Transport protocols: limitations, evolution obstacles and solutions for an actual deployment in the Internet

Mohamed Oulmahdi[1,2] Christophe Chassot[1,2] Nicolas Van Wambeke[3]
[1]CNRS, LAAS, 7 avenue Colonel Roche F-31400 Toulouse, France
[2]Université de Toulouse, INSA, F-31400 Toulouse, France
[3]Thales Alenia Space, Toulouse, France

mohamed.oulmahdi@gmail.com, christophe.chassot@laas.fr, nicolas@vanwambeke.net

Abstract — The Transport layer, designed for old networking contexts and now obsolete applications requirements, is inefficient. This paper discusses the reasons behind this inefficiency and the obstacles to the evolution of Transport protocols. The discussion is then extended to derive new requirements for the Transport layer, both functional and architectural, in order to ensure optimal performances in all current and future contexts. To meet these new requirements, a novel architectural design of the Internet Transport layer is proposed following a service-oriented and a component-based approach. The proposed solution allows for optimization of the Transport service performance, facilitation of its utilization, and is aimed at allowing the integration of new services as needed.

1. Introduction

The Transport layer is one of the most important layers of the Internet communication protocol stack. Located between the application and the network layers, this layer is expected to take into account both the application feature / requirements and the underlying network capabilities / constraints in order to provide the best end-to-end communication service, matching as much as possible the application requirements, still taking into account the opportunities and limitations of the underlying network.

In the last few decades, several new applications and networking technologies have emerged with very different requirements and characteristics, making initial Transport-level protocols (typically TCP and UDP in the Internet) no longer adapted. As a consequence, a lot of new Transport protocols and mechanisms have been proposed over the last 20 years in order to enhance both the Transport services offered to applications, and to optimize the usage of the different network technologies.

Unfortunately, the factual observation is that all these new proposals are either not integrated in the actual operating systems (Windows, Linux, etc.), or hardly used in practice by the application developers, which in majority continue to use suboptimal TCP and UDP solutions.

Since it is proved that these new propositions are widely more efficient than the actual used protocols, the main goals of this work are to identify the deployment obstacles of the new Transport propositions, and then to propose a new architecture of the Transport layer as a solution.

In this context, the contributions of this article are threefold.

The first contribution of the paper (section 2) is to analyze this general problem by identifying the obstacles that justify the lack of evolution of the Transport protocols in spite of the need.

- The first part of this analysis deals with the design choices that explain the performance limitations of the current Transport protocols.
- The second part explains in what tackling these sources of performance issue is a complex problem due to several main obstacles that no new Transport solution is able to put off in a global way.
- This analysis leads finally to conclude that the solution to be promoted does not consist in developing a new concept of “perfect” Transport protocol but requires a revision of the current Transport layer architecture.

Following this novel approach, the second contribution of the paper (sections 3 and 4) is to provide a new design of the Transport layer, with the aim to allow and facilitate the integration of both existing and new Transport solutions, their updating, and their usage by the applications.

Finally, the impact of the new architecture on the performances is discussed in section 6.

2. Transport protocols: analysis and requirements towards an effective deployment of the news solutions in the Internet

To tackle the evolution of both the application features / requirements and the network capabilities / constraints, a huge amount of research works dealing with desirable evolution of Transport protocols have been and are still published for more than 20 years. One of the main goals of these contributions is to provide adaptation of the protocol services to new applications requirements using more efficient mechanisms taking into account the evolution of network technologies. However, and in spite of their well-known limited services and inefficient behaviors in some contexts, TCP and UDP still remain the *de facto* two main Transport protocols used to support the end-to-end data transfers for Internet distributed applications.

After a state of the art of this rich scientific matter (section 2.1), section (2.2) analyses the reasons explaining the lack of effective deployment of new Transport solutions in the Internet. On these bases, section (2.3) presents the positioning of our proposition together through a set of requirements that need to be satisfied with the aim to allow an effective deployment of new Transport protocols.

2.1. State of the art of the Internet Transport protocols

Sub-optimality, or "performance issue", of a Transport protocol may be observed:

- from the application point of view, with the absence of some required services, or the presence of non-adapted ones;
- from the network point of view, whose expectation is an as best as possible usage of its bandwidth resources.

On this basis, two main reasons explain the performance issues of an existing Transport solution:

- when the solution is to be used in a more exigent application context, i.e. when new applications have stronger (or more complex) requirements than the initial ones, such as real-time multimedia application that have stronger delay and general throughput requirements than classical web application;
- and/or when the existing solution is to be used in a less efficient network context than the initial one, such as wireless links that leads (among other) to higher bit error rates or more variable access throughput, comparing to wired networks.

In front of these performance issues, three main approaches have been and are still adopted in the design of Transport protocols:

- the first one consists in designing one can be called "generalist" protocols, aimed at supporting as many as possible applications and network contexts using (almost) the same mechanisms whatever the context. Obviously, the main weakness of this kind of protocols is due to the generality principle of their design, which inherently leads to suboptimal performances;
- the second approach consists in designing "specific" protocols, aimed at providing optimal performances for a given and precise application and/or network context. The main weakness of this kind of protocols is due to the specificity principle of their design, which inherently leads to the reduction of their deployment contexts; as a result, such solutions are difficult to consider for a large spectrum of application / network specificities;
- finally, the third approach consists in designing "adaptive" protocols, whose internal mechanisms and parameters may be configured depending on the targeted application and/or network context. The main weakness of this kind of "ideal" solution firstly comes: 1) from the "intelligence" that need to be acquired to decide and then deployed the adequate mechanisms between the end-to-end entities, and 2) from the cost of this intelligence that need to be subtracted from the expected benefits.

The following sections detail these three categories and then conclude on their effective deployment.

2.1.1 Generalist Transport protocols

Generalist Transport protocols have been designed with the aim to get acceptable performances in the context of an as large as possible set of applications distributed over an as large as possible set of network technologies.

