transport layer of the receiver responsible for TCP ACK generation is selected as a candidate for detachment. TCP ACK generation is well-defined, can be easily abstracted into a standalone module, and operates on a packet-flow basis, making it an excellent candidate for the abstraction and for detachment.

The main goal of TCP ACK detachment is related to performance optimization derived from the avoidance of TCP ACK packet transmission over the radio link between the mobile terminal and the base station, which in most of the cases becomes a bottleneck of the end-to-end connection. Thus, avoiding its transmission will lead to the capacity increase on the uplink channel for a particular mobile node (in case of channels separated in time or frequency, like in cellular networks) or to the entire cell capacity increase in case of shared medium (like in WiFi case).

Originally, the idea of TCP ACK substitution with a short request sent at the link layer, called ARQ proxy, was presented by the authors in [14] for cellular and in [15] for WiFi networks, while in this paper we briefly review its main architectural techniques from the point of view of distributed protocol stacks design.

Fig. 2 presents the details of the proposed distributed protocol stack solution. Specifically, it shows TCP ACK generation detachment from the mobile node protocol stack and its implementation at the base station providing MRE on top of physical and link layers.

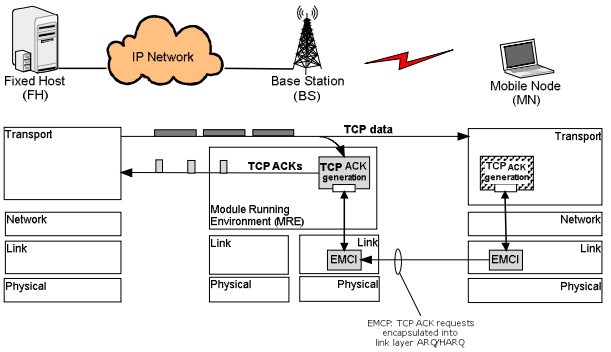

Figure 2. ARQ proxy architecture.

The base station continuously sniffs packets destined to various mobile terminals. Whenever a TCP data packet is detected, it generates a TCP ACK confirming the reception of TCP data packet (assuming the case it is received in sequence). For TCP ACK generation, ARQ proxy simply copies the required fields (IP addresses, port numbers, and flow sequence numbers) into the corresponding fields of a TCP ACK packet template previously allocated in the memory. The generated TCP ACK is not immediately released into the IP network core, but stored locally in a hash table at the base station.

At the mobile node, whenever a standalone TCP ACK packet needs to be transmitted, the IMCI attempts to replace its transmission with a short request sent at the link layer. To this aim, it computes TCP ACK unique identification and sends it to EMCI located at the link layer. TCP ACK identification can be obtained by application of hash functions onto the TCP data which trigger TCP ACK generation [14], or alternatively link layer sequence numbers can be used (like in case of WiFi [15]).

The EMCI is designed to operate on top of ARQ/HARQ protocols running at the link layer, which enable signaling overhead reduction. Specifically, whenever a mobile node needs to output a standalone TCP ACK packet acknowledging in-sequence segment delivery, it issues a short request encapsulated into ARQ/HARQ link layer frame. Upon reception of this frame, the base station obtains TCP ACK packet from the memory and sends it towards fixed sender located in the WAN.

IV. Performance Evaluation Results

The main performance advantage of the discussed ARQ proxy approach comes from substitution of the transmission of TCP ACK segments (which are considered as ordinary data by the link layer) with a short request encapsulated and sent along with ARQ/HARQ frames at the link layer. It should be underlined that in TCP ACK generation performed at the base station does not require TCP flow-related information and thus can only be used for in-sequence segment acknowledgement. In other scenarios such as TCP connection establishment, generation of duplicate TCP ACKs, or TCP ACKs carrying information about exhausted buffer at the receiver, TCP ACKs are not substituted and sent using regular scheme.

Fig. 3 and Fig. 4 provide a quantitative measurement of TCP flow performance improvement in terms of data throughput improvement and Round Trip Time (RTT) reduction enabled by application of ARQ proxy approach in a WiFi cell environment. Simulation scenario corresponds to a throughput of a single flow for an IEEE 802.11b access point running at 11Mb/s physical data rate. The rest of the details related to the scenario and simulation setup along with other measurements can be found in [15].

