Towards Adaptable Networking: Defining the Protocol Optimization Architecture Requirements

Martin André[1] and Fumio Teraoka[2]

[1] National Institute of Information and Communications Technology
4-2-1 Nukui-Kitamachi, Koganei, Tokyo, 184-8795, Japan
andre@nict.go.jp
[2] Faculty of Science and Technology, Keio University
3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Kanagawa, 223-8522, Japan
tera@ics.keio.ac.jp

Abstract. With the recent trend in computer networking that tends to make the networks more dynamic, appeared the need for network protocols to accommodate against changing conditions. In this context, it is important to rely on a clean architecture to share information between protocol entities and to perform optimizations, in order to achieve what we define as adaptable networking. In this paper, we try to identify the needs for a network protocol optimization architecture and describe the care that should be taken when considering such architecture. We perform a study based on observation of the implications of information sharing between network protocol entities and define some requirements for a successful adaptable network architecture. Then we show how these recommendations could be applied in the legacy layered model by presenting an example architecture that respects the identified principles. This work is expected to serve as a basis for any future adaptable network architecture.

Keywords: cross-layer, architecture, adaptable network, optimization.

1 Introduction

In the early years of computer networks, communication link with a neighbor was expected to be stable. The OSI model has established itself as the de facto standard for network communication, offering a clear separation between the network protocol layers. It made the architecture able to easily incorporate new links and nodes, and to accommodate new protocols, applications, and devices. Recently, this base statement has changed. With the apparition of wireless networks, network condition is varying in time and it is often impossible to rely on stable communication links. In addition, over the past few years, radically new network topologies and applications, some of them hardly fitting in the strict layers of OSI model, have appeared. The user has become mobile , expecting ubiquitous connectivity, and network ground shifted from static to dynamic. One of the new challenges is to adapt to such changing conditions.

F. Granelli et al. (Eds.): MOBILIGHT 2009, LNICST 13, pp. 282–291, 2009.

Consequently, the OSI model started to be frequently transgressed with the intention of making the protocol entities more aware of the environment. This violation of the strict layer boundary is often referred to as *cross-layer optimization*. However, the term "cross-layer" is frequently employed through the misuse of language to describe any technique of network protocol optimization. Indeed, optimizations are not only made possible thanks to communication between non-boundary layers but collaboration between distant nodes has to be envisaged too. Moreover, the term cross-layer implies a dependency on the legacy layered model and as a result excludes alternative network protocols models approaches. In the following of this paper, we will prefer the term *adaptation* that we define as the network protocols optimization thanks to information sharing.

A great amount of literature describes mechanisms that allow sharing of information in order to permit optimization in domains such as ad-hoc networks, Quality of Service (QoS), security, or mobility. The proposed mechanisms usually adopt a case by case approach that targets a single issue where a generic infrastructure that helps protocol entities to make the best decision depending on the environment and to accommodate against the changing condition would be necessary. Here also, quite some work has been made in that direction and several generic optimization architectures have been proposed, notably CLASS [1], ECLAIR [2], MobileMAN [3], and more recently Hydra [4].

The rest of this paper is organized as following. We will start by performing analysis of adaptable networking in Section 2, where we discuss the following subjects: extension of the scope of view, implementation of information coming from the network, communication model, and integration with the network protocols model. In Section 3 we present as an example a conceptual architecture that follows the previously identified precepts. Finally, Section 4 concludes this paper and proposes a direction for future work.

2 Adaptable Networking Study

In this section, we analyze the implications of information sharing and we try to identify the key design factors that are needed for an adaptable network architecture. This results in recommendations for a successful architecture.

2.1 Extension of the Scope of View

Supposing that a protocol works in an optimum manner at its own level, it can perform awfully at a higher level because it will, by the nature of the layering model, behave selfishly without taking into account other protocols' constraints. TCP, for example, achieves terrible performances on a link prone to errors. We introduce the concept of *scope of view* as the quantity of information visible by a protocol entity. The idea is to extend the scope of view of protocol entities in order to permit better decisions based on the environment. This is the very principle of adaptable networking.