The latest examples are the Internet Transport protocols TCP [22], UDP [23], SCTP [24], MPTCP [25], etc. which try to cover all Internet contexts in all their heterogeneity.

As several of these protocols have been designed considering the worst network case and the most stringent application requirements in terms of order and reliability, they induce useless treatments when the network provides better conditions or when the application has less stringent requirements, which leads ton suboptimal performances [1].

2.1.2. Specific Transport protocols

Obviously, a specific Transport protocol is expected to lead to optimal performances when applied in the context for which it has been designed.

However, it is not unusual that such solutions are not so efficient and are subject to several issues even in their dedicated context. This is due to two main reasons:

- the Transport solution is specific to either application or network but not the two together [2],
- the solution is not enough specific [3].

In the first case:

- the solution may, for instance, be dedicated to a given set of "similar" applications but it does not consider the differences between the possible underlying network technologies. So, even if the solution is specific to a given set of applications, it is still general from the network point of view. This affirmation may be illustrated through protocols that have been designed to take into account multimedia applications features / requirements, but which run in the same way whatever the underlying network technology. Consequently, they are not efficient in several technologies, like satellites links [7], ad-hoc networks [3] or sensors networks [4];
- similarly, a Transport protocol may have been designed specifically to cope with a/some given network technology(ies), like WLANs, but are inefficient for several specific types of applications like multimedia ones [5] [6].

In the second case, the Transport solution tries to be specific to a given application and/or network context like protocols designed to support multimedia applications in wireless networks. However, the solution may be not enough specific, either from the application point of view applications, for the network or for both application and network.

For instance:

- some solutions do not consider the different "preferences" that could be expressed by multimedia applications, where some ones may wish to prioritize bandwidth [12] while others may prefer prioritize delay [9];
- also, they do not consider the differences in wireless networks characteristics between LANs [10] or ad-hoc networks [11]. Therefore, the lack of specificity or in the degree of specificity makes specific solutions not optimal even if they are applied in their dedicated area.

2.1.3. Adaptive Transport protocols

Most of the existing Transport solutions may be considered as "monolithic", in this way that they are neither configurable nor adaptive, and run in the same way for all application and network contexts.

The "configurability" capability of Transport protocols is however a expectation that has been explored more than 20 years ago with the initial "partial order connection concept" proposed in [8], which introduced "order" and "reliability" as parameters of order / loss control mechanisms, aimed at taking into account multimedia application order / loss tolerance to reduce both end-to-end delay and bandwidth consumption.

More generally, the configurability of a Transport solution is its ability to adopt different behaviors for different application and / or network contexts. This property may concern both the mechanisms to be activated (congestion control, error control, etc.) and their parameter [26].

The configurability can be "static", i.e. defined at the design time of the protocol; in this case, the corresponding code needs to be recompiled each time a configuration change is required. It can also be "dynamic", for instance when it is done at the connection establishment time, or even better if the protocol behavior may be changed during the communication [14].

Finally, a dynamic configurability may be "autonomous" if the changes are operated transparently for the applications [13, 15]. In this ideal case, the protocol is said to be "adaptive".

As introduced in section (2.1), adaptive protocols require to be provided with richer monitoring, analysis, decisions and execution capabilities that the other protocols and it is still a challenge to demonstrate their actual benefits. However, it is conceivable to make the promotion of this kind of protocols as the best solution in order to tackle performances issues of current Transport protocols.

Let us however note that if some adaptive Transport protocols are considered by their authors to be fully extensible [13] [15], this extensibility is in fact limited. Indeed, a real extensibility should be based on a list components file like it is the case for several operating systems components, such as hardware drives. It is for instance possible to add as many drivers as needed without affecting the system or its complexity, because the system manages a file in which it simply adds an entry every time a new driver is added. The architecture of current adaptive protocols has not yet such capacities and it is then a still unsolved problem to add new mechanisms "on the fly" in the set of available ones.

2.1.4. Conclusion

The previous state of the art was aimed at analyzing in what the design choices of Transport protocols have a non-negligible impact of the performances issue.

Next Table 1 summarizes the different categories that have been exhibited together with their main features in terms of performances and deployment spectrum (i.e. application and network context).

	Performance	Covered context	References
Generalist protocols	Non optimal	Large	[22][23][24][25]
Specific protocols	Optimal	Small (specific)	[6][7][9][11][12]
Adaptive protocols	Optimal	Large	[2][15][26]

Table 1: Categories of Transport protocols

In spite of their *a priori* appropriate design that allows encompassing both general and specific protocols, adaptive protocols are very few deployed in the Internet, and it is often a lost fight for their designers or implementers to convince the operating system developers of the opportunity to integrate them in their systems. The next section (2.2) provides a deep analysis of the difficulty.

2.2. Obstacles to the effective deployment of new Transport protocols in the Internet

Several obstacles may be identified that explain the lack of effective deployment of new Transport solutions in the Internet. This section provides an analysis of the context that leads to exhibit those obstacles from the points of view of the different concerned actors.

2.2.1. Points of view and expectations

Three types of actors have to be considered in the analysis of the problem: the application developer, the protocol developer, and the operating system developer, which have different expectations that need to be addressed.

The application developer is concerned by the usability of the service and by the compliance of the service to the application needs. The developer has not to know the complexity of the overall set of possible solutions. Moreover, it is not realistic to imagine a developer rewiring its application each time a novel Transport solution appears on the market.

The protocol developer is concerned by the acceptation of a new solution or the improvement of an existing one. As a result, once a new solution has been developed, this one must be promoted:

- from the application developers, with the aim to convince them of the added-value of using the new solution;
- from the operating systems developers, with the aim to convince them of the added-value of integrating the solution into their system;

Finally, the system developer is concerned by the ease of integrating a new solution in its system or to update an existing solution. His/her preoccupation is to minimize code rewriting for each added or updated operation.

