UML-based Modeling Entity Title Architecture (ETArch) Protocols

Diego Alves da Silva,
Natal Vieira de Souza Neto,
Flávio de Oliveira Silva
and Pedro Frosi Rosa

Faculty of Computing
Federal University of Uberlândia
Uberlândia, MG, Brazil
Email: diegoalves@cti.ufu.br,
natal@mestrado.ufu.br,
flavio@facom.ufu.br,
pfrosi@ufu.br

Michel dos Santos Soares

Faculty of Computing
Federal University of Sergipe
São Cristóvão, SE, Brazil
Email: mics.soares@gmail.com

***Abstract*—The approaches used to model communication protocols suffered several changes in past years. Some of the modeling languages are not used anymore because of their complexity, others because of their inherent limitations that were pointed out over time. Even today an approach that can represent a protocol in many abstraction levels is welcome. The objective of this article is to introduce an approach to model a communication protocol using the Unified Modeling Language (UML). The purpose is to create models that are able to represent an Internet architecture in many abstractions levels and different concerns, including structural level and services definitions. Besides, we propose an evaluation of the generated model, showing main advantages, such as representing architectural modeling, and limitations, such as representing time, non-functional constraints and physical resources when modeling communication protocols using UML.**

***Keywords*-UML; protocols; architectural modeling**

I. Introduction

The design and development of a real-time communication protocols must ensure security, reliability and response time capability. In other words, the protocol must not reach unsafe or not allowed states, must forecast all possible states and must comply with time constraints. With the purpose of getting a high abstraction level of states and message exchanging between them, and also to allow the validation of the required properties, different modeling languages were considered for modeling real-time protocols in past years, i.e., State Machines [1][2], Petri Nets [3], and LOTOS [4].

The mentioned languages have the same modeling purpose, they are all focused on modeling behaviour, but with little focus on the structural and architectural elements, i.e., the Petri Nets language is fully visual; however it does not have natively ways of modeling time constraints and architectural representations. Then, in order to solve issues such as time constraints, customizations of the language as Time Petri Nets and Coloured Petri Nets [5] were created. Another modeling language, the Language of Temporal Ordering Specification (LOTOS), is tightly algebraic and has a specification with complex symbology, which may be one of the causes why the language is not adopted as default language for protocol modeling [6].

The International Organization for Standardization(ISO)/ Open Systems Interconnection (OSI) reference model for communication has several problems [7][8]. These issues are at the network layer, mainly related to the Transmission Control Protocol (TCP)/ Internet Protocol (IP) model. This model basically consists of five layers: application, transport, network, link and physical. Each layer is responsible for ensuring a portion of the service, and does not guarantee that it is sent to the backsheet to solve this. Much of the services are made in the application layer, but some could be made in layers over the network core, such as the transportation and network layers [9]. This paper presents the modeling of a Future Internet protocol architecture using the UML modeling language, as well as standards and an approach to model elements and behaviours in different abstraction levels.

The reminder of the paper is as follows. In Section 2, an overview of related works about protocol modeling is described. In Section 3, the architecture and the services of the proposed Etarch protocol are presented. In Section 4, the modeling of services provided by the protocol is described using the UML modeling language. In Section 5, the modeling is evaluated and the advantages and limitation are discussed.

II. Related Works

Finite State Machine (FSM) consists of a mathematical model of computing. The FSM have an alphabet of input and output, states, and transitions that connect states. With a finite number of states, its main feature is determinism. Modeling communication protocols using State Machines consists of dividing the system into communicating components, in which each component is a State Machine. One advantage of this approach is the possibility of automatic validation of the model. The main limitations are the low abstraction level and the problem of the high number of created states to represent operations between components. Wu and Loui [10] presented the idea of modeling asynchronous protocols for communication across unreliable channels using finite-state machines communicating via an unreliable shared memory. It is shown that there are robust protocols for deletion and insertion errors. The state machine and intermediate variables were applied to solve the problem of difficulty of regulate the input variables and complex properties that can not be described in temporal logic and verify the related properties of the data flow control module, overcoming the incompleteness of the traditional methods [11].

Another modeling language that has been applied to model protocols is the Timed Automata [12], which consists in an approach of State Machine to treat time and clock modeling. The UPPAAL environment allows modeling and validation of Timed Automata models [13][14]. One of main properties of Timed Automata is that, although the set of configurations is in general infinite, checking reachability properties is decidable. However, an animation of Timed Automata cannot be determined, and inclusion checking is undecidable [12], except for deterministic timed automata. This basically forbids the use of timed automata as a specification language [15].

Since its introduction in 1962, but mostly after 1985, Petri Nets, a graphical and mathematical language, has been widely used to model communication protocols [16]. The language provides interesting modeling possibilities for real-time communication protocols, such as directly supporting modeling of concurrency, resource sharing and asynchronous events. The absence of compositionality is the main criticism raised in models created using Petri Nets [17][18]. Therefore, the level of abstraction is relatively low when comparing with UML. In addition, ordinary Petri Nets are not able to model temporal constraints [19]. In order to deal with the time modeling limitation, time extensions were proposed to the basic Petri net theory. The modular modeling of real-time communication protocol can be made using Time Petri Nets [20]. Another example is the Time Petri Nets model with Register (TPNR), which allows modeling of communication time delay [19]. However, this approach has a limited time structure such as the representation of composition time.