The question we would like to answer is to what extent we should broaden the scope of view. We will examine the effect of the extension of scope of view

on several important aspects such as complexity, reliability, latency, and performance either locally to a node or globally for a whole network. Fig. 1 depicts the effect of extending the scope of view over these properties, all represented vertically. The horizontal axis represents the four different identified levels of scope of view, with no information shared which reflects the strict conformance to the conventional OSI layered model, the information shared inside a node which is typically called cross-layer optimization, the information shared between nodes that consist in adding a local network view to a node, and finally the information shared between networks. The arrow shows the direction and the range where the effect increases, while the stripped and plain patterns respectively show that the effect is negative or positive.

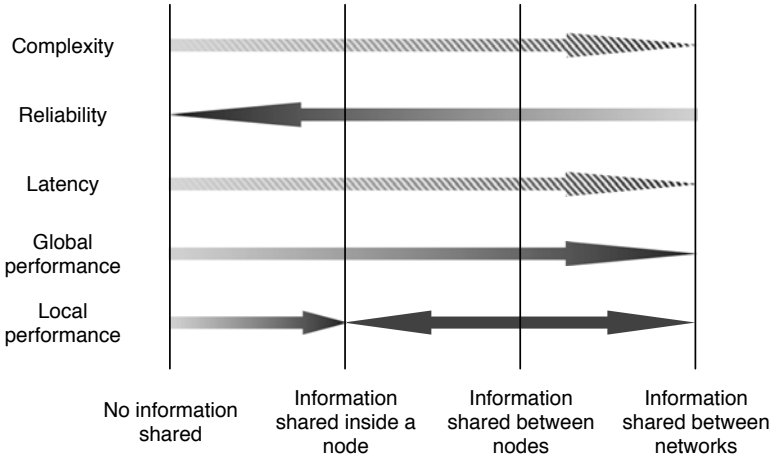

Fig. 1. Effect of extending the scope of view

The broader the scope, the higher the complexity of the architecture and consequently the higher the risk of stability issue. Indeed, "complex" and "unstable" are often synonyms. When more elements are involved, the risk of a failing entity increases too. In addition to complexity and possible failure risk, extending the scope of view also introduces overhead and thus latency. This is particularly visible when information begins to be shared among nodes, caused by the delay of messages transportation over the network. Latency can be problematic when adapting to a phenomenon that evolves rapidly. The delay introduced makes the decision not accurate and possibly makes the desired optimization under-efficient.

Despite these pernicious effects induced by the extension of the scope of view, allowing the sharing of information and the cooperation between protocol entities or nodes can be very beneficial to the global (at a network level) or the local (at a node level) performance. This is the whole point of adaptable networking.

Concerning the local performance however, the choice can be made, thanks to the information coming from the network, to voluntary decrease its performance in favor of the global performance. This choice will then depend on how priorities are set, either to optimize the performance of a single node or the one of a whole network eventually at the expense of some nodes constituting the network. There is no ideal scope of view that suits all constraints. Also, the choice of the scope of view level will rely on the decision that has to be made:

- some decisions have no influence outside of a protocol entity and don't need external information. This is the case of most protocol entities internal mechanisms.
- some decisions require only communication local to a node between protocol entities in order to guarantee optimal performance. For example, TCP may consider the type of data link before reducing the congestion window size after a packet loss is detected.
- some decisions require external information coming from the network in order to have a better vision and guarantee smart choice in full knowledge of the environment. Routing algorithms in a mesh networks scenario are concerned by this type of decision.

The optimization architecture has to deal with the challenge of determining the suitable scope of view for a given situation, i.e., finding a good consensus between advantages and drawbacks. The scope of view must be adapted to the situation and depends on the system requirements and on the decision that need to be made.

2.2 Network-Related Information

We've exposed that the scope of view is not something fixed and that several scenarios must be envisaged. The network view is a logical evolution to information sharing inside a node, and it is necessary under certain condition to achieve optimal results. The architecture should in consequence be able to consider information coming from the network as input for decision making.

A trivial approach would be to let the nodes exchange information directly, but this raises the issue of the security. How can one give credibility to a node? Indeed, the information coming from a distant node modifies the vision of the network environment and is susceptible to induce a decrease in the local performance. It is thus primordial to trust this information in the first step. Besides, a malicious node may inject false information leading to wrong decisions. The system must be prepared against potential attacks. In addition to the security and reliability problems, there is a concern about confidentiality. Some information may be sensitive and, in consequence, shouldn't be accessed by anyone. As a matter of fact, it is necessary to authenticate the nodes in order to establish a trusted network.