2.2.2. Obstacles that need to be tackled

Based on the evolution of the application, network and terminal context, and taking into account the conclusions of the state of the art presented in section (2.1), five problematic points may be exhibited as obstacles to the actual deployment of the new Transport solution in the Internet. These obstacles may concern one or several of the previous identified actors.

a) Problem of complexity

The complexity of the current protocol solutions may lead to the exhibition of three problems:

- Transport protocols are quite complex by nature because of their operating principles, the techniques on which their mechanisms are based, and finally their algorithms. This complexity has a direct impact on the offered service, which required to be known to tackle the applications requirements. For example, a window-based congestion control mechanism, like the one applied in the basic version of TCP, induces both throughput and delay variations that are incompatible with the need in constant throughput and bounded delay of interactive multimedia applications. As a result, in front of this complexity, an application developer should have a detailed knowledge of the behavior of the underlying protocol before developing their applications. Clearly, this requires a detailed knowledge for developers whose Transport protocols is not the area of expertise.
- Transport protocols are also complex in their use. Indeed, the use of their service is done via an application programming interface (API), which generally requires the mastery of knowledge in several areas, including system programming. As a result, an application developer should know several API it he/her wanted to benefit from the different possible solutions. This is clearly not desirable from an application developer point of view, which

certainly would prefer having to learn a single API, for all the possible underlying solutions.

- Finally, the proliferation of proposals for Transport protocols and APIs confronts the application developer to a third difficulty, which is the choice of the best solution tailored to its application.

b) Problem of dependency of the application to the invoked protocol

The complexity of the protocol selection is not the only problem resulting from the evolution of the possible solutions. Indeed, this choice is made in design time, i.e. at the time of the application design. This means that the application is written for a given protocol and has to use its own service during all its life cycle.

The resulting problem comes from the fact that the chosen protocol cannot be adapted to evolving application requirements (or when network constraints change), except in rewriting the code of the application. Note that this problem does not only concern the new protocol solutions, but also any migration from a given application to another existing Transport protocol.

This dependency of the application to the underlying protocol is also partly responsible for the problem of scalability and deployment described hereafter.

c) Problem of extensibility of the existing solutions

The extensibility of the existing solutions is a crucial issue as it raises several problems depending on the adopted point of view:

- The integration of a new solution, or the updating of an existing solution, is a problem for the operating systems developers. Indeed, any protocol modification requires changes in the hosting systems, dealing with their integration in the new versions of the system, the updating of the earlier versions or the management of the compatibility between both. These implications explain the reluctance of the operating systems developers with regard to the adoption of new solutions; this also explains why the system environments are heterogeneous in terms of supported Transport solutions.
- From the perspective of the protocol developers, the previous statement is a real obstacle given the high probability of non-acceptance of new solutions by the systems, or in the best case, given the significant delays in the integration of their solution (usually several years).
- Finally, from the perspective of the application developers, these heterogeneous environments lead themselves to be careful when using new solutions, nobody being ready to accept that its application has operational problems for the simple reason that it runs in a system that does not support the desired Transport protocol.

In summary, the application developers are waiting for wide deployment of the solutions before using them, while at the same time, the systems developers are waiting for a use of these solutions by the application developers. This vicious circle constitutes the main cause of the deployment problem described below.

d) Deployment problem (of new protocol solutions)

Deployment of new Transport solutions is the most restrictive problem in the evolution of the Transport layer. It includes the issues of extensibility and protocol dependency in addition to a crucial problem: the acceptability of the solution by the network.

Indeed, middleboxes, whatever their functionalities, often deploy operating policies based on protocol types and their content. Some of them allow the transmission of only certain protocols (generally TCP and UDP), and a new protocol might not be able to cross through a middleboxe even if it is supported by the end systems. Any modification of an existing protocol or the integration of a new protocol requires updating these middleboxes.

e) Configurability problem

With the evolution of applications requirements, the "monolithic" services offered by TCP and UDP have become inadequate.

The lack of configurability of these protocols has several impacts. First, the applications are obliged to use the overall set of services providing for instance by TCP (reliable service, ordered service, etc.) inducing both useless data transfers and process time, but also in certain case a degradation of the targeted services (a reliable service has a cost that can expressed in term of additional delay from the application point of view).

The non-configurability of such protocols, which makes that applications are obliged to use the full set of services, explains the proliferation of new protocols offering no new services, but only some subsets of services existing in the current protocols.

2.3. Proposition and requirements for an effective deployment of Transport protocols

Based on the previous analysis, the approach proposed in this article is not to define a new protocol. Differently, it is to take advantage of all existing protocol solutions within a novel architecture for the

Transport layer, in order to meet as best as possible the applications requirements still taking into account the capabilities of the network and end-systems.

The architecture must be general enough to be used by any application without any source code adaptation. However, this generality must be achieved not by a single solution having the same behavior in any context, but by an adaptive solution that can be general and specific in the same time: general by the capability to be used by any application and to be run over any network technology, and specific by its capability to adopt a specific behavior depending on the supported application features/requirements and the underlying network capabilities/constraints.

Also, the architecture must be easily extensible, without or with minor implications for operating and network systems, this in order to be capable to integrate any new Transport solution. This extensibility will ensure the evolution of the Transport layer as well as applications and networks evolution, and motivate developers to continuously update and create more adapted Transport solutions.

The integration of new solutions must have no effect not only for systems, but also for the existing and future applications. The proposed design for the Transport layer must adopt a loose-coupling software architecture paradigm to separate applications from technical details of Transport solutions. By this, architecture extensions, either by addition of new solutions or by updating of existing ones, will be transparent for the applications, which then could be distributed over any solution offering the same service.

Without any systems or applications impact, a Transport architecture will then ensure its extensibility, and consequently will be adaptable to any evolution of applications requirements or networks technologies. It is also thanks to those conditions that new Transport protocols would have the possibility to be integrated in the operating systems.

Another requirement is related to the “acceptability” part of the protocol deployment problem. This problem results from the independent management of Transport protocols in systems. The only way to resolve this issue is to apply a global management for all protocols and then to impose a congestion control for each Transport connection. When access networks managers will be sure that there is no equity problem in the bandwidth sharing, they will allow any protocol to transit in their network.

