В роботі досліджені питання побудови і взаємодії розподілених сенсорних мереж в архітектурі Інтернету речей і системах автоматизованого управління динамічними інфраструктурними об'єктами. Проаналізовано особливості та структура мультимедійних потоків цифрової телеметрії і пакетних даних між контроллерами сенсорних мереж міського транспорту. Запропоновано спосіб модифікації стандартного мережевого інтерфейсу Ethernet на підрівні управління логічним з'єднанням (LLC) за технологією "сирих сокетів" (Raw Socket) для спільної передачі багатоканальної телеметрії і пакетних даних. Розроблено програмний симулятор конвеєрно-модульного перенесення на мові Python в операційній системі Linux Ubuntu, в якому використано метод динамічного структурування даних тегами розмітки. Актуальність даної роботи обумовлена необхідністю подальшого підвищення міжмережевої інтероперабельності при побудові гетерогенних систем Інтернету речей. В результаті проведених досліджень обгрунтовано застосування конвеєрно-модульного перенесення (КМП) для обміну даними телеметрії з обмеженням затримок в системах контролю безпеки міського транспорту. Проведені випробування симулятора конвеєрно-модульного перенесення підтвердили релевантність і логічну несуперечливість основних принципів кодування, передачі і декодування мультимедійних даних в каналі зв'язку КМП. Отримані результати створюють науково-методичні передумови для поповнення існуючого стека TCP/IP новим протоколом міжмережевої взаємодії з обмеженням затримок, який може використовуватися спільно з протоколом IP в додатках реального часу Інтернету речей, і перш за все, в системах управління безпекою міського транспорту

Ключові слова: сенсорна мережа, Інтернет речей, взаємодія реального часу, контроль затримок, конвеєрно-модульний перенос

UDC 621.391.3
DOI: 10.15587/1729-4061.2019.162305

MODELING THE CONVEYOR-MODULAR TRANSFER OF MULTIMEDIA DATA IN A SENSOR NETWORK OF TRANSPORT SYSTEM

V. Tikhonov
Doctor of Technical Sciences, Professor*
E-mail: victor.tykhonov@onat.edu.ua
O. Tykhonova
Lecturer*
E-mail: elena.tykhonova@onat.edu.ua
O. Tsyra
PhD, Senior Lecturer*
E-mail: aleksandra.tsyra@gmail.com
O. Yavorska
Senior Lecturer*
E-mail: yavorskayao7@gmail.com
A. Taher
PhD
Department of Computer Technical Engineering
Islamic University
Kufa str., Najaf, Iraq, 54001
E-mail: abdallahqays@gmail.com
O. Kolyada
Senior Lecturer**
E-mail: e.shapenko@i.ua
S. Kotova
Senior Lecturer**
E-mail: e.shapenko@i.ua
O. Semenchenko
Senior Lecturer**
E-mail: e.shapenko@i.ua
E. Shapenko
Senior Lecturer**
E-mail: e.shapenko@i.ua
*Department of Telecommunication Networks
O. S. Popov Odessa National Academy of Telecommunications
Kuznechna ave., 1, Odessa, Ukraine, 65029
**Department of Transport Systems and Road Safety
National Transport University
M. Omelianovycha-Pavlenka ave., 1, Kyiv, Ukraine, 01010

1. Introduction

With the increase in the number of terminal devices, the worldwide Internet is gradually migrating towards the "Internet of Things" (IoT). In a more distant future, it is moving towards the "Internet of Everything" (IoE). Herewith, both traditional clients and servers and automated control systems (ACS) of various sensor objects in transport, energy, and other industries, are integrated in a common network infrastructure [1, 2].

Any distributed ACS functions by exchanging telemetry data between the ACS-controller and sensor network objects in real-time mode. Such automated control systems are sensitive to data transmission delays in the control loop, [3, 4]; these time delays can generate parasitic self-oscillations in the system, [5].

In the last decades, special hardware and software tools have been actively developed to build sensor networks [6, 7]. Many developments of this type exhibited as proprietary commercial products; therefore, a large variety of methods, tools and protocols emerged, as well as difficulties in standardizing these products raised. In this regard, the current direction in the field of telecommunications is to further increase the interoperability of heterogeneous sensor networks while their aggregation into the infrastructure of the Internet of things.

2. Literature review and problem statement

In the architecture of the Internet of Things (IoT), by analogy with the TCP/IP stack for the Internet, we distinguish three main layers for the open systems interoperability.