But authenticating the nodes isn't sufficient. Let us imagine the case where a node has been authenticated but which is spreading, intentionally or not, erroneous information. This node may be relaying outdated information for instance.

The optimization architecture needs to allow the validation of the information's relevancy by multiplying the source of information, for example by verifying to other nodes any information that can be double-checked. This could be the role of a central server to confront information announced by the nodes constituting the network and to detect inconsistency.

Finally, performance must also be taken into account. It is clearly more efficient in term of message exchange to gather nodes' information on a central server to achieve optimum network cooperation rather than to rely on direct communication between the nodes when unicast messages are exchanged. The latter one is expected to generate lots of messages for full cooperation between the nodes. However, multicast or broadcast message exchange between the nodes has the same order of complexity as communication with a central server. Fig. 2 represents the comparison between the central server and the distributed cooperation in unicast and shows how message exchange complexity is exponentially growing when the number of nodes is increasing.

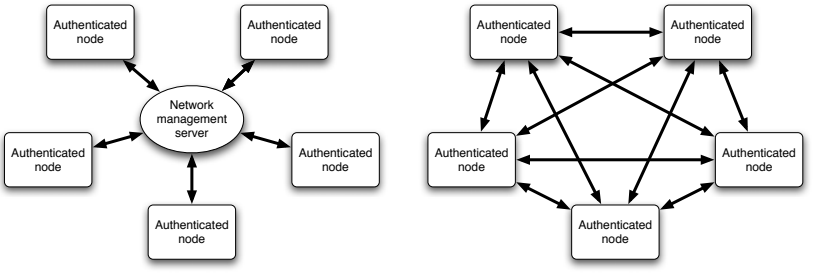

Fig. 2. Central server versus distributed cooperation

In conclusion, it is more appropriate for the implementation of the network view to introduce a new entity that acts as a trusted and reliable network management server, similarly to what is proposed in the IEEE 802.21 [5] developing standard aiming at providing a media independent handover mechanism. This choice has also the benefit to allow greater possibility for additional features. Direct communication between the nodes may not be completely left over, though, but it requires setting up an authentication mechanism.

2.3 Communication Model

There are conceptually three types of communication possible when talking about cooperation between functional units:

- Information data exchange;
- Event subscription and notification;
- Command requests and responses.

The IEEE 802.21 developing standard defines similar services in its Media Independent Handover Function (MIHF), respectively the Media Independent Information Service (MIIS), the Media Independent Event Service (MIES), and the Media Independent Command Service (MICS).

As shown in [6], introduction of dependencies between network protocol entities may lead to conflicting interactions and adaptation loops. The article also points out that the architecture is critical for adoption of technology, as it fixes for the most part the ease of development and the longevity. In other words, this is the role of the architecture to prevent potential conflicts. Considering the cooperation inside a node we can distinguish two types of communication between the protocol entities depending on whether they are subject to conflicts or not. On the one hand, information data exchange as well as event subscription and notification belong to the same category and are considered harmless. Command requests, on the other hand, are potentially harmful.

The communication for the first category of interactions, not likely to cause conflicts, should be performed directly between protocol entities to permit good reactivity for the exchange of information. This applies to information requests and event subscriptions and notifications. Concerning the communication for command requests, however, we should introduce a *supervision entity* that has a global view of the system requirements in terms of QoS, energy consumption, and security and that controls the potentially conflicting interactions.

Similarly to the cooperation inside a node, cooperation between the nodes requires a central authority having the knowledge of the whole network (see Sec. 2.2) and acting as a network management server. The envisaged type of communication is done through a publish and retrieve information manner between the nodes and the network management server, and commands are originated from the network management server to the nodes.

2.4 Integration with the Network Protocols Model

The layered model brings a lot of architectural concepts that are very efficient in today's network topology and which highly contributed to the Internet success. However, the model presents a certain rigidity which makes it difficult to accommodate evolutions, leading researchers to envisage alternatives. One consists in violating the strict layer boundary and is known as cross-layer. Others consist in non-layered approach to the design of network protocols. Among the new paradigms, the Role Based architecture (RBA, [7]) organizes communication in non hierarchical functional units called roles and defines the concept of network heap as a replacement of network stack. The SILO architecture [8] has an intermediate approach and proposes to build network silos (logical blocks stacks) that are assembled on-demand based on the application requirements. We need to think how the optimization architecture should fit in a network protocols model.