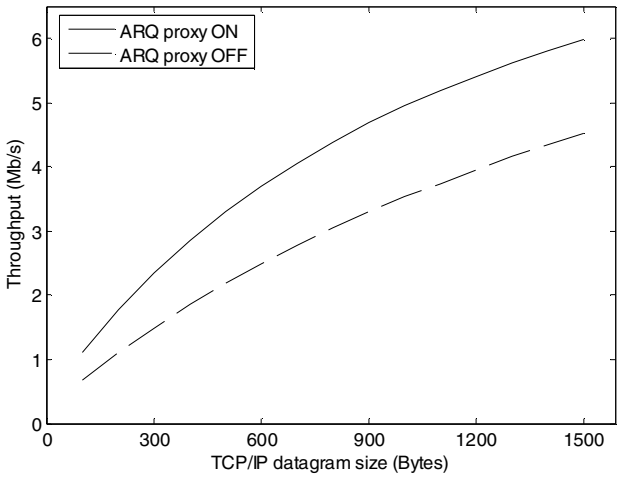

Figure 3. TCP throughput imrovement with ARQ proxy.

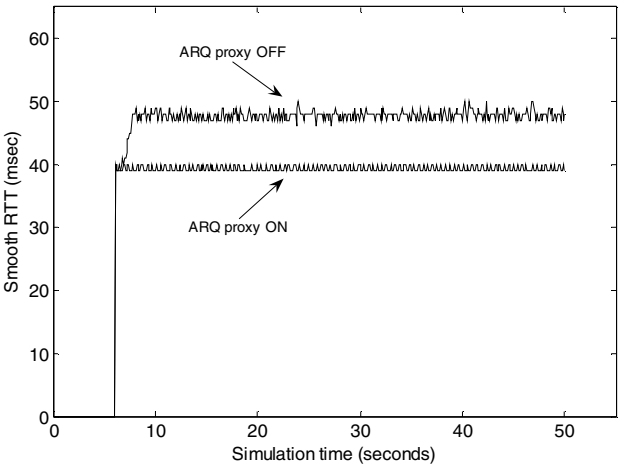

Figure 4. TCP Round-Trip Time (RTT) reduction with ARQ proxy.

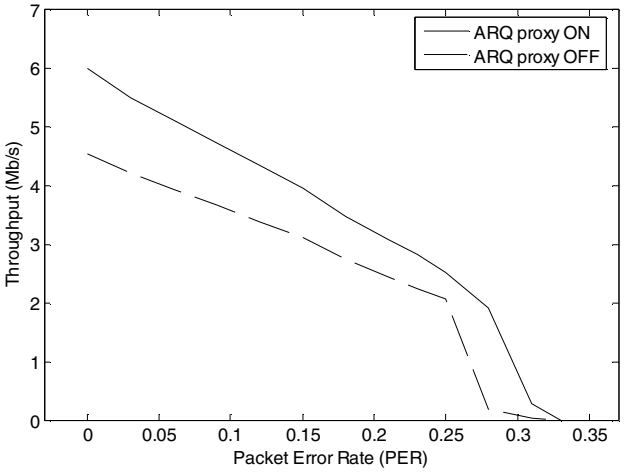

Figure 5. TCP throughput against wireless link errors.

Summarizing, ARQ proxy performance ranges between 25% and 100% depending on TCP/IP datagram size. Additionally, it improves the delay on TCP receiver feedback channel as well as demonstrates higher tolerance to link errors (see Fig. 5) present at the radio link.

V. Discussion

The case study presented in the previous section demonstrates one of many possible applications of the proposed framework for the design of Distributed Protocol Stacks. Generally, reallocation of some protocol stack functions in the network creates an additional degree for protocol stack optimization. However, usually it comes at a price of reduced interoperability due to the necessity to maintain connection between the detached functional block and the host protocol stack. This becomes a limiting factor in cases with dynamic topology and high mobility.

In the presented case study, in order to maintain interoperability and fast handover we avoided the maintenance

of any state related dependence between the protocol stack of the mobile node and pieces of code running at the base station. Specifically, the TCP ACK generation module running at the base station generates TCP ACKs based on the in-transit data flow analysis without requiring any connection state information.