LOTOS allows the creation of many ways of transformation and validation of communication protocols. There are some examples of services of protocols implemented in LOTOS [21][22]. All approaches of LOTOS share the same problem, namely, the complexity of models. Besides, as the model is created in the early phases of a project, this property may difficult the construction of a complete model [23].

The UML [24] is currently widely applied in the software industry [25]. There are several approaches that use UML models as base for protocols [26][27][28]. Furthermore, UML is an extensible language, which makes it possible to create stereotypes and data types using the language metamodel. The mechanisms of extensibility allow to customize and extend UML resources, adding new building blocks, properties and specifying a new semantic, turning the UML adequate to specific domains. The Logical Link Control and Adaptation Layer Protocol (L2CAP) for wireless channel with bluetooth technology was modelled using UML, more specifically using the Sequence and State diagrams [27].

As was previously described, many modeling languages were applied to model communication protocols, and each one with specific characteristics. There are languages with focus on definitions of algebraic expressions and others on behavior and states. Therefore, in order to model a real-time communication protocol, it is necessary to use a modeling language that is capable of representing a lower abstraction level of modeling, including algebraic expressions, a model structure, robust time transformation and an abstract model. However, from the user point of view, it is also necessary to use a modeling language that is capable of modeling higher levels of abstraction, which makes more sense to the end user. Most commonly used modeling languages for communicating protocols lacks these characteristics.

III. ETARCH ARCHITECTURE

The Entity Title Architecture [29] (ETArch) is a clean state network architecture, where naming and addressing schemes are based on a topology-independent designation that uniquely identifies an entity, called Title, and on the definition of a channel that gathers multiple communication entities, called Workspace. A key component of this architecture is the Domain Title Service (DTS), which deals with all control aspects of the network. The DTS is composed by Domain Title Service Agents (DTSAs), which maintain information about entities registered in the domain and the workspaces that they are subscribed to, aiming to configure the network devices to implement the workspaces and to allow data to reach every subscribed entity.

Through ETArch, communications are handled by the Workspace. Therefore, ETArch inherently allows the integrated support of multicast and mobility within the Workspace that can be viewed as a logical bus interconnecting multiple entity instances (e.g., a service, a sensor, a smartphone, a host, or even a process). Its behavior is inspired by the multicast technology, where data is sent once by a source to the workspace, and all associated entities will receive.

The operation of ETArch, on which a centralized entity is responsible for the behavior of the forwarding plane, meets Software-Defined Networking (SDN) concepts [30], implemented in ETArch by the OpenFlow. OpenFlow [31] is an instantiation of SDN already available in a number of commercial products and used in several research projects. It separates the data plane from the control plane of the network, allowing a separate entity (i.e., the OpenFlow Controller) to manage and control the underlying data plane, configuring the forwarding table of the switches, via a well-known service-oriented API. This enables switches to be (re)configured on the fly, enabling flexible and dynamic network management [32] and allowing to bring life to the workspace driven communication concept.

Considering the ETArch networking model, the network itself is composed by several DTSAs that are configured in the model tree. When a workspace is requested by an entity that does not have DTS and workspace, it prompts the DTS higher-level information from that workspace, and the DTS asks the next level and so on, in a structure similar to the Domain Name System (DNS) used nowadays [33].

In order to support its concepts, ETArch defines protocols in the data and control plane. In the control plane, the signaling approach provides the services related with the life cycle of entities and workspaces, such as to register an entity at the Domain Title Service (DTS) or to create a workspace, attach and detach entities to a given workspace.

The Entity Title Control Protocol (ETCP) is responsible for the communication between an entity and the Domain Title Service Agent (DTSA), while the DTS Control Protocol (DTSCP) is responsible for the communication between DTSAs inside the DTS.

A. *Main ETCP primitives*

- ENTITY_REGISTER: Registers an entity at the DTS. To be registered an entity must present its title, capabilities and communication requirements. To communicate the entity must first register itself.
- WORKSPACE_CREATE: Creates a workspace locally at the DTSA. If the workspace has a public access after the successful creation, DTSA will advertise the workspace by inserting an entry at the Workspace Database.
- WORKSPACE_ATTACH: Attaches an entity to a workspace. To accomplish the attachment process, the DTSA will obtain all network elements and will configure them to extend that workspace. If the DTSA does have the information about the workspace, using the DTSCP protocol, it will send a WORKSPACE_LOOKUP primitive.
- ENTITY_UNREGISTER: Removes an entity from the DTS.
- WORKSPACE_DETACH: Removes an entity from an existing workspace.
- WORKSPACE_DELETE: Deletes a workspace and performs all clean up necessary at the NE of the current DTSA.