An adaptable network architecture has strong requirements that need to be considered with regards to the integration with a network protocols model. First, besides being essential for viability of a solution and strongly contributing to its

adoption, maintainability has a substantial role to play. A good architecture has to be able to deal easily with evolution of the networks, in particular, it must take a special care about the case of a new protocol entity and the one of a new possible optimization.

Considering the case of a new protocol entity, it must integrate easily in the system, i.e., being readily able to communicate with existing protocol entities. For this purpose, the architecture needs to provide a standardized interface to communicate with protocol entities by means of a public Application Programming Interface (API). In addition, existing protocol entities should be able to interact with this new one with ideally no modifications. To this goal, we should introduce an *abstraction level* for the protocols' common capabilities that ensures forward compatibility between protocol entities.

By providing a standardized interface to communicate easily with a protocol entity, we also resolve the problems relating to the case of a possible new optimization. These are not restricted anymore to the limited vision of the architecture designer, and unforeseen interactions can easily be implemented. A public API gives assurance that the set of optimizations is extensible.

In conclusion, the optimization architecture should provide modularity to integrate smoothly in any existing or future network architecture, independently of the network protocols model. In addition, a great importance should be given to maintainability, by providing a public interface to communicate with the protocol entities and introducing an abstraction level for protocols' common capabilities.

3 Example Architecture

We will here present how the principles exposed in Section 2 can be put into practice in the legacy layered model. Fig. 3 represents an example architecture conforming to these principles.

3.1 Layer Abstraction

The first modification made to the legacy layered model consists in the addition of a level of abstraction to cope with protocols differences within the same layer and permit a generic use. It gives the assurance that the compatibility between protocol optimizations is maintained. In our example it takes the form of a standardized API to access the common layer's features and of an extension to the protocol entity (PE) called the *Abstract Entity* (AE), represented in light stripped pattern on Fig. 3. The AE is the interface used for all the communication and cooperation with the protocol entity. Protocol entities are not required to implement an AE, but this extension is a prerequisite in order to communicate in the optimization architecture.

RFC 5184 [9] presents an example of layer abstraction and proposes a standardized API for the link layer abstractions.

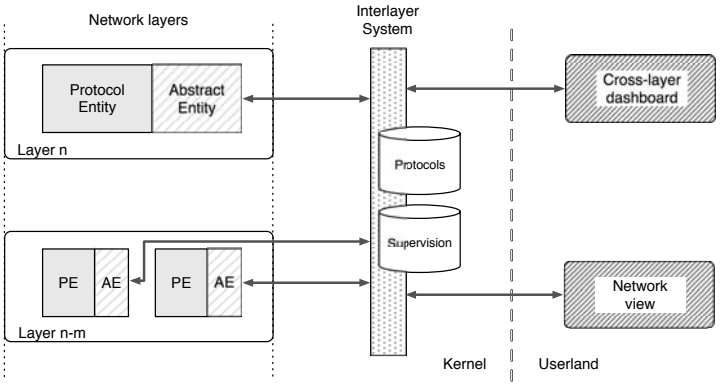

Fig. 3. An example architecture

3.2 Interlayer System

We introduce an entity called *interlayer system*, represented in dotted pattern on Fig. 3, that allows communication between all the protocol entities. It can be considered as the message delivery service, as every message exchange is done via the interlayer system. To this end, the interlayer system stores information about the protocol entities that are registered with it. In addition, it plays the role of the supervision entity that controls the potentially conflicting interactions thanks to a global view of the system requirements, as shown in Section 2.3. It thus stores all necessary information used to avoid interaction conflicts. Inappropriate controls can for example be rejected by the interlayer system either because it is in contradiction with global policy (QoS, energy, etc.) or it is conflicting with another control having higher priority.

3.3 Kernel-Userspace Bridge

The last part of this example architecture illustrate how a kernel-userspace bridge can be used to implement extensions, represented in dark stripped pattern on Fig. 3, that provide additional input for the decision making. Optimization will typically take place in the kernel space of the Operating System (OS) with no operation from the user. However, there are cases where external information input is necessary such as to provide to the nodes a certain knowledge of the network in order to do better optimizations, as shown in Section 2.2. The communication between the node and the network management server should be implemented in the user space of the OS as a daemon and connection with the interlayer system (Sec. 3.2) could be assured thanks to a kernel-userspace bridge.