1) The access layer of terminal sensor devices, which represent distinct segments of a sensor network (sensor access networks).

2) The aggregation layer of sensor access networks into IoT-domains.

3) The interconnection layer of geographically and/or functionally separated IoT-domains of the entire Internet of things, each of which combines several segments of sensor access networks.

Consider the main characteristics of each of these three layers of interoperability in the architecture of the Internet of Things (IoT).

The first IoT-layer (i.e. sensor access networks) is typically built on the equipment of a particular manufacturer. Thus, diverse sensor access networks often meet difficulties in coherent operation because of various technical solutions offered by competing companies. Known approach at this IoT-layer is utilization of different Ethernet modifications, aka "Real Time Ethernet" (RTE), [8]. The RTE-technologies are used for both wired and wireless communications.

The second IoT-layer (aggregation of sensor access networks into IoT-domains) is not always possible to build entirely on the equipment of one manufacturer. Therefore, at the second layer, the problem arises of joining disparate hardware/software technologies from competing manufacturers.

In the TCP/IP model of the contemporary Internet no particular layer had been designed for real-time open system interconnection. Therefore, the conjunction of heterogeneous sensor networks into a common domain acts as an intermediate sublayer in TCP/IP architecture, which is located between the access layer and the internetworking layer (denote it L1.5).

The third IoT-layer (integration of distributed sensor domains) coincides with the second layer of the TCP/IP stack (i.e. IP-layer). However, the IoT-domains interaction in real-time mode on IP-protocol does not always meet the QoS requirements in M2M systems or services (e.g. urban traffic safety management).

Known methods and protocols of telemetry data exchange in sensor networks and IoT-segments mainly support the multichannel real-time data transmission at the access layer, [9].

For the real-time Ethernet (RTE), particular mechanisms are used to synchronize individual processes of the sensor network in RTE compatible protocols. Examples of such mechanisms are the open application protocol of the Industrial Ethernet family (EtherNet/IP, [10]), as well as the open industry standard based on the TCP/IP stack (PROFINET, [11]). One of the most popular IoT technologies is EtherCAT (industrial standard of the Industrial Ethernet family, [12]). Local sensor networks and control systems widely use the Ethernet Powerlink real-time data transfer protocol ([13]), as well as the third generation of the Serial Real time Communication System standard (SERCOS III, [14]).

The Ethernet Powerlink protocol provides for scheduling exchanges on the bus at the expense of the allocated time intervals in which access to the bus is allowed only to one of the devices. Also known is the IEEE 1588 standard, which allows synchronization of subscriber timers, [15]. A comparative analysis of the standards and protocols mentioned above is given in [16]. According to this analysis, the most widely used is the EtherCAT standard, developed by Beckhoff. The EtherCAT protocol specification is currently available only to members of the co-founders organization, which increases the cost of EtherCAT devices.

The EtherCAT protocol operates on packets transmitted within an IEEE 802.3 Ethernet frame (with an Ethertype 88a4 field) or within a UDP/IP datagram. The EtherCAT network segment combines one master device with its MAC address and many slave devices (without their own MAC addresses). The EtherCAT segment has a logical ring topology (although the physical topology can be of any other type like bus, star, etc.).

The 802.3 Ethernet frame payload field contains an embedded EtherCAT packet. The EtherCAT packet is indivisible and consists of a header (2 bytes) and one or more messages. The data sequence does not depend on the physical order of the nodes in the network, and the addressing can be serviced in any order. Each packet sent by the slave device sequentially traverses all the nodes of the segment with a specified time cycle.

Multicast and broadcast data transfer between final recipients is also possible, and should be implemented on the master device in the current network segment. If IP routing is not required, the EtherCAT protocol can be inserted into a UDP/IP datagram. This makes it possible to use the TCP/IP stack for addressing in EtherCAT segments.

Technological solutions offered in the telecommunications market in the field of compatibility of various IoT platforms rely primarily on the TCP/IP stack. At the same time, manufacturers provide for expansion of the address space and additional flow control capabilities in the IPv6 protocol [17]. The task of interconnection of individual segments and domains of sensor networks is partially solved by prioritizing real-time packet traffic in the known models of integrated services (IntServ) and differentiated services (DiffServ), [18]. Increasing the priority for telemetry streams reduces the overall average data latency, but variations of time delay may not be acceptable in specific M2M systems.