B. *Main DTSCP primitives*

- WORKSPACE_LOOKUP: Sent by a DTSA to its resolvers, i.e., the other DTSAs
- WORKSPACE_ADVERTISE: Inserts, deletes, or updates the Workspace Database, by indicating that a DTSA is part of the DTSA set of a specific workspace. The Operation receives the level indicating the visibility of that workspace. The DTSA stored at the Workspace Database must be of the same level or can be a Master DTSA of the level right below.
- DTS_MESSAGE: Enables communication between different DTSAs inside the DTS. If the DTSA source knows the path to the DTSA destination, this path will be contained in the message header. Otherwise, the message will be forwarded to the resolvers, until one of them knows how to compute the path to the destination DTSA. If the Master DTSA of the Root Level cannot compute the path to destination, the message will fail.

IV. Case Study

The modeling of the Etarch architecture protocol using UML aims to present the structure of the elements, behaviours and time constraints in a high abstraction level. For this, in this paper, the Class, Sequence and Composite Structure diagrams are presented.

The Class Diagram is responsible to define a classifier. Within this diagram, it is possible to define attributes, methods, visibility and relationship between one and many classifiers. The Class diagram is important in protocol modelling to define the used types, the data structures and the defined elements, and how they relate to each other. This is also useful for modeling the behavior of the protocol.

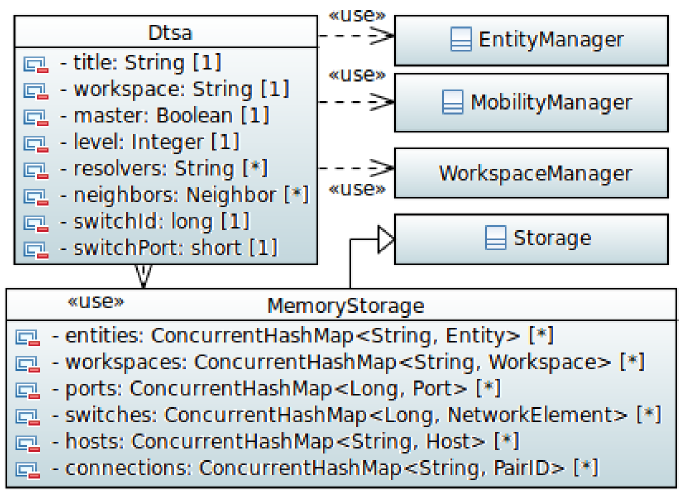
Dtsa
- title: String [1]
- workspace: String [1]
- master: Boolean [1]
- level: Integer [1]
- resolvers: String [*]
- neighbors: Neighbor [*]
- switchId: long [1]
- switchPort: short [1]
«use»
EntityManager
«use»
MobilityManager
WorkspaceManager
«use»
Storage
«use»
MemoryStorage
- entities: ConcurrentHashMap<String, Entity> [*]
- workspaces: ConcurrentHashMap<String, Workspace> [*]
- ports: ConcurrentHashMap<Long, Port> [*]
- switches: ConcurrentHashMap<Long, NetworkElement> [*]
- hosts: ConcurrentHashMap<String, Host> [*]
- connections: ConcurrentHashMap<String, PairID> [*]

Figure 1. Class Diagram - Main Elements

Fig. 1 presents some of the most important classes. The DTSA and Storage classes show the representation of attributes and the other classes in the diagram are a simple sample of elements definition. The Millisecond definition is used to define constraint unit. The elements shown in Fig. 1 are just a sample of some elements defined during modeling. This definition aims to show the level of abstraction represented by the class diagram, which is the representation of attributes and message definition, not focusing on the internal structure of the element, which would be modeled using the state-machine diagram.

The Composite Structure diagram allows to define a detailed view of a classifier structure, the relationship between attributes, input and output interfaces and data flow. In the Etarch architecture, the DTSA is one of the most relevant elements. Therefore, this structure is defined using a Composite Structure diagram. As an entity can be any device, and this behaviour is not relevant to architecture behaviour, then this internal structure will not be modelled.

For the best visualization, the DTSA definition is divided into three pictures, and then the DTSA is divided into two modules, the Resource Adapters that consist of flow control, which are defined as the bases communication protocol. The

Building Block is a composite element responsible of the service control and data storage of protocol data.

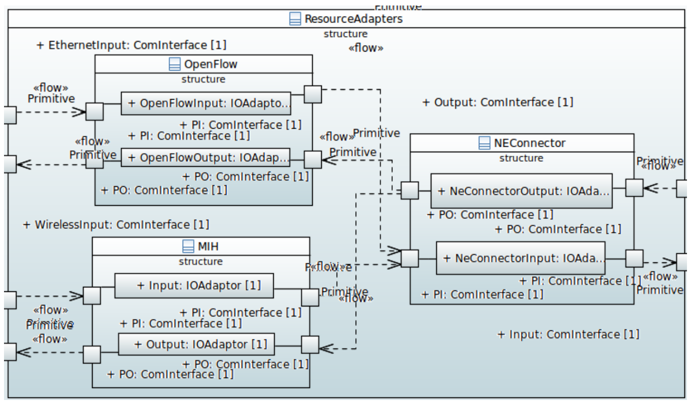
ResourceAdapters
structure
+ EthernetInput: ComInterface [1]
OpenFlow
+ OpenFlowInput: IOAdapto...
+ PI: ComInterface [1]
+ OpenFlowOutput: IOAdap...
+ PO: ComInterface [1]
+ Output: ComInterface [1]
NEConnector
+ NeConnectorOutput: IOAda...
+ NeConnectorInput: IOAda...
+ WirelessInput: ComInterface [1]
MIH
+ Input: IOAdaptor [1]
+ Output: IOAdaptor [1]
+ Input: ComInterface [1]
«flow»
Primitive