Another example extension would be an optimization dashboard that gives the complete control of the optimization system to the user. It could be used to enable and disable certain optimization, set system-wide constraints determining

the priorities, or even consult statistics. This extension could help researchers and developers to realize performance measurements of certain optimizations.

4 Conclusion

Our observations on the implications of information sharing for the network protocols resulted in recommendations usable for any adaptable network architecture that we will briefly summarize. There is no ideal scope of view for the network protocol entities, but rather an optimal scope depending on the constraints and it is the role of the architecture to find the best compromise. Network view, as part of the extension of the scope of view, is the natural evolution of cross-layer and makes a lot of sense in adaptable networking. In order to improve the potential for additional facilities, its implementation requires to place a new entity on the local network that provides a global view and acts as a management server. Direct communication between the nodes should still be allowed, but in any case, authentication of the nodes has to be provided. In the same manner, a logical entity that has the global knowledge of the constraints and which purpose is to regulate the communications inside the node is mandatory to assert the reliability of the architecture. Its role is to protect against inappropriate use and conflicting interactions. Concerning the integration of the optimization architecture in a network protocols model, designers should give priority to maintainability of their solution by providing a public interface to communicate with the protocol entities and introducing an abstraction level for protocols' common capabilities. Designers should also think as generic as possible in order to avoid dependence of the architecture over the design of the network protocols.

We presented an architecture that gives an example how the exposed principles could be put into practice in the layered model. The maintainability is achieved thanks to a layer abstraction and the use of a standard API. As for the reliability, a central entity called the interlayer system takes care of supervising all communications between the protocol entities. Finally, the example architecture presented in this paper can easily be extended by means of a kernel-userland interface, and one can imagine applications such as the implementation of the network view or a dashboard to control the system.

Our plans for future work will first be to confront existing solutions against the identified requirements and propose a new solution if none is found satisfactory. Some work remains to be done on the supervising entity. We will explore possible mechanisms that can be used to avoid and resolve the potential conflicts between the interactions. We will also investigate more in depth the communication methods between the nodes and the network management server. Finally, we have in the project to implement a solution to validate our concepts and prove the efficiency and the robustness of our architecture.

We expect this document to serve as a basis for any future work on adaptable network architecture.

References

1. Wang, Q., Abu-Rgheff, M.A.: Cross-layer signalling for next-generation wireless systems. In: Wireless Communications and Networking, vol. 2, pp. 1084–1089. IEEE Press, New York (2003)
2. Raisinghani, V.T., Iyer, S.: Cross-layer feedback architecture for mobile device protocol stacks. Communications Magazine 44(1), 85–92 (2006)
3. Borgia, E., Conti, M., Delmastro, F.: Mobileman: design, integration, and experimentation of cross-layer mobile multihop ad hoc networks. Communications Magazine 44(7), 80–85 (2006)
4. Choi, S.H., Perry, D.E., Nettles, S.M.: A Software Architecture for Cross-Layer Wireless Network Adaptations. In: 7th Working IEEE/IFIP Conference on Software Architecture, pp. 281–284 (2008)
5. IEEE Computer Society: Media Independent Handover Services. Draft Standard for Local and Metropolitan Area Networks. IEEE Computer Society (2008)
6. Kawadia, V., Kumar, P.R.: A cautionary perspective on cross-layer design. IEEE Wireless Communications 12(1), 3–11 (2005)
7. Braden, R., Faber, T., Handley, M.: From protocol stack to protocol heap: role-based architecture. SIGCOMM Comput. Commun. Rev. 33(1), 17–22 (2003)
8. Dutta, R., Rouskas, G.N., Baldine, I., Bragg, A., Stevenson, D.: The SILO Architecture for Services Integration, controL, and Optimization for the Future Internet. In: ICC 2007, IEEE International Conference on Communications, vol. 1, pp. 1899–1904. IEEE Press, New York (2007)
9. Teraoka, F., Gogo, K., Mitsuya, K., Shibui, R., Mitani, K.: Unified Layer 2 (L2) Abstractions for Layer 3 (L3)-Driven Fast Handover. IETF RFC 5184 (2008)