

Semantically Enriched Services to Understand the Need of Entities

Flávio de Oliveira Silva^{1,3}, Alex Dias², Caio César Ferreira³,
Eduardo De Souza Santos³, Fabíola Souza Fernandes Pereira⁴,
Isabelle Cecília de Andrade³, João Henrique de Souza Pereira¹,
Lásaro Jonas Camargos³, Luiz Cláudio Theodoro³,
Maurício Amaral Gonçalves³, Rafael Pasquini³, Augusto José Venâncio Neto⁵,
Pedro Frosi Rosa³, and Sergio Takeo Kofuji¹

¹ University of São Paulo, Brazil

{flavio,kofuji}@pad.lsi.usp.br, joaohs@usp.br

² Federal Institute of Triângulo Mineiro, Brazil

alexdias@iftm.edu.br

³ Federal University of Uberlândia, Brazil

{caiocf,isabelle,mauricio}@algartelecom.com.br, eduardo@doutorado.ufu.br,

{lasaro,pasquini,frosi}@facom.ufu.br, lclaudio@feelt.ufu.br

⁴ Algar Telecom, Brazil

fabiolas@algartelecom.com.br

⁵ Federal University of Ceará, Brazil

augusto.deti@ufc.br

Abstract. Researchers from all over the world are engaged in the design of a new Internet, and Software-Defined Networking (SDN) is one of the results of this engagement. Net-Ontology uses a SDN approach to bring semantics to the intermediate network layers and make them capable of handling application requirements and adapt their behaviour over time as required. In this paper we present an experimental evaluation of Net-Ontology and a feature comparison against the traditional TCP/IP stack. This paper extends our earlier work towards a Future Internet, showing a viable approach to introduce semantics at network lower layers by contributing to bring richer and efficient services.

Keywords: Future Internet, Enrich Services, Network Ontology, SDN, DTS, Workspace.

Introduction

The evolution of the intermediate network layers have been lagging behind that of the lower and upper layers. The Internet Protocols, specified more than three decades ago, are the likely culprit; the application needs have changed by leaps, while the TCP/IP has only been patched, trying to meet these requirements. Over the last few years, the networking community has strived to correct this phenomenon[1, 3, 4, 21].

Researchers from all over the world are engaged in the design of a new Internet, from the ground up. This so called *clean slate* approach, frees the research from the legacy of the current architecture and fosters innovations[18]. At a future time, when results should be deployed, the research will then be refocused to the transition from the current Internet to the future Internet

One of the results of this effort to create the Future Internet is Software-Defined Networking (SDN)[5, 6]. SDN enables researchers to innovate and experiment new network protocols, naming and addressing schemes, such as the one presented in this paper, which aims at bridging the evolutionary gap between upper, lower, and the intermediate network layers by using a richer semantics [15, 16].

FINLAN (Fast Integration of Network Layers) [9, 13, 14, 19] aims at providing high adaptability through the use of semantic concepts based on ontology, with the elimination of static routing and addressing tied to physical location, resulting in a better and efficient utilization of the network infrastructure.

FINLAN defines two intermediate layers that communicate between each other using OWL (Web Ontology Language), but that clearly differentiate in function: DL-Ontology and Net-Ontology.

The DL-Ontology layer is essentially responsible for data transfer between the Physical layer and the upper layers, handling the semantic communication between the peer entities and bringing a richer capacity to express their requirements. On the other hand, the Net-Ontology layer is accountable for handling service needs, as it is capable of understanding specific requirements from the application and adapting the communication to support them only when required, using DL-Ontology to deal with the semantic communication.

In this chapter we present the Net-Ontology layer, which sits between the DL-Ontology layer and the application. We also present its implementation and a first experimental evaluation. The implementation presented is based on the Title Model[17], our vision regarding future networks.

The remainder of this work is organized as follows: Section 1 describes the Net-Ontology. Section 2 shows the Net-Ontology implementation and Section 3 the experimental results. The conclusions are presented in Section 4.

1 The Net-Ontology

The DL-Ontology is the lower layer of the FINLAN stack depicted in Figure 1, and enables the communication using concepts expressed in OWL over the Physical layer.