Figure 2. Composite Strucuture Diagram - Resource Adaptors

In Fig. 2, we introduce the structure of the element that represents protocols which standardizes protocol messages of Etarch. In the architecture, this protocol will be used to send messages until a responsible element that will treat and transform the message into architecture ETCP or DTSCP protocol's messages. In this case, the responsible element is called NE Connector.

In Etarch, two protocols are used to standardize the communication: OpenFlow [34] and Media Independent Handover (MIH) [35]. Requests originating from Ethernet will use OpenFlow protocol and requests originating from Wireless will use the MIH standard.

As there is no difference in input and output flow modeling, a distinction has been made in modeling the data flow in the elements. The implementation of elements must represent the concept shown in the modeled structure. As the element shown in Fig. 2 is responsible to intermediate messages in request and response, all input services requests to the DTSA must go through this module.

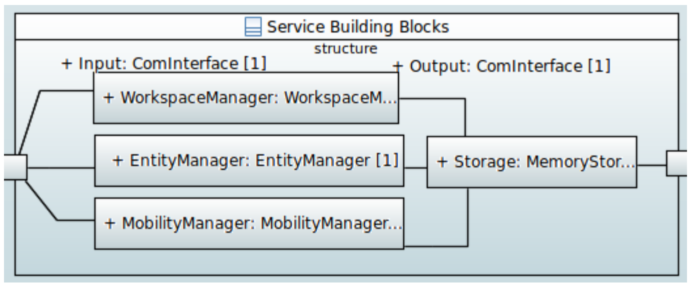
Service Building Blocks
structure
+ Input: ComInterface [1]
+ Output: ComInterface [1]
+ WorkspaceManager: WorkspaceM...
+ EntityManager: EntityManager [1]
+ Storage: MemoryStor...
+ MobilityManager: MobilityManager...

Figure 3. Composite Strucuture Diagram - Building Blocks

The module Building Blocks is depicted in Fig. 3. There are four internal elements in this module. The Workspace Manager that is accountable for all workspace related operations, as creation, attachment, detachment and deletion. The Entity Manager that treats entity requirements, as register and unregister. The Mobility Manager is responsible for mobility operations, as handover among others. The Storage is a generic structure to represent a database, and all the other structures of the same module to finish operation needs to modify the database.

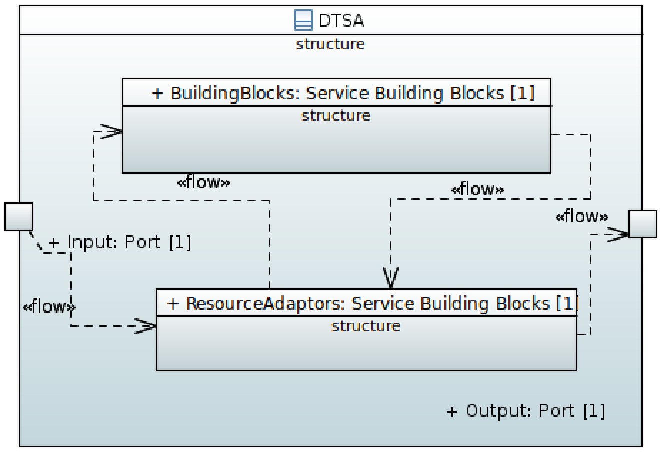
DTSA
structure
+ BuildingBlocks: Service Building Blocks [1]
structure
«flow»
«flow»
«flow»
+ Input: Port [1]
«flow»
+ ResourceAdaptors: Service Building Blocks [1]
structure
+ Output: Port [1]

Figure 4. Composite Strucuture Diagram - DTSA Structure

Fig. 4 represents the relationship in high abstraction level between models presented in Figs. 2 and 3. The request from an Entity must follow what is defined in the DTSA structure. In each module, there are range of behaviours. The most representative behaviours in this paper are Entity Register, Workspace Create, Workspace Attach and Workspace Lookup.

The definition of relevant service behaviour is performed using Sequence Diagrams. The visualization of parameters added in the message hampers visualization when the model is exported to image. Therefore, the name of parameters is added in the name of message.