Finally, the targeted Transport architecture must provide a better programming interface to application developers in order to not impose them to master all related technical considerations, especially for ones that are not specialized in system and programming area. Therefore, the architecture must have an easy access interface, limiting the interaction with applications to high-level descriptions.

Next Figure 1 summarizes the sources of performance issues and the corresponding architectural obstacles that limit the deployment of new Transport protocols.

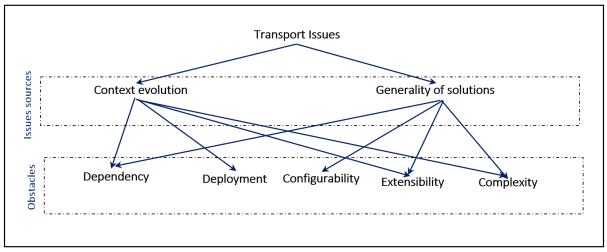


Figure 1: Issues sources and architectural obstacles

3. Service-oriented and component-based architecture

Section 3 and 4 present our proposal for a new architecture of the Transport layer. In a first time (section 3), we illustrate the used design paradigms and how they allow tackling the limitations of the current architecture of the Transport layer. Next section 4 describes the complete architecture of our proposition.

3.1 Description of the approaches

Our architecture is service-oriented and component based. This association of these two paradigms should offer all what the Transport layer needs to ensure its evolution, regarding itself, applications and networks.

The service-oriented approach has been created in order to remove the dependency between heterogeneous, and eventually independent systems when must interact to offer or get services. Its principle is to limit the interactions with external systems to a set of services, and to abstract all the details concerning how these services are really realized. The service approach offers an independency between systems, which may communicate with any system offering the same services, without any modification. In addition, an internal modification in a given system will not affect the other ones, as long as its interface is not changed.

Contrary to the service-oriented approach, the component-based one is related to the internal architecture of the system. It consists in dividing the global system in a set of relatively simple subsystems. The aim is multiple. It concerns first the simplicity of the design process, by allowing a progressive

development by components. Also, the component-based approach facilitates the detection and the resolution of bugs, and the maintenance and the extensibility of the system.

Because the Transport layer issues deal with both the Transport layer itself, the applications and the network, we combine these two paradigms in our proposed architecture in order to resolve the whole of issues. We describe in the next section the capabilities offered by these approaches and the corresponding tackled issues.

3.2 New architecture capabilities

The service-oriented approach concerns the interface between applications and the Transport layer. By this, it allows changing integrally the interaction way between them and limits it to the invocation of "services" instead of protocols. For their part, protocols will be defined by independent services evocable separately as required, following a component-based approach. The next sections list the capabilities offered by these two approaches, where each one allows the resolution of a part of the global issue (described in section 2.2.2).

3.2.1. Abstraction

The interface we propose is completely different from those offered by classical Transport API. It is no longer based on primitives, but on services, and the Transport layer is no longer shown by applications as a set of protocols but as a set of services. The way these services are implemented internally by the Transport layer is completely transparent for applications, which will have to request only these services (with eventual parameters) and it is to the Transport layer to choose the optimal way to offer them[1]. The abstraction principle concerning the Transport mechanisms is currently under study in a dedicated working group of the IETF (TAPS). However, the group objectives are currently limited to the specification of a set of standardized Transport services.

Such high level of abstraction, limiting the interaction between the applications and the Transport layer will have several benefits, and allow the resolution of a number of issues.

First, this remove the dependency between applications and their protocols interfaces, and consequently remove all issues resulting from the dependency obstacle. The applications may now run on any environment, whatever the Transport mechanism used inside, as long as it offers the desired services. This increases the portability of applications, and the need to rewrite them for specific environments, or to support new capabilities. This will be very useful for the autonomic part of the architecture we propose, by selecting the best mechanism based on the network characteristics, since the offered service is the same.

In addition, this transparency of internal implementation of services makes naturally all internal modifications also transparent for applications, as longer as the services interfaces stay the same. It is now possible to add or update Transport components without taking account applications.

The last benefit is about the complexity resulted from the increasing of the number of existing Transport solution (the first part of the complexity issue), and the impossibility of the application developer to have enough knowledge to choose the most adapted solution to their applications. Now, developers have only to express their needs in terms of Transport services, without any knowledge about the underlying mechanisms. Also, the service-oriented interface will be easier to use than the current sockets. This will increase the productivity, and reduce the needed competency to develop with the Transport layer.

The abstraction is the most important capability offered by the service-oriented approach. In fact, it allows the resolution of almost all issues involving applications. The other capability is the configurability, offered in combination with the component-based approach, is presented in the next. The rest of capabilities concerns the internal issues of the Transport layer, and therefore, will be resolved by the component approach.

3.2.2. Extensibility

We consider the extensibility as the capability of a system to make modifications (adding, updating, …) of its extensible components without impact on the rest of the system. The lack in the extensibility is mainly caused by the high dependency on the different parts of a system, resulting of only on or too few modules. Thus, the modification of any part needs to recompile the entire code source of the system. For Transport protocols, the problem is more important, because their implementation in the kernel.

The component-based approach has as principal to separate a complex system to a set of independent, and relatively simple, subsystems. This independency concerns not the source files, but the resulting compiled, generally as shared libraries. Each part (component or module) is compiled separately and therefore can be maintained independently of the others. The granularity of the decomposition of the systems

[1] Details about services and their implementation will be described in section 4

depends on the context of the system. In our architecture, we consider a component for a service. Each service is realized by an entire component. It is possible that a given service is offered by several types of components, but each one offer the entire service, and only one can be used at the same time.

The architecture is based on a list of components that can be loaded dynamically. All the components offer the same common interface, allowing the Transport layer to be capable to use them without having to know them previously. It must just through the list of components, and load those are necessary according to the requested services.