The Net-Ontology layer is responsible for supporting the service needs of the upper layer and deliver them to the DL-Ontology layer, built according to the FINLAN Ontology. In this approach, the Net-Ontology is able to understand specific requirements of a given application that may arise over communication and provide them.

For example, let us suppose that two persons, P_1 and P_2 , are chatting over the Internet, using the application *FinChat* that runs over the FINLAN stack. In a certain moment, they want to start a secret conversation. To *FinChat* meet

this need, the only thing it has to do is to inform the Net-Ontology layer that from now on the chat is to be confidential. The Net-Ontology layer is able to understand this need and act accordingly modifying all packets exchanged from that moment.

The Net-Ontology consists, basically, of two main modules: *requirement analysis* and *requirement manager*, as depicted in Figure 1.

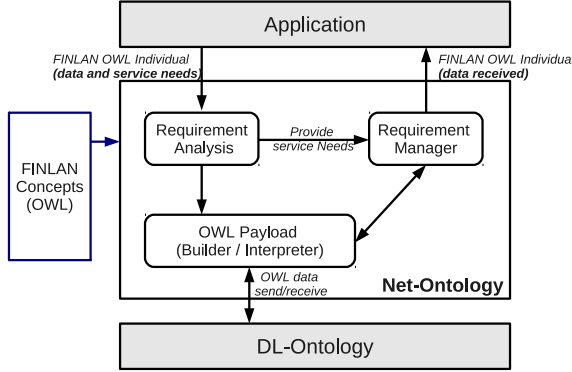


Fig. 1. Net-Ontology modules

The *requirement analysis module* (RAM) is responsible for handling the application requests regarding communication requirements. To accomplish this, RAM uses the Leśniewski's logic as proposed in [8]. The purpose is to manage the services requirements over time. This module recognizes what technological features are necessary to satisfy a given requirement, in a given moment, combining them in logical formulas.

As an example, let us suppose that a service $S1$, in a moment t_1 , may need to establish communication with the service $S2$, with a specific requirement. The RAM will verify that this upper layer requirement can be provided by the technological requirements $R1$ and $R2$. In another moment t_2 , $S1$ wishes to improve security, using confidentiality in the conversation. For so, it is necessary the technological requirement $R3$. These scenarios will be interpreted by the analysis module and represented by the following axioms:

$$S1S2t_1 \rightarrow R1 \wedge R2 \quad (1)$$

$$S1S2t_2 \rightarrow (R1 \wedge R2) \wedge R3 \quad (2)$$

The *requirement manager module* (RMM) takes the rules requirements and transform them into FINLAN ontology fragments. Besides that, this module is able to interpret and deploy the algorithms correlated with each requirement of the ontology in the network stack.

Taking the aforementioned example, in the moment t_1 , RMM receives the requirements R_1 and R_2 from RAM. It will then use the FINLAN Ontology, and add to the packages an OWL fragment, representing that P_1 has requirements R_1 and R_2 in *FinChat*.

From now on, the packages will be transmitted containing new information. When the *FinChat* of P_2 receives an OWL package, meaning that R_1 and R_2 are required, the RMM will be able to understand and make use of the necessary algorithms. The intelligence for the network to understand and implement the applications needs is the main responsibility of the *requirement manager module*.

The requirements, manipulated by RAM, are stored at the Domain Title Service (DTS), which consists of a distributed system over the network elements responsible for maintaining the entities available in a domain and their communication requirements over time. It plays an important role at central aspects of networking like naming and addressing, and has the ability to share the context among communicating entities. This sharing is provided by the workspace.

The workspace is a logical bus which contains network elements required to support the communication of the entities. The workspace is created by an entity willing to communicate with a specific purpose and thus defines its requirements and capabilities. A new entity can be joined to an existing workspace and, in such event, the logical bus can be adapted to handle its communication.

All entities that shares a workspace takes part in the same communication. The data is sent once by a source to the workspace and is received by all the others, thus making an efficient use of the physical layer.

In the next subsection it will be presented a complete case of how the Net-Ontology modules interact with the others FINLAN layers and the DTS.

1.1 FINLAN Semantic Communication

The communication between the FINLAN layers occurs in a semantic way, by using OWL. Below, it is presented an example to illustrate how this communication happens.