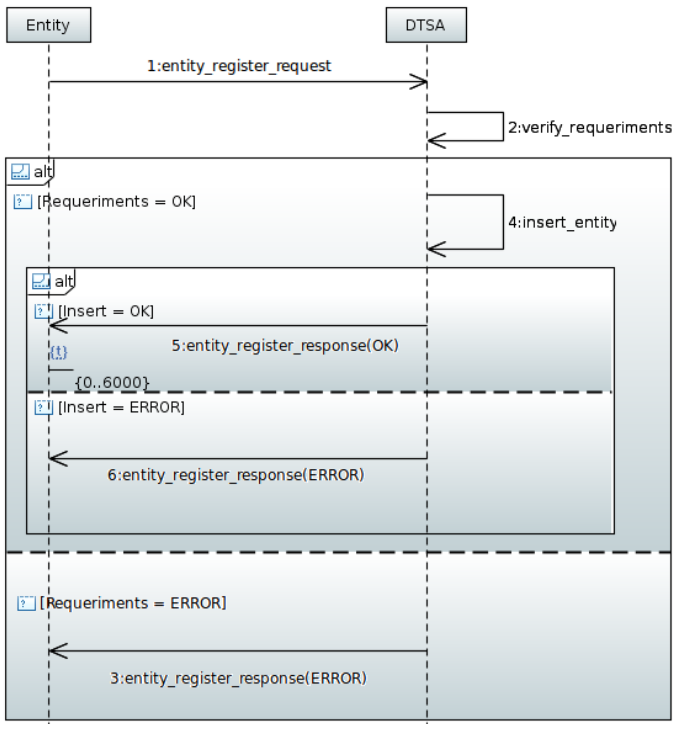

Figure 5. Sequence Diagram -Entity Register

Fig. 5 shows the Sequence Diagram of the entity register service in DTSA. In this diagram, the communication channel is abstracted in such a way that when a call goes to the device DTSA it passes through the flow structure described in the Resource Adaptors. The UML used resources are Time Constraint and Combined Fragment of the “alt” type. The predicates among guards mean the result of called operation.

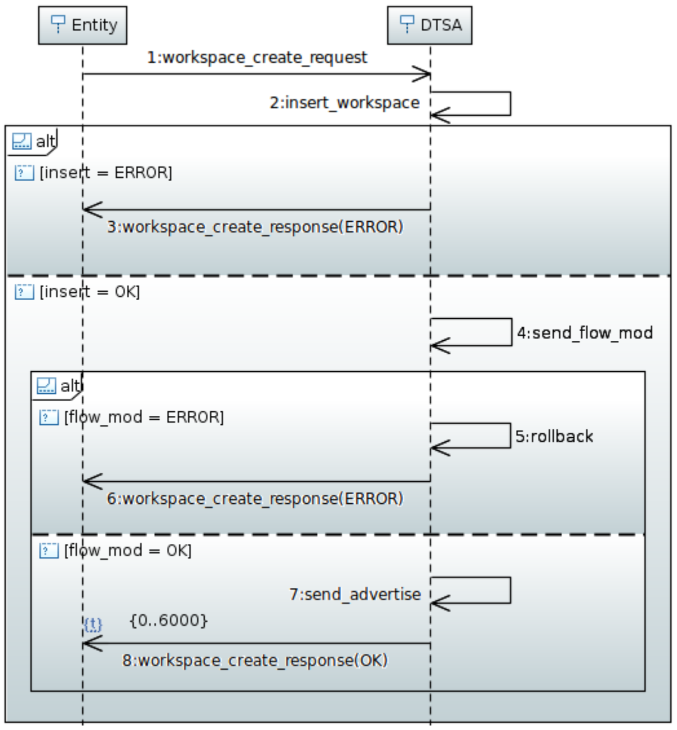

Figure 6. Sequence Diagram - Workspace Create

Fig. 6 represents the flow of workspace creation. In order to execute this operation, Resource Adaptors elements as Building Blocks elements are used. When a workspace is created, its basic information will be saved in storage.

Fig. 7 and Fig. 8 are related. The Workspace Lookup is a sub-process of Workspace Attach. The reference of this diagram is not presented in the figures to improve presentation, but the reference is already in a tool level. In this operation, the time constraints of Workspace Lookup must be taken into account in Workspace Attach. Workspace Lookup has two time constraints, in the search for the next DTSA level the time constraint is essential to the search operation ends.

By proposing an approach of communication protocols modeling, it is important to analyse related works, in particular such one based on state machine. Changes in the modeling language could be tracked by using a mapping function to translate it into a state machine diagram.

This paper presents the first step of a work that aims to create a formalizable scope of UML, UML profiles and enabling enhance the modeling of communication protocols, i.e., it aims to define a set of elements that we can apply transformation rules to a validatable method, or even create such a method. Certainly, it will be necessary to use resources of models transformation between different modeling languages. The Sequence Diagrams, despite of being a little explored approach in this context, are visually more representatives than others modeling techniques. For a formal validation of