A wide range of IoT researches is focused on application-layer software interfaces, [19]. To increase the IoT scalability, researchers in academia and industry developed recommendations on heterogeneous IoT platforms, [20]. The problem of IoT standardization is dealt with by the Open Interconnect Consortium (OIC), AllSeen Alliance, oneM2M, OMALwM2M, ETSI M2M and other international organizations. A popular trend in IoT development is network function virtualization (NFV) on a software defined network (SDN), [21]. This approach implies an additional local controller embedded between the SDN control plane and application plane. Also,

technological developments of W3C's SemanticWeb at IoT application layer, such as the Resource Description Framework (RDF) and SPARQL [22], as well as Web Ontology Language (OWL) are known, [23]. On the way of IoT globalization and transition to integrated regional networks, the interoperability provision becomes a key issue of the 21st century [24]. In [25], the authors note that solving the interoperability problem for segments and domains of the IoT network yields up to 40 % of additional economic gain.

As a part of NGN researches at the telecommunication networks department of A.S. Popov ONAT (Ukraine), the new concept of packet networks interoperability is developed, [26–30]. This concept provides for the adaptation of basic data link layer technologies (primarily Ethernet) and the TCP/IP stack expansion to design IoT sensor network segments. This concept relies on special protocol for conveyor-modular data transfer in real time mode with latency control (CMT) acting along with conventional IP. The CMT method targets both traditional multimedia applications (digital telephony, video conferencing, etc.) and the telemetry data exchange in IoT sensor networks. Time delays are limited in real-time applications by installing virtual connections between individual controllers of sensor networks. In particular, the adaptation of Ethernet LLC sublayer has been proposed. The CMT supports QoS provision for real-time applications at the second (internetworking) layer of the TCP/IP model. This is higher than the access layer of industry sensor networks standards (such as EtherCAT, Ethernet Powerlink and others). On the other hand, the CMT does not affect the application layer of open systems interconnection (as in the above-mentioned technologies and protocols such as NFV/SDN, RDF and OWL). The CMT based IoT interoperability provides a packet-overhead to latency-control compromise with respect to conventional real-time data encapsulation, along with the ease of inter-layer data processing. The price of such a solution is the need to further expand the standards for the most promising data link layer technologies (Ethernet, WiFi and LTE). Physical interfaces, as well as the MAC sublayer of the data link layer of the underlying technologies, do not require modification to implement the CMT protocol. To support the conveyor-modular transfer of real-time data through modified link-layer frames, several variants of the CMT protocol are described in [25–30], including, based on IEEE 802.3 wired Ethernet, IEEE 802.11 radio (WiFi), and mobile LTE communication. Analyzing related works [8–17, 19, 20, 24, 26–31] we conclude that promissing increase of IoT interoperability is available by the use of conveyor-modular transfer of multimedia data (CMT) developed in the Odessa National Academy of Telecommunications [26–30].

However, more researches are needed for CMT method verification and related protocol specification. An important step in network protocol enhancement is computer simulation of basic algorithms and processes of the open system interaction that underlie the proposed method.

3. The aim and objectives of the study

This work aims conveyor-modular transfer simulation for multimedia data processing in the IoT sensor network domain of urban transport management system. On this premise, the following tasks are considered:

– Substantiation of an appropriate algorithm for multichannel telemetry transmission in urban transport management system;

– Development of the data link layer interface for conveyor-modular transfer via the Ethernet frames;

– The software simulation of multi-channel real-time data transfer combined with IP-packets delivery.

4. Substantiation of the algorithm for multichannel telemetry transmission

Consider the three-layered architecture of a distributed transport management system, Fig. 1. *The first* layer embraces four segments with local controllers (C1–C4) aggregated in two second layer domains. The number of segments in one domain depends on specific transport management system. For example, the first domain combines sensor networks for subordinated transport means control; the second one integrates air traffic and meteorology services, flight control, etc. The regional transport management system may also contain subsystems of road control, sea and river transport, each of which forms its own departmental domain of the sensor network on the Internet of things. The domain controller communicates with the segment controllers through some network environment. Ways to build this environment are of interest from the point of view of the interoperability problem, and are discussed further in this paper.

Any sensor network domain is a subnet on the second (aggregation) layer of IoT hierarchy. At the same time, sensor domains are entities of the 3-rd layer of IoT hierarchy incorporated through the IoT infrastructure by the central processor of sensor network, which interacts with domain controllers. Any sensor network under a common administration policy forms an IoT autonomous system (AS-IoT). Currently, the AS-IoT systems are mainly deployed on the access layer and are heterogeneous on object dynamics and QoS requirements. Today, the main protocol for interworking on the Internet is IP. However, consider the distributed objects management, IP acts as a sublayer of transport tunnel for telemetry data exchange. Other sublayers of this TCP/IP stack tunnel are UDP/TCP transport protocols along with the Real-Time Transport Protocol (RTP) and RTP control Protocol (RTCP). The distributed control system processes the upstream telemetry of sensor states and downstream telemetry of actuator commands.