Let us suppose a scenario where *John* and *Paul* are chatting using the application *FinChat* that runs over the FINLAN layers, through the workspace WKS.1. In a first moment t_1 , they are just talking about irrelevant issues and are not concerned about any additional feature that *FinChat* can offer to them. So, the packages travelling in the network are very simple, and the Net-Ontology has not introduced any new requirement at the communication, in this case, only the DL-Ontology handles their communication. A code snippet example can be:

```
<Message rdf:ID="Message_1">
  <workspaceID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    WKS.1</workspaceID>
  <source rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Paul</
    source>
  <destination rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    John</destination>
  <payload rdf:datatype="http://www.w3.org/2001/XMLSchema#byte">Hello!
    How are you doing??</payload>
</Message>
```

After some time, at instant t_2 , *John* starts an important subject, and selects the feature *delivery guarantee* of *FinChat*. This means that from now on, *FinChat* requires delivery guarantee to the network. The Figure 2 shows the messages flow that will be sent and received between the *FinChat* entities and the DTS, to attend this request.

With a new requirement, the Net-Ontology layer is triggered, and the *requirement analysis module* checks that it is necessary the technological requirement of a delivery guarantee algorithm. *John's FinChat*, then, sends the following control message to DTS:

```
<ControlMessage rdf:ID="ControlMessage_1">
  <Application rdf:ID="FinChat">
    <HasNeed>
      <DeliveryGuarantee rdf:ID="DeliveryGuarantee_01"/>
    </HasNeed>
  </Application>
  <workspaceID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    WKS.1</workspaceID>
  <source rdf:datatype="http://www.w3.org/2001/XMLSchema#string">John</
    source>
  <destination rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    DTS</destination>
  <payload rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    AddNeed</payload>
</ControlMessage>
```

After registering *John's* need, the DTS will send him a confirmation message:

```
<ControlMessage rdf:ID="ControlMessage_1R">
  <Application rdf:ID="FinChat">
  <workspaceID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    WKS.1</workspaceID>
  <source rdf:datatype="http://www.w3.org/2001/XMLSchema#string">DTS</
    source>
  <destination rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    John</destination>
  <payload rdf:datatype="http://www.w3.org/2001/XMLSchema#string">OK</
    payload>
</ControlMessage>
```

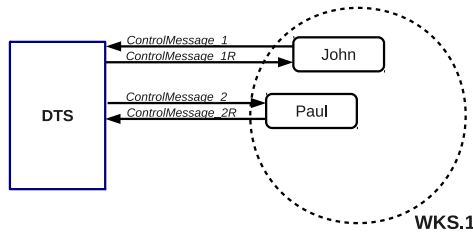


Fig. 2. Message flow example for a new requirement

At the same time, DTS will also send to *Paul*, who is in the same workspace as *John*, a control message, asking if the need requested is supported:

```

<ControlMessage rdf:ID="ControlMessage_2">
  <Application rdf:ID="FinChat">
    <HasNeed>
      <DeliveryGuarantee rdf:ID="DeliveryGuarantee_01"/>
    </HasNeed>
  </Application>
  <workspaceID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    WKS.1</workspaceID>
  <source rdf:datatype="http://www.w3.org/2001/XMLSchema#string">DTS</
    source>
  <destination rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    Paul</destination>
  <payload rdf:datatype="http://www.w3.org/2001/XMLSchema#string">isOK<
    /payload>
</ControlMessage>

```

If *Paul's FinChat* can supply the delivery guarantee feature, the response below is sent to DTS and it is established a communication with support to delivery guarantee:

```

<ControlMessage rdf:ID="ControlMessage_2R">
  <Application rdf:ID="FinChat"/>
  <workspaceID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    WKS.1</workspaceID>
  <source rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Paul</
    source>
  <destination rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    DTS</destination>
  <payload rdf:datatype="http://www.w3.org/2001/XMLSchema#string">OK</
    payload>
</ControlMessage>

```

In case of *Paul's FinChat* with no support for the delivery guarantee, this feature will not be present in the communication between both applications.