3.2.3. Reusability

Most of new protocols, and especially new versions of existing ones, propose some little differences comparing the previous version. However, because of the monolithic aspect of their architecture, we need a complete new protocol, even the difference is minimal. The component based approach offer the possibility to reuse the existing components, and in this case, have only the different components of the new version. This reduces the complexity of the system, the development time, and the system usage.

3.2.4. Scalability

The other part of the complexity (concerning the overload of operating system) comes mainly from the fact that the whole of the protocol is implemented in the kernel, and directly linked to the system. By a component approach, this link will concern only the elements that may interact with the system; the other ones will be implemented independently of it. This means that it becomes possible to have as much as needed of Transport services and mechanisms, since their only cost will be limited to the memory space occupied by the components modules files.

3.2.5. Configurability and adaptability

The other capability offered by the component-based approach is the possibility to choose an optimal set of services. In fact, for the classical monolithic architectures, the applications are forced to use the whole of the service offered by the underlying protocol. By decomposing the protocols in elementary services, it becomes possible to choose any subset from them, and offer to the application only what it needs, optimizing therefore the performances.

In addition, these independent services can be offered by different ways, each one corresponds to a different component, and adapted to a specific network context. Thus, the Transport layer can choose different components to offer the same service following the network state.

The configurability starts before all by a custom request of the application, by a specific list of services. These capabilities are offered by the association service/component, which makes it very easy to translate a list of requested services to a list of corresponding components.

Next Table 2 summarizes the pertinence of the service-oriented and component-based architectural approaches in front of the previous identified issues.

Approach / Issue	Complexity	Configurability	Dependency	Extensibility
Service-oriented	x	x	x	
Component-based	x	x		x

Table 2: Architectural approaches and corresponding issues

4 Description of the architecture

This section describes the main principles of the proposed Transport layer architecture. The design proposes modification of the foundations of classical approaches, where a set of independent protocols composes the Transport layer. In the proposed solution, a management layer is added. This additional component to the architecture is used to coordinate connections by creating, modifying, and closing them. For each connection, the choice of the protocol mechanisms and parameters is done in order to best fit the applications needs while also taking into account the network state at the time of decision as well as its possible evolutions.

The management component is instantiated as a singleton object, by such it is common to all applications and connections on a given endpoint. It is independent from connections, and interacts with them through a unique standard interface. When a connection is created, the management component can modify its composition (i.e. in response to an application request, or a change in network state). The interaction between the applications and their associated connections for sending and receiving data is made directly without implication of the management component.

The architecture is illustrated in Figure 2. The next sections provide further details of each component.

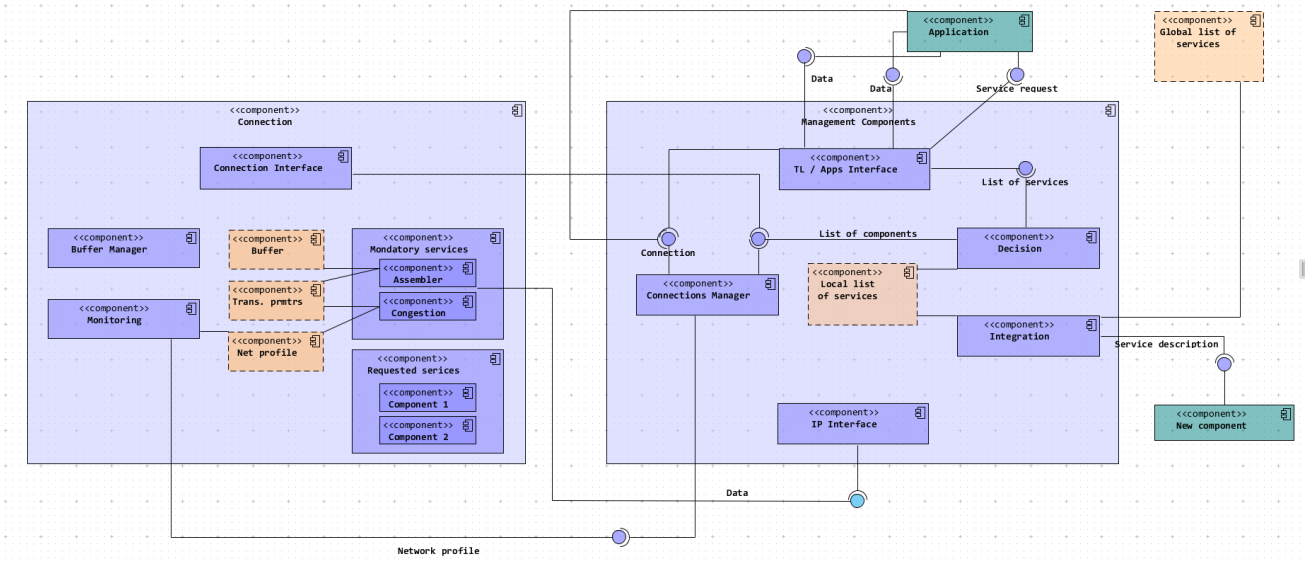


Figure 2: Components diagram of the architecture (common components on the right, connection components on the left)

4.1 Management (Common) components

The management components (right part of Figure 2) are responsible for processing applications requests by selecting the most appropriate components and instating them in the connection. In addition to this feature, the management components manage the integration of new components into the architecture. Moreover, in order to ensure fairness among multiple connections in their access to the network layer, the management components perform a round robin scheduling of the connection buffers in their interface with the network layer. The Transport layer manages two databases in which it stores information about the services offered to the applications and the components offering them.

4.1.1 Databases

Global services database: a list of available services provided to the applications developers. This database is globally unique (i.e. common for all systems and users). The database items are described as abstracted services without any information about the component implementations offering these services.

Local services database: Opposite to the global database, this database is private and available only on a given system. It stores the correspondence between services and the component implementations offering them. This database is used by the decision component to satisfy application requests. The construction and maintenance of the database is ensured by the integration component.