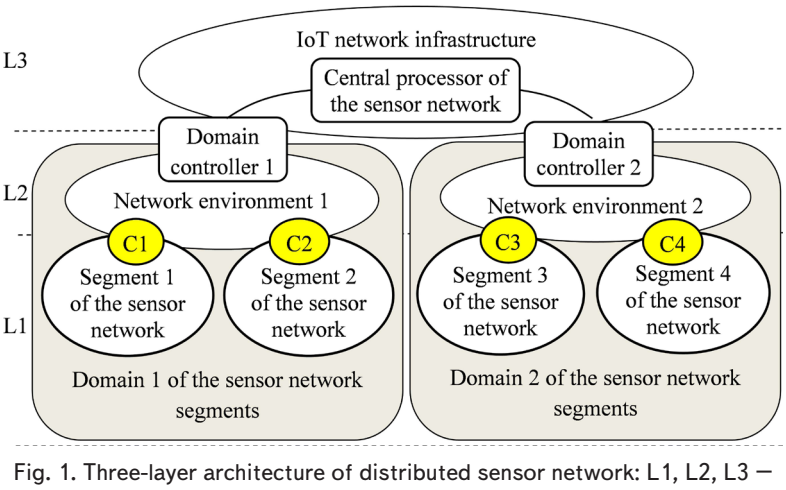

Fig. 1. Three-layer architecture of distributed sensor network: L1, L2, L3 — the layers of open systems interconnection in a sensor network

The IoT telemetry of sensor networks has the following properties.

1) The distinct object telemetry is mainly formed by short messages sequence.

2) The telemetry must be recurrent and low latent for sustained object control.

3) The telemetry QoS requirements widely differ depending on objects dynamics.

For instance, a modern urban traffic management system targets future unmanned vehicles control requiring millisecond or less telemetry latency. At the same time, the control subsystem of traffic lights is not so critical to the small time delays in data transmission. The telemetry time delay is formed by two main parts:

1) Unremovable latency caused by the finite time of electromagnetic wave propagation;

2) Operational delay in data processing nodes (mainly, the overall queuing time in switching/routing nodes).

An urban transport management network may include video monitoring subsystem with local data storage. However, the one-way video data transmission or streaming does not require high dynamics or strict time delay control, and therefore, can be carried out using UDP/TCP in batch mode, which is typical for IP networks. Thus, when building distributed AS-IoT, the coherence of many IoT-domains must be ensured in a wide range of QoS-requirements, along with the packet overhead minimization.

This problem solution involves integration of parallel data streams in a common physical communication channel which is divided into many heterogeneous logical subchannels (telemetry transmitted in circuit-switching mode, and data files delivered in packet-switching mode). Since different sensor networks are usually performed at the access layer on the equipment of one manufacturer, the local compatibility can be ensured within any segment. Therefore, the IoT access layer naturally allows for a large variety of competitive solutions, and objectively, it is difficult to unify. The interoperability and protocol compatibility issues mainly emerge on the second layer of IoT-hierarchy, Fig. 1. If sensor networks are compact and compatible (i. e. allocated on the common territory of one company), then a special backbone network can be created on the data link layer to aggregate these networks into domains.

The domain of a sensor network, whose entire infrastructure is subordinated to the common administration policy, is turned into an autonomous system of sensor network. So the AS-segments compatibility becomes an internal task of AS-administrator. In this case, achieving interoperability is less limited by the current standards. If the segments of a sensor network are located in different areas of the city or functionally divided, then a special network infrastructure for IoT-segments aggregation is neither technically nor economically expedient. In this case, the existing telecommunications infrastructure can be utilized based on the conventional TCP/IP protocols. Consider sensor networks interoperability. A relatively simple aggregation option is the use of IP-based Internet. To support this option, the sensor network controllers must operate on the TCP/IP platform. This approach prevails today in the IoT sensor networks.