Notice that through the Net-Ontology, FINLAN is able to register the services needs into the DTS. From now on, it can manage what is the best way to deliver FINLAN packages.

If a third person, *Ringo*, wants to join the conversation, *Ringo's FinChat* will handshake with DTS to check if it has support to *DeliveryGuarantee_01*. This scenario is illustrated in Figure 3.

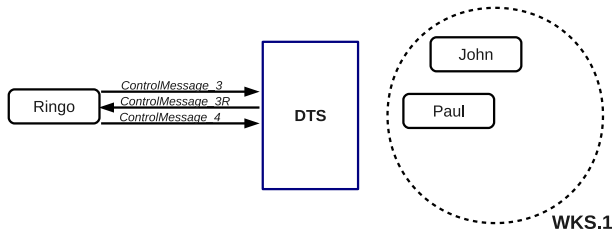


Fig. 3. Message flow example for a Join into Workspace *WKS.1*

The following messages are exchanged and *Ringo* joins the workspace *WKS.1*. After the joining and, hence, sharing of the workspace, *Ringo's FinChat* and all the other entities will receive the same data messages without the need of multiple data flows.

```

<ControlMessage rdf:ID="ControlMessage_3">
  <Application rdf:ID="FinChat">
    <workspaceID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      WKS.1</workspaceID>
    <source rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Ringo<
      /source>
    <destination rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      DTS</destination>
    <payload rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Join<
      /payload>
  </ControlMessage>

<ControlMessage rdf:ID="ControlMessage_3R">
  <Application rdf:ID="FinChat">
    <HasNeed>
      <DeliveryGuarantee rdf:ID="DeliveryGuarantee_01"/>
    </HasNeed>
  </Application>
  <workspaceID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    WKS.1</workspaceID>
  <source rdf:datatype="http://www.w3.org/2001/XMLSchema#string">DTS</
    source>
  <destination rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    Ringo</destination>
  <payload rdf:datatype="http://www.w3.org/2001/XMLSchema#string">isOK<
    /payload>
</ControlMessage>

<ControlMessage rdf:ID="ControlMessage_4">
  <Application rdf:ID="TestApplication">
    <workspaceID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      WKS.1</workspaceID>
    <source rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Ringo<
      /source>
    <destination rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      DTS</destination>
    <payload rdf:datatype="http://www.w3.org/2001/XMLSchema#string">OK</
      payload>
  </ControlMessage>

```

It is also important to mention that after the exemplified handshakes between DTS and *FinChat* entities, the *requirement manager module* is responsible for guaranteeing that the exchanged packages during the chat will have the necessary information, implementing the algorithm *DeliveryGuarantee_01*. For example, differently from the *Message_1* structure, the messages must have an identification field, through which the control of lost packages is made.

2 Implementation

Our FINLAN stack implementation consists of a Java library that uses communication interfaces through Raw Sockets. The linking between Java and C portions of the code was done in Java Native Interface (JNI) [16, 19], as depicted in Figure 4.

It is observed that the application *App.java* should use the API available in the library *Finlan.jar* to establish communication. In this way, when an application sends a packet, it communicates with the Net-Ontology sending its characteristics. According to these characteristics, the *Requirement Module Analysis* determines, through an inference engine, the application needs and proceeds

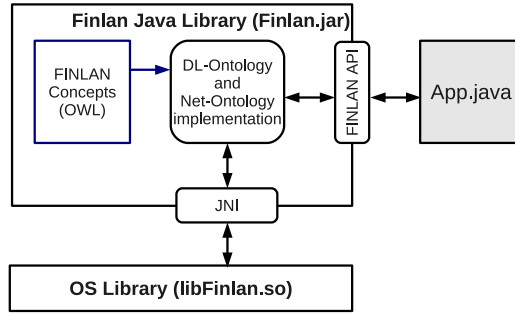


Fig. 4. FINLAN Modules Implementation

with the delivery of these. After the completion of the relevant operations, Net-Ontology sends the primitive to the DL-Ontology which, in turn, takes care of sending the packet through the JNI interface to the *libFinlan.so* library.