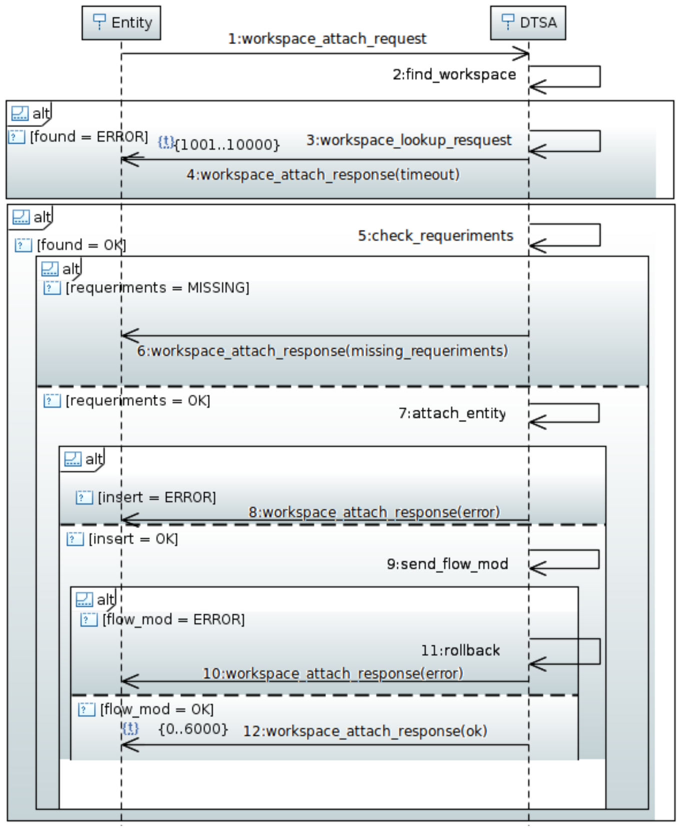

Figure 7. Sequence Diagram - Workspace Attach

Sequence Diagrams, we envisage the use of approaches such as transformation into Petri Nets [36].

Still in the perspective of model transformation, the approach of behavior modeling of communication protocols through Sequence Diagrams services can use synchronization techniques between Sequence Diagrams and other diagrams [37][38].

The modeling of ETArch Protocol, by UML language, follows two ways, being the first one structural modeling, by involving the use of Class and Composite Structure diagrams. The second, through the Sequence diagrams, thus, the necessary elements are: lifeLines, synchronous and asynchronous messages, combined fragments (alt and loop), time and duration observation. Through these elements there are two approaches for transforming models in Petri Nets, in [36] is possible to transform messages, lifelines and combined fragments, however, an approach for modeling observation time and duration is not displayed. Ribeiro and Fern [39] introduced and explained an approach that supports the transformation selected elements in Coloured Petri Nets.

V. Conclusion

This work has shown the modeling of a DTSCP and ETCP communication protocols using the UML language. The UML language has many resources to model components structure, which helps describing the high level of an architectural view. However, the language does not provide a formal definition to communication channel. Therefore, behaviour modeling can

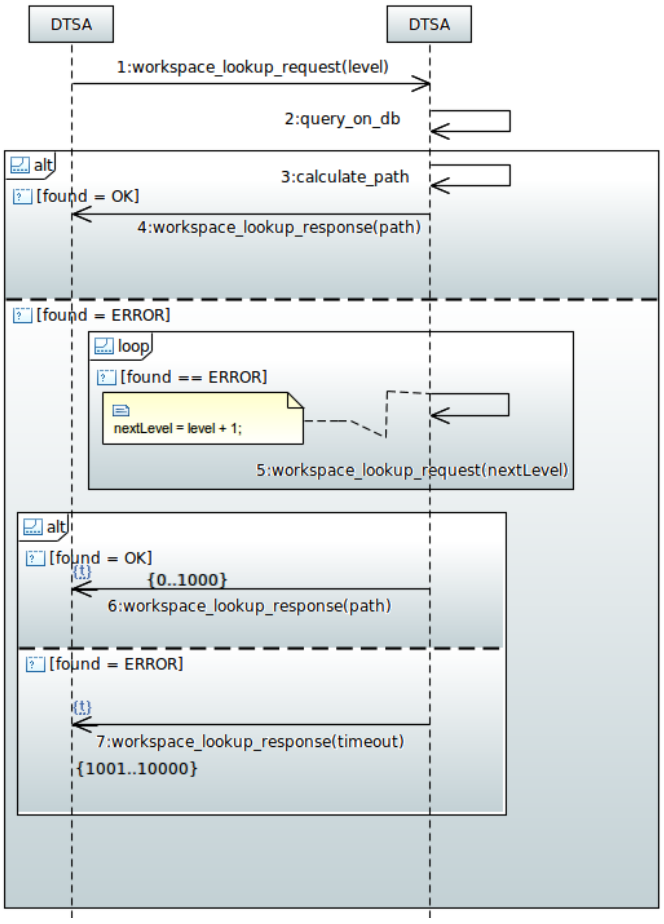
DTSA
DTSA
1:workspace_lookup_request(level)
2:query_on_db
alt
3:calculate_path
[found = OK]
4:workspace_lookup_response(path)
[found = ERROR]
loop
[found == ERROR]
nextLevel = level + 1;
5:workspace_lookup_request(nextLevel)
alt
[found = OK]
{0..1000}
6:workspace_lookup_response(path)
[found = ERROR]
7:workspace_lookup_response(timeout)
{1001..10000}

Figure 8. Sequence Diagram - Workspace Lookup

not represent bandwidth constraints, such as limitation can change the behaviour of time constraints.