As a result, in case network topology changes rapidly and the protocol stack of mobile node losses connection with the detached TCP ACK generation module of one base station its operation is not affected.

VI. Summary and Future Perspectives

This paper introduces a novel approach for the design of protocol stacks, which can be employed in the design of next generation protocol stack as well as for current optimization of TCP/IP in heterogeneous network environment.

The core of the proposed framework allows an abstraction of functional blocks inside protocol layers, their detachment and movement into the network core, while ensuring optimal communication with the protocol stack. This approach is the result of combination of fundamental leading paradigms evidenced in TCP/IP protocol stack design, i.e. layering, cross-layering, and agent-based networking.

The proposed approach is called Distributed Protocol Stacks due to its ability to outsource and implement certain protocol stack function in the network core, while maintaining control over their functionality. Following the presentation of general framework, the paper presents a case study providing concrete implementation details and analyzes the main issues.

Future work will be aimed at adapting the concept to the cognitive networking paradigm, in order to support self-adaptation of the network also in terms of protocol stack functionalities.

References

[1] B. Jain and A. Agrawala, “Open Systems Interconnection,” New York: McGraw-Hill, 1993.

[2] M. W. Murhammer and E. Murphy, “TCP/IP: Tutorial and Technical Overview,” Upper Saddle River, NJ: Prentice-Hall, 1998.

[3] D. Kliazovich, F. Granelli, G. Pau, and M. Gerla "APOHN: Subnetwork Layering to Improve TCP Performance over Heterogeneous Paths," IEEE Next Generation Internet Design and Engineering (NGI), Valencia, Spain, April 2006.

[4] T. S. Rappaport, A. Annamalai, R.M. Buehrer, and W.H. Tranter, “Wireless communications: past events and a future perspective,” IEEE Communications Magazine, vol. 40, no. 5, May 2002, pp. 148 – 161.

[5] C. Barakat, E. Altman, and W. Dabbous, “On TCP performance in a heterogeneous network: a survey”, IEEE Communications Magazine, vol. 38, no. 1, January 2000, pp. 40 – 46.

[6] G. Xylomenos, G.C. Polyzos, P. Mahonen, and M. Saaranen, “TCP performance issues over wireless links”, IEEE Communications Magazine, vol. 39, no. 4, April 2001, pp. 52 – 58.

[7] S. Shakkottai, T. S. Rappaport, P. C. Karlsson, “Cross-layer design for wireless networks,” IEEE Communications Magazine, vol. 41, no. 10, pp. 74 – 80, October 2003.

[8] Qi Wang and M. A. Abu-Rgheff, “Cross-layer signalling for next-generation wireless systems,” Wireless Communications and Networking (WCNC), vol. 2, pp. 1084 – 1089, March 2003.

[9] V. Srivastava and M. Motani, “Cross-layer design: a survey and the road ahead,” IEEE Communications Magazine, vol. 43, no. 12, pp. 112 – 119, December 2005.

[10] K. Ramakrishnan, S. Floyd, and D. Black, “The Addition of Explicit Congestion Notification (ECN) to IP,” RFC 3168, September 2001.

[11] J. Steinberg and J. Pasquale, “Improving User Relocatability, Practicality, and Deployment in the Web Stream Customizer System,” Proc. Tyrrhenian Intl. Workshop on Digital Communications (TIWDC), Ischia, Italy, Sept. 2007.

[12] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden, “A Survey of Active Network Research,” IEEE Communications Magazine, vol. 35, no. 1, pp. 80 – 86, January 1997.

[13] Q. Mahmoud, “Cognitive Networks: Towards Self-Aware Networks,” Wiley-Interscience, September 2007.

[14] D. Kliazovich, F. Granelli, S. Redana and N. Riato, “Cross-Layer Error Control Optimization in 3G LTE, IEEE Global Communications Conference (GLOBECOM), Washington, DC, U.S.A, December 2007.

[15] D. Kliazovich, N. Ben Halima, and F. Granelli, “Cross-Layer Error Recovery Optimization in WiFi Networks,” Tyrrhenian International Workshop on Digital Communication (TIWDC), Ischia island, Naples, Italy, September, 2007.