3 Experimental Results

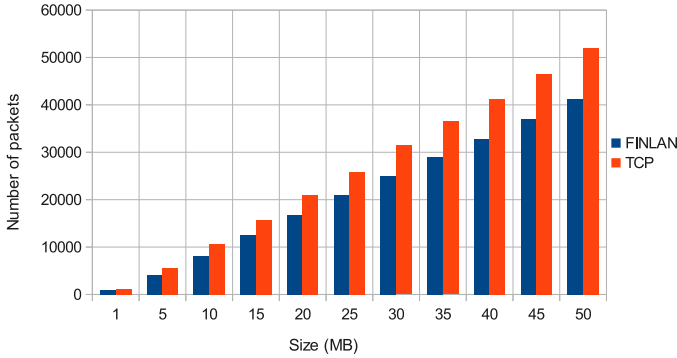
To validate the implementation of this work, it was performed experiments comparing FINLAN with the TCP/IP Architecture protocols. The goal of these experiments was to show the behavior of the ontology use in a file transfer operation with the delivery guarantee need activated, illustrating the use of the Net-Ontology layer in FINLAN.

The TCP/IP protocols, by definition, already implements the delivery guarantee feature, when the TCP transport protocol is used. To FINLAN support this need, this work implemented it in the Net-Ontology layer, using the algorithm described in [14]. This algorithm is a mechanism to ensure that all packages sent are received and works as follows: when the need delivery guarantee is activated by the application layer, all packages, sent from this moment contains a new field representing the number of the package.

In parallel, there is a mechanism of confirmation requests and responses messages: the source host informs the packages already sent, requesting the lost ones. The destination, on the other hand, answers which packages it did not receive. This change of confirmation messages is orchestrated by the RTT variable, proposed in [7], which consists of the best estimate (for that moment) for the send and receive time up of the packets destination.

3.1 Network Traffic Evaluation

The experiments were performed over the following environment: hosts with 4GB of RAM, CPU Intel® Core™2 DUO @ 2.10GHz, running Linux operational system with kernel 2.6.41.10-3.fc15.x86_64. The files transferred have size of 1, 5, 10, 15, 20, 25, 30, 35, 40, 45 and 50MB. The RTT variable was set to a fixed



(a) Number of packets

Size	FINLAN	TCP	Reduction (%)
1 MB	847	1168	37,9%
5 MB	4174	5594	34,0%
10 MB	8140	10631	30,6%
15 MB	12566	15733	25,2%
20 MB	16722	21060	25,9%
25 MB	20920	25277	25,6%
30 MB	25017	31423	25,6%
35 MB	29089	36495	25,5%
40 MB	32786	41143	25,5%
45 MB	37070	46585	25,7%
50 MB	41207	52067	26,4%

(b) Percentual reduction

Fig. 5. FINLAN and TCP traffic comparison

value of 1 second. Figure 5 shows the results, comparing the number of packets transmitted in both: FINLAN and TCP.

It is possible to observe that in the scenarios of this experimentation, FINLAN had a smaller number of transmitted packets. In the transfer operation of 10MB, for example, FINLAN transmitted 8140 packets, while the TCP transmitted 10631 (one difference of 30.6 percent).

This is due to the delivery guarantee algorithm implemented in FINLAN that sends confirmation messages in intervals of the RTT, informing only the lost ones, in a period, to be re-transmitted, while TCP transmitted several number of ACK packages. This confirms that the network traffic packets is decreased using the delivery guarantee algorithm implemented over a stack that semantically understands the concepts and adapts the messages from this understanding.

To illustrate the primitives in these experiments, Figure 6 shows snapshots from the Wireshark of two packets captured during the transmission of the 50MB file. The first one, in Figure 6(a), is the confirmation request of the source entity, called “fabiola”, informing that the range of packages from 133 to 367 was sent. The Figure 6(b) represents the response, confirming the lost packages, through