The time constraints of UML allow the definition of different types for minimum and maximum time, however, it does not allow the creation of relationship between units of measure. The use of combined fragments is limited in static values in predicates.

A limitation of the UML language to define communication protocols is related to definitions of scenarios, because to do this it is necessary to deal with more than one flow definition. In one hand, this possibility is an advantage, but on the other hand its possible definition of data flow is not coincident. For example, in the definition of the Composite Structure Diagram, the data flow is from attribute A to B, and in the Sequence Diagram it is possible to define a message from B to A, injuring the previous definition.

Many ideas can be explored for future work. According to the resources used in this work, it is possible to think about automatic transformation of sequence diagrams to Petri Nets [39] [36] with the purpose of providing formal verification of models. In [36], the great advantage is the use of transformation to a simple Petri Net. However, this transformation is not enough to present the architectural structure defined in the element that is performing the actions. It is necessary a method to transform and attach all related diagrams, and to do this we must define a data flow validation. Therefore, it is possible to model the Etarch protocol services using UML language and the model is able to represent behaviour, time constraints and an abstract architecture of involved elements. The main disadvantage is that it is not possible to validate the complete model, taking into account all structures.

References

[1] "Information Processing Systems - Open Systems Interconnection- 'ESTELLE- A Formal Description Technique Based on tan Extended State Transition Model," 1988.

[2] G. von Bochmann, "Finite State Description of Communication Protocols," Computer Networks, vol. 2, 1978, pp. 361–372.

[3] S. Simonak, S. Hudak, and S. Korecko, "Protocol Specification and Verification Using Process Algebra and Petri Nets," in Computational Intelligence, Modelling and Simulation, 2009. CSSim '09. International Conference on, 2009, pp. 110–114.

[4] O. Ganea, F. Pop, C. Dobre, and V. Cristea, "Specification and Validation of a Real-Time Simple Parallel Kernel for Dependable Distributed Systems," in Emerging Intelligent Data and Web Technologies (EIDWT), 2012 Third International Conference on, 2012, pp. 320–325.

[5] J. Liu, X. Ye, and J. Li, "CP-Nets Based Methodology for Integrating Functional Verification and Performance Analysis of Network Protocol," in 11th ACIS International Conference on Software Engineering Artificial Intelligence Networking and Parallel/Distributed Computing (SNPD), June 2010, pp. 41–46.

[6] M. Yusufu and G. Yusufu, "Comparative Study of Formal Specifications through a Case Study," in International Conference on Information Science and Technology (ICIST), March 2012, pp. 318–321.

[7] S.-S. Park and N. Shiratori, "Distributed Systems Management Based On OSI Environment: Problems, Solutions, and Their Evaluation," in IEEE 13th Annual International Phoenix Conference on Computers and Communications, 1994, pp. 384–.

[8] J. Day and H. Zimmermann, "The OSI Reference Model," Proceedings of the IEEE, vol. 71, no. 12, Dec 1983, pp. 1334–1340.

[9] E. D. S. Santos, F. S. F. Pereira, J. H. de Souza Pereira, L. C. Theodoro, P. F. Rosa, and S. T. Kofuji, "Meeting Services and Networks in the Future Internet," in Future Internet Assembly, ser. Lecture Notes in Computer Science, J. Domingue, A. Galis, A. Gavras, T. Zahariadis, D. Lambert, F. Cleary, P. Daras, S. Krco, H. Muller, M.-S. Li, H. Schaffers, V. Lotz, F. Alvarez, B. Stiller, S. Karnouskos, S. Avessta, and M. Nilsson, Eds., vol. 6656. Springer, 2011, pp. 339–350.

[10] M. Wu and M. Loui, "Modeling Robust Asynchronous Communication Protocols with Finite-State Machines," IEEE Transactions on Communications, vol. 41, no. 3, 1993, pp. 492–500.

[11] W. Hua, X. Li, Y. Guan, Z. Shi, L. Dong, and J. Zhang, "Formal Verification for SpaceWire Communication Protocol Based on Environment State Machine," in 8th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM), Sept 2012, pp. 1–4.

[12] R. Alur and D. L. Dill, "A Theory of Timed Automata," Theoretical Computer Science, vol. 126, no. 2, 1994, pp. 183–235.

[13] X. Wu, H. Ling, and Y. Dong, "On Modeling and Verifying of Application Protocols of TTCAN in Flight-Control System with UPPAAL," in International Conference on Embedded Software and Systems, 2009, pp. 572–577.

[14] O. Al-Bataineh, T. French, and T. Woodings, "Formal Modeling and Analysis of a Distributed Transaction Protocol in UPPAAL," in 19th International Symposium on Temporal Representation and Reasoning (TIME), 2012, pp. 65–72.

[15] C. Baier, N. Bertrand, P. Bouyer, and T. Brihaye, "When Are Timed Automata Determinizable?" in Automata, Languages and Programming, ser. Lecture Notes in Computer Science, S. Albers, A. Marchetti-Spaccamela, Y. Matias, S. Nikoletseas, and W. Thomas, Eds. Springer Berlin Heidelberg, 2009, vol. 5556, pp. 43–54.