When IP based aggregation of sensor networks, the IoT-domain loses the privilege of AS common administration policy; therefore, the sovereign admin policy is solely preserved at the access layer. Such a solution meets QoS issues in telemetry long distance delivery, because of IntServ/DiffServ models limitations. Besides, transmission the short real-time messages via the TCP/IP stack results in excessive overhead. The TCP/IP redundancy in sensor telemetry delivery is due to complicated multilayered RTP segments encapsulation (12-byte header). In turn, the RTP protocol segments are packed into UDP transport protocol segments (8-byte header). Next, UDP segments are embedded in IP packet (20-byte header for IPv4). Again, IP packets are enclosed in L2 frames (i. e. Ethernet frames with 18-byte header). The total overhead of TCP/IP encapsulation is now 58 bytes. In case of 2-byte telemetry unit, the overall IoT-channel utilization yields inefficient figure of $2/(2+58)\cdot 100\ \% \approx 3.3\ \%$.

The IP-aggregation of sensor networks turns the 3-layer IoT-architecture (Fig. 1) into a 2-layer framework, where the last two layers merged into one IP-based interworking layer. So, the sovereign admin policy drops to the layer of sensor network segments. Such a decision complicates the M2M systems design on the IoT platform. The integration of sensor network segments into IoT-domain via the public IP-network solely retains the AS-policy privilege when "transparent" telemetry traffic. This "transparency" implies telemetry latency control in a wide range of QoS requirements, as well as elimination of the conventional protocol overhead.

The sensor network domain formed by the sensor network integration through the transparent public network, can be considered as a virtual autonomous system (VAS). As noted above, the transparency of the IoT packet transporting system for telemetry traffic of sensor network segments can be achieved by conveyor-modular transfer of multimedia data (CMT, [26–30]). There are two principal CMT-algorithms.

The first algorithm for expanding the interoperability of the packet network provides for the introduction of one additional virtual circuit switching protocol VCP at the interworking layer (IP layer in the TCP/IP model), along with the current IP protocol (regardless of the specific IP protocol version) [26, 27]. The idea of this approach is as follows. In the gateway, a simple local network with one communication channel and a pair of Ethernet network interfaces is formed. In this network, the MAC addresses of the sender and receiver are known in advance, and there is no need to switch frames to the MAC addresses of the receiver. Frames are generated by the sender and processed by the receiver using the Raw Socket Ethernet technology. The Raw Socket Ethernet enables a system programmer to independently form a frame structure, including header fields, payload, and checksum. Due to this, in a single Ethernet frame instead of the traditional IP packet and header fields, there are two blocks of data:

1) Block of real-time data of variable length (which contains a tuple of segments of real-time data from many objects, for example, telemetry from sensors of the state of control objects);

2) Packet data block (which contains one or more fragments of a common packet queue); the length of this block is dynamically calculated as the rest of the overall frame payload after the real-time telemetry block is allocated.

According to the CMT method, the Raw Socket Ethernet frames circulate in both directions of the gateway with a constant duty cycle, forming a modular transport conveyor. The maximum possible frame length (within the selected Ethernet standard) is limited by the frequency of circulation and the bandwidth of the communication channel. The specific length of each frame changes dynamically depending on its actual payload. In the absence of data on the next duty cycle, the frame is omitted. While conveyor-modular data transmission, the time delay is predetermined by frame duty cycle

in a particular gateway. Real-time segments are transmitted via pre-established virtual connections based on channel resource reservation at each gateway hop. This eliminates sporadic telemetry queuing and latency deviation. The overall time delay yields the sum of all the intermediate gateway duty cycles. The considered above first algorithm of multimedia data integration with one additional virtual circuit switching protocol VCP has the following peculiarity. During each transmission cycle, a block of packet data is formed in accordance with dynamically shaped telemetry block which always occupies the first part of the frame payload. It means, that IP-packet queuing data is scheduled to the rest slot of the current frame. For this type of data scheduling, the necessary part of the queue of concatenated IP packets is truncated. Therefore, when there is a packet data queue, the frame is always full, which ensures high efficiency of communication channel utilization (more than 90 %).

The second algorithm of multimedia data integration developed in ONAT ([29]), provides for the introduction of two additional channel switching protocols along with conventional IP protocol. The first of the two novel protocols of the second algorithm (designated as VCP) supports the above method of virtual circuits switching with time delay control. The second protocol in this algorithm (designated as LCP) is intended for fast label switching of logical connections without latency control and resource reservation (by analogy with MPLS, but implemented at the interworking layer). The LCP protocol data segments are served by known methods; however, additional gain is achieved by overhead and delays reducing.