No.	Time	Source	Destination	Protocol	Length	Info
	305 9. 301837			DL-Onto	232	DL-ontology
+ Frame 305: 232 bytes on wire (1856 bits), 232 bytes captured (1856 bits)						
+ DL-Ontology						
0000	3c 43 6f 6e 66 69 72 6d	61 74 69 6f 6e 4d 65 73	<Confirm ationMes			
0010	73 61 67 65 20 72 64 66	3a 61 62 6f 75 74 3d 22	sage rdf :about="			
0020	23 43 32 34 22 3e 3c 72	64 66 3a 74 79 70 65 20	#CR24"><r df:type			
0030	72 64 66 3a 72 65 73 6f	75 72 63 65 3d 22 68 74	rdf:reso urce="ht			
0040	74 70 3a 2f 2f 77 77 77	2e 77 33 2e 6f 72 67 2f	tp://www .w3.org/			
0050	32 30 30 32 2f 30 37 2f	6f 77 6c 23 54 68 69 6e	2002/07/ owl#Thi			
0060	67 22 2f 3e 3c 53 6f 75	72 63 65 20 72 64 66 3a	g"/><Sou rce rdf:			
0070	72 65 73 6f 75 72 63 65	3d 22 23 66 61 62 69 6f	resource="#fabio			
0080	6c 61 22 2f 3e 3c 44 65	73 74 69 6e 61 74 69 6f	lav"><De stinatio			
0090	6e 20 72 64 66 3a 72 65	73 6f 75 72 63 65 5d 22	n rdf:re source="			
00a0	23 44 65 66 61 75 6c 74	22 2f 3e 3c 4d 65 73 73	#Default "><Mess			
00b0	61 67 65 49 44 20 72 64	66 3a 75 6e 73 69 67 6e	ageID r df:unsigned			
00c0	65 64 4c 6f 6e 67 3d 22	31 33 33 20 33 36 37 22	edLong="133 36"			
00d0	2f 3e 3c 2f 43 6f 6e 66	69 72 6d 61 74 69 6f 6e	/></conf irmatio			
00e0	4d 65 73 73 61 67 65 3e		Message>			

(a) Confirmation request

No.	Time	Source	Destination	Protocol	Length
	307 9. 326581			DL-Onto	54
+ Frame 307: 541 bytes on wire (4328 bits), 541 bytes captured (4328 bits)					
+ DL-Ontology					
0000	3c 43 6f 6e 66 69 72 6d	61 74 69 6f 6e 52 65 73	<Confirm ationRes		
0010	70 6f 6e 73 65 4d 65 73	73 61 67 65 20 72 64 66	ponseMes sage rdf		
0020	3a 61 62 6f 75 74 3d 22	23 43 52 32 34 22 3e 3c	:about=" #CR24"><		
0030	72 64 66 3a 74 79 70 65	20 72 64 66 3a 72 65 73	rdf:type rdf:res		
0040	6f 75 72 63 65 3d 22 68	74 74 70 3a 2f 2f 77 77	ource="h ttp://ww		
0050	77 2e 77 33 2e 6f 72 67	2f 32 30 30 32 2f 30 37	w.w3.org /2002/07		
0060	2f 6f 77 6c 23 54 68 69	6e 67 22 2f 3e 3c 53 6f	/owl#Thi ng"/><So		
0070	75 72 63 65 20 72 64 66	3a 72 65 73 6f 75 72 63	urce rdf :resourc		
0080	65 3d 22 23 6a 75 75 75	75 22 2f 3e 3c 44 65 73	e="#juuu u"/><Des		
0090	74 69 6e 61 74 69 6f 6e	20 72 64 66 3a 72 65 73	tination rdf:res		
00a0	6f 75 72 63 65 3d 22 23	44 65 66 61 75 6c 74 22	ource="# Default"		
00b0	2f 3e 3c 4c 6f 73 74 4d	65 73 73 61 67 65 51 75	/><LostM essageQu		
00c0	61 6e 74 69 74 79 20 72	64 66 3a 75 6e 73 69 67	antity r df:unsig		
00d0	6e 65 64 4c 6f 6e 67 3d	22 32 32 30 20 32 32 31	nedLong="220 221		
00e0	20 32 32 32 20 32 32 33	20 32 32 34 20 32 32 35	222 223 224 225		
00f0	20 32 32 36 20 32 32 37	20 32 32 38 20 32 32 39	226 227 228 229		
0100	20 32 33 30 20 32 33 31	20 32 33 32 20 32 33 33	230 231 232 233		
0110	20 32 33 34 20 32 33 35	20 32 33 36 20 32 33 37	234 235 236 237		
0120	20 32 33 38 20 32 33 39	20 32 34 30 20 32 34 31	238 239 240 241		
0130	20 32 34 32 20 32 34 33	20 32 34 34 20 32 34 35	242 243 244 245		
0140	20 32 34 36 20 32 34 37	20 32 34 38 20 32 34 39	246 247 248 249		
0150	20 32 35 30 20 32 35 31	20 32 35 32 20 32 35 33	250 251 252 253		
0160	20 32 35 34 20 32 35 35	20 32 35 36 20 32 35 37	254 255 256 257		
0170	20 32 35 38 20 32 35 39	20 32 36 30 20 32 36 31	258 259 260 261		
0180	20 32 36 32 20 32 36 33	20 32 36 34 20 32 36 35	262 263 264 265		
0190	20 32 36 36 20 32 36 37	20 32 36 38 20 32 36 39	266 267 268 269		
01a0	20 32 37 30 20 32 37 31	20 32 37 32 20 32 37 33	270 271 272 273		
01b0	20 32 37 34 20 32 37 35	20 32 37 36 20 32 37 37	274 275 276 277		
01c0	20 32 37 38 20 32 37 39	20 32 38 30 20 32 38 31	278 279 280 281		
01d0	22 2f 3e 3c 4e 75 6d 62	65 72 4d 65 73 73 61 67	"/><Numb erMessag		
01e0	65 49 44 20 72 64 66 3a	75 6e 73 69 67 6e 65 64	eID rdf: unsigned		
01f0	4c 6f 6e 67 3d 22 33 36	37 22 2f 3e 2e 2a 3f 3c	Long="36 7"/>.*?<		
0200	2f 43 6f 6e 66 69 72 6d	61 74 69 6f 6e 52 65 73	/Confirm ationRes		
0210	70 6f 6e 73 65 4d 65 73	73 61 67 65 3e	ponseMes sage>		