[16] K. Saleh, "Synthesis of Communications Protocols: An Annotated Bibliography," SIGCOMM Comput. Commun. Rev., vol. 26, 1996, pp. 40–59.

[17] N. A. Anisimov and M. Koutny, "On Compositionality and Petri nets in Protocol Engineering," in PSTV, ser. IFIP Conference Proceedings, P. Dembinski and M. Sredniawa, Eds., vol. 38. Chapman & Hall, 1995, pp. 71–86.

[18] C. Lakos, J. Lamp, C. Keen, and B. Marriott, "Modelling Network Protocols with Object Petri Nets," in Proc. of Workshop on Petri Nets Applied to Protocols. Springer-Verlag, 1995, pp. 31–42.

[19] K. El-Fakih, H. Yamaguchi, G. v. Bochmann, and T. Higashino, "Protocol Re-synthesis Based on Extended Petri Nets," 2000.

[20] A. Masri, T. Bourdeaud'huy, and A. Toguyeni, "Network Protocol Modeling: A Time Petri Net Modular Approach," in 16th International Conference on Software, Telecommunications and Computer Networks, 2008, pp. 274–278.

[21] C. Kant, T. Higashino, and G. V. Bochmann, "Deriving Protocol Specifications from Service Specifications Written in LOTOS," Distrib. Comput., vol. 10, no. 1, 1996, pp. 29–47.

[22] M. Kapus-Kolar, "Comments on Deriving Protocol Specifications from Service Specifications Written in LOTOS," Distributed Computing, vol. 12, 1999, pp. 175–177.

[23] T. Bolognesi and E. Brinksma, "Introduction to the ISO Specification Language LOTOS," Comput. Netw. ISDN Syst., vol. 14, no. 1, 1987, pp. 25–59.

[24] G. Booch, J. Rumbaugh, and I. Jacobson, The Unified Modeling Language User Guide, (Addison-Wesley Object Technology Series), A.-W. Professional, Ed. Addison-Wesley Professional, 2005.

[25] C. Lange and M. Chaudron, "An Empirical Assessment of Completeness in UML Designs," in Proc. Conf. Empirical Assessment in Software Engineering, 2004, pp. 111–121.

[26] M. Jaragh and I. Saleh, "Protocols Modeling using the Unified Modeling Language," in Proceedings of IEEE Region 10 International Conference on Electrical and Electronic Technology, vol. 1, 2001, pp. 69–73.

[27] K. Sekaran, "Development of a Link Layer Protocol using UML," in International Conference on Computer Networks and Mobile Computing, 2001, pp. 309–315.

[28] A. Bagnato, A. Sadovykh, E. Brosse, and T. E. Vos, "The OMG UML Testing Profile in Use–An Industrial Case Study for the Future Internet Testing," 15th European Conference on Software Maintenance and Reengineering, vol. 15, 2013, pp. 457–460.

[29] F. de Oliveira Silva, M. Goncalves, J. de Souza Pereira, R. Pasquini, P. Rosa, and S. Kofuji, "On the Analysis of Multicast Traffic Over the Entity Title Architecture," in 18th IEEE International Conference on Networks (ICON), 2012, pp. 30–35.

[30] Open Networking Foundation, "Software-Defined Networking: The New Norm for Networks," 2012. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

[31] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, 2008, pp. 69–74, ACM ID: 1355746.

[32] H. Kim and N. Feamster, "Improving Network Management with Software Defined Networking," IEEE Communications Magazine, vol. 51, no. 2, 2013, pp. 114–119.

[33] S. T. K. Flavio Oliveira Silva, Joao Henrique. S. Pereira and P. F. Rosa, "Domain Title Service for Future Internet Networks," in SBRC WPEIF, 2011.

[34] B. Sonkoly, A. Gulyas, F. Nemeth, J. Czentye, K. Kurucz, B. Novak, and G. Vaszkun, "On QoS Support to Ofelia and OpenFlow," in European Workshop on Software Defined Networking (EWSDN), 2012, pp. 109–113.

[35] D. Griffith, R. Rouil, and N. Golmie, "Performance Metrics for IEEE 802.21 Media Independent Handover (MIH) Signaling," Wirel. Pers. Commun., vol. 52, no. 3, 2010, pp. 537–567.

[36] M. S. Soares and J. Vrancken, "A Metamodeling Approach to Transform UML 2.0 Sequence Diagrams to Petri Nets," in Proceedings of the IASTED International Conference on Software Engineering, 2008, pp. 159–164.

[37] J. Whittle and J. Schumann, "Generating statechart designs from scenarios," in Software Engineering, 2000. Proceedings of the 2000 International Conference on, 2000, pp. 314–323.

[38] R. Grønmo and B. Møller-Pedersen, "From sequence diagrams to state machines by graph transformation," in Proceedings of the Third International Conference on Theory and Practice of Model Transformations, ser. ICMT'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 93–107.

[39] O. R. Ribeiro and J. M. Fern, "Some Rules to Transform Sequence Diagrams into Coloured Petri Nets," in In 7th Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, 2006, pp. 237–256.