Such paradigm de-correlating an abstract description of services from the components implementing those services is a current practice in the service-oriented architecture. The two databases contribute to ensure the abstraction, the extensibility, the configurability and the adaptability of the Transport layer.

4.1.2 Software components

The software components are responsible of managing services, from their integration in the Transport layer, to their instantiation in response to the applications requests taking into account the network state. In what follows, we will detail the different software components that compose the architecture and, for each of them, identify the capabilities (identified in section 2.2.2) that these components provide. Furthermore, the optional nature of some of the components contributes to the adaptation capability identified earlier in the sense that, when not required (i.e. for complexity or implementation reasons), the modularity of the proposal allows for their removal.

Integration Component: ensures the extensibility and scalability of the Transport layer. This component is responsible of integrating new Transport services into the endpoint system after checking the integrity of their implementation and ensuring the reliability of their source. Indeed, only normalized and “manufacturer or community approved” service components implementation can be added to the Transport layer. Following integration, the local database is updated by adding in entry for the new component.

Applications interface: this component serves two functions. It is an interface between application and the Transport layer as well as a relay between the other components. It contributes to the abstraction of the services offered by the Transport layer and the configurability through the decision component. For applications, the applications interface is responsible to process their service requests. It communicates the necessary components in order to create a connection meeting the requirements of the application. It is also possible for applications to modify the service request during communication. In the connection establishment process, the application interface manages the interactions between the components.

Decision Component: ensure extensibility in conjunction with the integration component as well as adaptability through the monitoring component, and configurability through the applications interface component. The role of this component is the automatic selection of the associated Transport services component implementations for a given service request. To achieve this objective, it bases decision on the local services database in which it searches for the best suited component for each requested service taking into account the current network state. It is used by the applications interface when creating the connection or when changing the service, and by the connections when a change in network profile is detected.

Connections manager: responsible of the creation and the management of the different connections. It is used by the applications interface during connection establishment. The connections manager creates connections for the applications using the Transport layer and maintains a connections table. During a connection, the manager can be used to implement changes to connections either following an application request or changes of the network profile.

Network interface: This component has two roles; it offers an abstraction of the Network layer to the different connections (multiplexing and de-multiplexing) and it ensures a fair sharing of the local link bandwidth between the different connections.

4.2 Connections components

Once a connection is created, the applications directly interact with it for data transmission. For management interactions that might be required during communication, these are performed via the application interface.

Connections components (left part of Figure 2) are responsible for the effective data transmission. They are specific for each connection, and consequently instantiated every time an application requests a service.

4.2.1 Shared memories

The shared variables are used by different components in order to exchange information.

Buffer: temporary store for the data to be transmitted. It is used by the connection interface to set or get data to and from applications, and by the assembler component to set or get data from and to network interface.

Transmission parameters: These variables are used by the assembler component to manage the data flow, capturing the state and state changes of some Transport service components implementation. As an example, the "throughput" parameter may be determined by a component like the congestion control. The list of such parameters is extensible, and the available ones depend on the Transport components that are deployed for a given connection.

Network profile: This variable is used by the monitoring component in order to detect variations in the network state. It contains the values monitored by Transport services. The list of values to be monitored is also extensible and depends on the components present in this connection.

4.2.2 Software components

There are two types of components. Some ones are used for the internal management of the connection, and some others are used to the effective transmission, i.e. they implement Transport services.

Connection interface: constitutes the interface, on one side with the application and with Transport layer on the other side. This interface is divided in data and control parts. The data part is dedicated to an application. Data transmission is the only direct interaction between the application and its connections. The control part is dedicated to the Transport layer (via the application interface). The connection uses this interface to signal network profile changes. The Transport layer uses this interface to adapt the connection service components when needed.

Buffer manager: manages access to the buffer by the connection interface and the communication components. This choice is motivated by the existence of several components that can decide on packets discarding from the buffer (e.g. a reliability component). The management will coordinate the discarding of packets among the involved services implementations. Access to the buffer is exclusively reserved to the buffer manager, and any operation by

the other component must be performed through of the buffer manager.

Monitoring: contributes to the adaptability together with the decision component. The monitoring component permanently checks and detects network states variations to determine profiles. Upon profile change, these are transmitted to the Transport layer for reaction (i.e. possible change in service composition). The monitoring component is not used to detect the different network parameters (bitrate, delay, …). These parameters are computed and published by the Transmission components (e.g. the reliability component for the loss rate parameter). The monitoring component only analyzes these parameters in order to detect eventual variations.

Transport components: they are the components that are responsible of the effective transmission and implement the requested services by the application. They constitute the dynamic part of the connections. Indeed, their instantiation depends on the requested services and the network state. There are two types of components: mandatory and optional. The mandatory ones are services that the Transport layer offers independently of the application requirements. Nevertheless, it is possible that the components implementation for mandatory service varies depending on the network profile; however, they are always offered by the system. These components are mainly: the assembler component (creates data packets and transmit them to each of the components implementation before reaching the network interface).

Table 3 summarizes the Transport layer components classified based on their respectively provided capability.

Component / Capability	Abstraction	Configurability	Reusability	Extensibility	Scalability
Interface	x	x			
Decision		x	x	x	
Integration			x	x	x
Monitoring		x			

Table 3: Transport layer components and corresponding issues

4.3 Dynamic description

In this part we describe the important behavioral aspects of our architecture (Figure 4).