(b) Confirmation response

Fig. 6. Snapshots of FINLAN confirmation messages

the field *LostMessageQuantity*. According to this capture, the packages from 220 to 281 were lost and only them were re-transmitted.

4 Conclusions

This work presented the Net-Ontology Layer, experimental results of its implementation and how it is possible to use ontology at the intermediate networks layers to understand and support different entities needs.

The results of using ontology to support the delivery guarantee need demonstrate an optimization of more than 30 percent of the packets sent in a file transfer, compared with the traditional TCP/IP protocols usage.

By the Net-Ontology use, it was demonstrated the possibility to substitute the traditional TCP/IP protocols used at the transport and network layers. This brings more semantic power for the Future Internet networks, as the network intermediate layers become able to better understand the entities needs.

Future Internet is being constructed with worldwide collaboration and is based on research and experimentation. Our previous work showed [17, 19] how FINLAN approach and the Title Model Ontology can work together with different efforts regarding the future, while the work presented details on how these proposals can come true.

As future works, it is expected to experiment the Net-Ontology implementation in different testbeds, such as OFELIA [11] and FIBRE (Future Internet testbeds/experimentation between BRazil and Europe)[2, 20]. In complement, it will be finished the actual working in progress to the experimentation using OpenFlow [10]. Also, experimental tests using workspaces for multicast aggregation [12] are being executed at OFELIA testbed.

The research and experimentation results show that we are facing a viable approach to introduce semantics at network lower layers, by contributing to bring richer and efficient services.

Acknowledgment. This work received contributions from the FINLAN project and other researchers from the MEHAR program. The authors would like to thank all FINLAN and MEHAR members for the extensive discussion and collaboration to the Future Internet perspectives.

Open Access. This article is distributed under the terms of the Creative Commons Attribution Noncommercial License which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

References

- [1] Clayman, S., Galis, A., Chapman, C., Toffetti, G., Rodero-Merino, L., Vaquero, L.M., et al.: Monitoring Service Clouds in the Future Internet. Towards the Future Internet - Emerging Trends from European Research, p. 115 (2010)
- [2] FIBRE: FIBRE Project (January 2012), <http://www.fibre-ict.eu/>
- [3] FIRE: FIRE White Paper (August 2009), http://www.ict-fireworks.eu/fileadmin/documents/FIRE_White_Paper_2009_v3.1.pdf

- [4] Galis, A., Denazis, S., Bassi, A., Giacomini, P., Berl, A., Fischer, A., et al.: Management Architecture and Systems for Future Internet. Towards the Future Internet - A European Research Perspective, p. 112 (2009)
- [5] Greenberg, A., Hjalmtysson, G., Maltz, D.A., Myers, A., Rexford, J., Xie, G., Yan, H., Zhan, J., Zhang, H.: A clean slate 4d approach to network control and management. SIGCOMM Comput. Commun. Rev. 35, 41–54 (2005), <http://doi.acm.org/10.1145/1096536.1096541>
- [6] Greene, K.: TR10: Software-Defined Networking. MIT - Technology Review (2009)
- [7] Jacobson, V.: Congestion Avoidance and Control. SIGCOMM Communications Architectures and Protocols, USA 88, 314–329 (1988)
- [8] Leśniewski, S.: Comptes rendus des séances de la Société des Sciences et des Lettres de Varsovie. Class III, pp. 111–132 (1930)
- [9] Malva, G.R., Dias, E.C., Oliveira, B.C., Pereira, J.H.S., Kofuji, S.T., Rosa, P.F.: Implementação do Protocolo FINLAN. In: 8th International Information and Telecommunication Technologies Symposium (2009)
- [10] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: OpenFlow: Enabling Innovation in Campus Networks. SIGCOMM Comput. Commun. Rev. 38, 69–74 (2008)
- [11] OFELIA: OFELIA Project (January 2012), <http://www.fp7-ofelia.eu/>
- [12] de Oliveira Silva, F.: Experimenting domain title service to meet mobility and multicast aggregation by using openflow. In: MyFIRE Workshop (September 2011)
- [13] Pereira, F.S.F., Santos, E.S., Pereira, J.H.S., Rosa, P.F., Kofuji, S.T.: Proposal for Hybrid Communication in Local Networks. In: 8th International Information and Telecommunication Technologies Symposium (2009)
- [14] Pereira, F.S.F., Santos, E.S., Pereira, J.H.S., Rosa, P.F., Kofuji, S.T.: FINLAN Packet Delivery Proposal in a Next Generation Internet. In: IEEE International Conference on Networking and Services, p. 32 (2010)
- [15] Pereira, J.H.S., Pereira, F.S.F., Santos, E.S., Rosa, P.F., Kofuji, S.T.: Horizontal Address by Title in the Internet Architecture. In: 8th International Information and Telecommunication Technologies Symposium (2009)
- [16] Pereira, J.H.S., Santos, E.S., Pereira, F.S.F., Rosa, P.F., Kofuji, S.T.: Layers Optimization Proposal in a Post-IP Network. International Journal on Advances in Networks and Services (2011)
- [17] Pereira, J.H.S., Silva, F.O., Filho, E.L., Kofuji, S.T., Rosa, P.: The Title Model Ontology for Future Internet Networks. In: Domingue, J., et al. (eds.) FIA 2011. LNCS, vol. 6656, pp. 103–114. Springer, Heidelberg (2011)
- [18] Roberts, J.: The clean-slate approach to future internet design: a survey of research initiatives. annals of telecommunications - annales des télécommunications 64(5-6), 271–276 (2009), <http://www.springerlink.com/content/e240776641607136/>
- [19] Santos, E., Pereira, F., Pereira, J.H.: Meeting Services and Networks in the Future Internet. In: Domingue, J., et al. (eds.) FIA 2011. LNCS, vol. 6656, pp. 339–350. Springer, Heidelberg (2011)
- [20] Stanton, M.: FIBRE-EU and FIBRE-BR (October 2011), http://www.future-internet.eu/fileadmin/documents/poznan-documents/Session2_3-International_coop/2-3-stanton.pdf
- [21] Tselentis, G., Domingue, J., Galis, A., Gavras, G., Hausheer, D., Krco, S., Lotz, V., Zahariadis, T.: Towards the future internet a European research perspective. IOS Press, Amsterdam (2009)