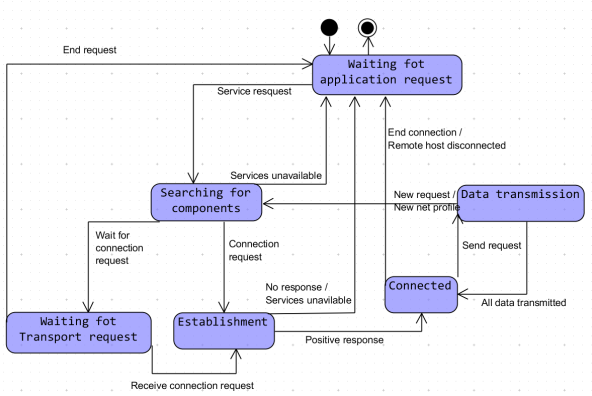


Figure 3: Description of the Transport layer behavior

The process starts in listening state, waiting for an application request. Request are composed of a list of desired services, and eventually with their parameters, in addition to the classical parameter necessary for the Transport layer, like the remote address or the application type. Given the extensible nature of the architecture, it is possible that some services are not available on a given system. To handle this case and to avoid long negotiation processes in case of unavailability, the list of requested services is divided into two parts: mandatory and optional services. The Transport layer can accept the request even if some of the optional services are not available. But for the mandatory ones, it must be available for the connection to be established. The same rule is applied in the server side; the connection request is accepted only if all mandatory services are available.

When the application interface receives the connection, it extracts the list of services and transmits them to the decision component, which in turn, searches the local database of services to determine the available components implementations offering the requested services. For the first use of the decision component, i.e. before the establishment of the connection, it considers only the application requirements, because the network state is not yet known. In case of more than one component offer a given service, the decision component chooses the default one. The initial choice can be further optimized after connection establishment. This choice refinement is activated by the detection of a new network profile by the monitoring component, or when the application requirements change.

Once the Application Interface component has verified the local availability of the services requested by the application, the same process is performed on the remote host. Once the availability issue is resolved, the Application Interface component invokes the Connections Manager component, which creates a new

connection by instantiating the corresponding service component implementations. A reference to this connection is returned to the application to communicate directly with it (without passing by the application interface). However, this communication is limited to the data transmission (send and receive data). If the application wants to change its services, or end the connection, it requests this to the application interface. In this case, the procedure is similar to the first request. However, it differs from this initial procedure in the sense that the decision is optimized by taking into account the network state based on the current network profile provided by the Monitoring component.

In the connection, for each packet of the application, the assembler component calls all the instantiated Transport service components implementation. These components can transform the data (e.g. compression), add header fields to the packet (e.g. sequence number), or both. Once the process is over, the assembler transmits the packet to the network interface responsible of its effective transmission in the network. At the remote side, the assembler follows the reverse process by passing the packet to all service components implementations in the reverse order.

The Transport service components implementations are responsible for the monitoring of the different values of the network, and write them in the network profile. The monitoring component subscribes to these values changes. When a profile change is detected, it is transmitted to the Transport layer, which can eventually decide to change the current service composition.

The next section discusses of the benefits and additional cost that are obtained and caused by the new architecture.

5 Architecture benefits and costs

Since the architecture proposal is in a conceptual state, this section discusses theoretically the expected benefits and the additional work that should cause supplementary cost comparing to the current architecture.

5.1 Benefits

Our proposition is based on the principle that specific Transport solutions are more efficient than general ones. This fact has been demonstrated many times though multiple enhancements for TCP and other protocols for specific application and/or network contexts [2]. These works demonstrated that these specific versions lead to important performances enhancement, both in terms of network bandwidth occupation and of (functional and non functional) services offered to the applications. In addition, some of these works offer new capabilities (such as the partially reliable error control), which are not available or cannot be integrated on general solutions. The problem comes from the difficulty, on one hand, to integrate all the solutions on one system, and on the other hand, to affect (automatically) each solution to its dedicated context.

Thus, we assume in our proposition, that if we offer an environment where all the specific solutions can be deployed and dedicated to the appropriate applications and networks, we will obtain optimal performances. Therefore, the first benefit of our proposition is to allow optimal Transport solutions to be integrated and correctly used.

Another benefit, related to the existence of as much solutions is in their usage by applications. The abstraction layer of our architecture makes it transparent for the applications all these details, and the usage of the Transport layer becomes not only easy, but easier than when it was composed by few solutions.

The last benefit, that constitutes one of our main objectives, is to guarantee the continuous evolution of the Transport layer. This is allowed by the extensibility capability, which offers the capability to integrate or update any services, without any impact to the Transport layer; moreover, this evolution is also allowed by the independency between the applications and the Transport components, thanks to the service approach used in the interface exposed to the applications.

5.2 Additional cost

The additional cost in terms of network bandwidth occupation, processing load or additional delay, should appear through the implementation of the functionalities ensuring the new capabilities of the architecture. This is expected, a priori, in the connection establishment, the decision and the monitoring processes. For the other functioning modes, like for the instantiation, or the composition of services, they will concretely be similar to the actual protocols modes, and will not have additional cost.

For the connection establishment process, the additional cost is associated to the list of services that have to be transmitted to the other host for the availability verification. The remote host, in case of unavailability of some optional services, will have to transmit the list of available ones. Taking into account the number of services (NB: referred by single identifiers) that can be associated to the applications, the additional cost including the information containing this list of services

will be very small. In addition, taking into account the optimization that this negotiation will offer after the connection established, and taking also into account the fact that this procedure is used only one time, we assume that this additional cost is negligible.

As far as the monitoring functionality is concerned, as described below, it is realized by the Transport components. Thus, instead to send additional traffic in the network in order to measure some values that are already known by the Transport components (like the loss rate thanks to the reliability component), the architecture includes a mechanism allowing the different Transport components to share their information with the monitoring component, avoiding additional traffic. The function of the monitoring component is therefore limited to checking this shared information with the aim to detect an eventual network state variation. Thus, its work is completely independent of the communication process, and will not induce any additional cost.

Finally, and as far as the decision process is concerned, it is clear that its cost is in processing and not in communication. We adopt for the decision a utility function approach, which calculates a utility value for each component and compare them to those of the current network profile with the aim to see the one offering the best correspondence. This problem is then be reduced to a classical maximization problem that can be resolved in a linear time regarding the number of available components. Taking into account this number, and the linearity of the decision algorithm, the decision process is then expected to be achieved in a negligible time.

The future implementation and experimentation works of the architecture are aimed at supporting these theoretical estimations of the additional cost. However, we believe that, taking into account the actual services and performances offered by the Transport protocols, and because of the limitations of the current Transport architecture, the additional cost will be negligible comparing to the enhancements that our proposal will be able to offer.

6 Conclusion

Transport protocols are an important piece of the Internet communication stack as it allows linking as best as possible the application level with the network one. Performance of such this conceptual link is a crucial issue, which is addressed for a long time. However, the fact is that both the application and the operating system developers continue to make use of suboptimal solutions.

The goal of this paper was to analysis this problematic and then to propose a novel design of the Transport layer aimed at allowing an actual integration of all the Transport solutions. After having identified the obstacles that need to be tackled, we studied how the application of service-oriented and component-based approaches allows leading to the design of such a new Transport layer. We showed that the service-oriented approach, in association with the component-based one, can effectively resolve the whole of the current Transport layer problems and limitations, with a minimal expected additionally cost.

The perspectives of this work are to formalize the proposed design for this new architecture of the Transport layer, and to demonstrate its benefits thanks to appropriate simulations.

7. References

[1] Chanson, S.T.; Ravindran, K.; Robinson, J., "The design and tuning of a transport protocol for local area networks," *Networks: Evolution or Revolution, Proceedings.* pp.631, 640, 27-31 Mar 1988.

[2] Schmidt, D.C.; Box, D.F.; Suda, T., "ADAPTIVE: a flexible and adaptive transport system architecture to support lightweight protocols for multimedia applications on high-speed networks," High-Performance Distributed Computing, pp.174, 186, 9-11 Sep 1992

[3] Henriksen, E.; Aas, G.; Rydningen, J.B., "A transport protocol supporting multicast file transfer over satellite links," Computers and Communications, pp.590, 596, 1-3 April 1992

[4] Zhenghua Fu; Xiaoqiao Meng; Songwu Lu, "A transport protocol for supporting multimedia streaming in mobile ad hoc networks," vol.21, no.10, pp.1615, 1626, Dec. 2003

[5] Shihada, B.; Pin-Han Ho, "Transport control protocol in optical burst switched networks: issues, solutions, and challenges," Communications Surveys & Tutorials, IEEE , vol.10, no.2, pp.70, 86, Second Quarter 2008

[6] Eshak, N.; Baba, M.D., "Design a new transport protocol (wireless TCP) to support mobility for mobile ad hoc networks," Telecommunication Technology, pp.144, 147, 14-15 Jan. 2003

[7] Anand, B.; Sebastian, J.; Soh Yu Ming; Ananda, A.L.; Mun Choon Chan; Balan, R.K., "PGTP: Power aware game transport protocol for multi-player mobile games," Communications and Signal Processing (ICCSP), pp.399, 404, 10-12 Feb. 2011

[8] Amer P, Chassot C, Connolly C, Conrad P, Diaz M. Partial Order Transport Service for Multimedia and

other Applications. IEEE/ACM Transactions on Networking, vol. 2, n° 5, October 1994.

[9] Ragothaman, V.; Baloch, F.; Pendse, R., "Transport of flight critical data over Internet protocol," Digital Avionics Systems Conference, 2005. DASC 2005. pp. 1-8, Vol. 1, 30 Oct.-3 Nov. 2005

[10] Hayashida, Y.; Baba, C.; Komatsu, M., "Stop-and-selective repeat ARQ scheme for a high-speed transport protocol," Networks, 1995. pp.101, 105, 3-7 Jul 1995

[11] Space Communications Protocol Specification (SCPS)—Transport Protocol (SCPS-TP),” October 2006.

[12] Byung-Seok Kang; Kwangcheol Shin; In-Young Ko; Pyoung-Yun Kim, "A Transport Protocol for Multimedia Transmission in Overlay Networks," Advanced Information Networking and Applications (AINA), pp.669, 674, 20-23 April 2010

[13] Young-Jin Kim; Kolesnikov, V.; Hongseok Kim; Thottan, M., "SSTP: A scalable and secure transport protocol for smart grid data collection," Smart Grid Communications (SmartGridComm), pp.161, 166, 17-20 Oct. 2011

[14] Linder, L.; Miloucheva, I.; Clausen, H.D., "A forward error correction based multicast transport protocol for multimedia applications in satellite environments," Performance, Computing, and Communications Conference, pp.419, 425, 5-7 Feb 1997

[15] Van Wambeke, N.; Armando, F.; Chassot, C.; Exposito, E., "Architecture and Models for Self-Adaptability of Transport Protocols," Advanced Information Networking and Applications Workshops, 2007, pp.977, 982, 21-23 May 2007

[16] T. W. Paper, “Service Oriented Architecture (SOA) and Specialized Messaging Patterns,” pp. 1–15.

[17] J. Ga, “SOA Background Concepts”, jboss Red Hat Division, 2008

[18] I. Member, I. Member, and N. Grumman, “Reference Architecture Foundation for Service Oriented Architecture Version 1. 0,” no. December 2012.

[19] S. Kinder, “IBM ’ s SOA Foundation,” pp. 1–68, 2005.

[20] P. Bianco, “Evaluating a Service-Oriented Architecture,” no. September 2007.

[21] C. Chi, Z. Feng, Y. Xue, H. Cai, and P. Zhang, “Component-based Protocol Stack Management for Reconfigurable Systems,” pp. 26

[22] IETF, “Transmission Control Protocol”, RFC 793, 1981

[23] IETF, “User Datagram Protocol”, RFC 768, 1980

[24] IETF, “Stream Control Transmission Protocol”, RFC 3286, 2002

[25] IETF, “Multipath Transmission Control Protocol”, RFC 6824, 2013

[26] Bridges, P.G. ; Wong, G.T. ; Hiltunen, M. ; Schlichting, R.D, "A Configurable and Extensible Transport Protocol", IEEE/ACM Transactions on Networking, Vol:15, 2007