At this stage of computer simulation of conveyor-modular transfer, we have chosen the first algorithm (with one additional protocol VCP), since its implementation is simpler and more accessible in the short term. The implementation of the second algorithm may be an actual task in the IoT sensor networks in a more distant future.

5. Interface development of the conveyor-modular data transfer via Ethernet frames

We'll build an interface of the conveyor-modular data transfer by modified Ethernet frames on Raw Socket technology using **Send** and **Receive** function modules of Python programming environment in Linux Ubuntu operating system. The *Send* module runs on the sending workstation, and the *Receive* module – on the receiving workstation. Both modules run in admin mode.

Fig. 2 shows the program code of the **Send** module, designed for constructing and transmission of modified Ethernet frames. The operator *from socket import socket, AF_PACKET, SOCK_RAW* loads the standard *socket* module and defines the parameters *AF_PACKET, SOCK_RAW*. The last two parameters are used to build a "raw socket" at the data link layer of the TCP/IP stack, i. e. for non-standard frame formation.

Fig. 3 shows the program code of the **Receive** module, designed to receive modified Ethernet frames generated and transmitted by the *Send* module described above. The *import optparse, socket, time, binascii* operator loads four standard modules from the Linux library. The *optparse* module is the parser of parameters contained in program code operators. In the *BUF_SIZE=1,600* code line, the buffer size is set to 1600 bytes for receiving an Ethernet frame.

```
# Send module (Python 2.7.12 for Linux Ubuntu 16.04.3 LTS)
# First terminal (Ctrl+Alt+T): run the program as administrator, e.g.
# sudo python /home/user/Desktop/Send.py
# The patch-cord must be plugged to either the switch or another PC
# The wired interface must be activated by switching the "on" mode

import time
from socket import socket, AF_PACKET, SOCK_RAW  # used for L2 Raw Socket
from binascii import hexlify # to use in decoding hexadecimal numbers
sock = socket(AF_PACKET, SOCK_RAW)
sock.bind(("eth0", 0)) # Interface name get from the netstat -r
DMAC="\x00\x13\xE8\x7E\x05\xF5"  #  An actual DMAC shell be taken
SMAC="\x3C\xF8\x62\xE1\x3A\x6B"  #  The source address
PAYLOAD=(' Hello! ') # FRAME length must be more then 6+6+2+4=18
FRAME=DMAC+SMAC+PAYLOAD
interval = 1;          n=1
lastTime = time.time() # returns time as a floating point number in
# seconds since the epoch, in UTC (e.g. 1234892919.6559320)
while n<=50:      #  unlimited cycle in Python
    now = time.time ()
    if now > lastTime + interval:
      sock.send(FRAME)
      print 'Sent frame: DMAC= '+hexlify(FRAME[0:6])+' SMAC=
      '+hexlify(FRAME[6:12])+'  PAYLOAD= '+FRAME[12:]+'  Time =',now
      lastTime = now
      n=n+1
    else:
      time.sleep(0.1)
```

Fig. 2. Program code of the **Send** module

```
# Receive module (Python 2.7.12 for Linux Ubuntu 16.04.3 LTS)
# The dedicated network interface must be switched "on"

import optparse, socket, time, binascii
BUF_SIZE = 1600
ETH_P_ALL = 3
Interface = "eth0"
(MAC1,MAC2)=("3cf862e13a6b","0013e87e05f5")
# Open socket
sock = socket.socket(socket.AF_PACKET, socket.SOCK_RAW,
socket.htons(ETH_P_ALL))
sock.setblocking(0)

# Repeat receiving frames
interval = 1
lastTime = time.time()
while True: # Non-stop cycle (To stop use the command "Ctrl+C" from the terminal)
   now = time.time()
#  try/except/else
   try:
       FRAME = sock.recv(BUF_SIZE) # recive frame from buffer in packet
   except socket.error:
     # print ("Error")
       pass
   else:
      DMAC =  binascii.hexlify(FRAME[0:6]).decode()
      SMAC =  binascii.hexlify(FRAME[6:12]).decode()
      PAYLOAD=FRAME[12:]
      if(DMAC==MAC1 or DMAC==MAC2):
          print "Recv frame: DMAC= "+DMAC+" SMAC= "+SMAC+"  PAYLOAD=",
PAYLOAD," Time =", now
```

Fig. 3. Program code of the **Receive** module

6. Software modeling of the multi-channel multimedia data transfer

Fig. 4 shows a functional scheme of a software simulator for modeling the processes of conveyor-modular transfer of the telemetry data and packet data between sensor networks controllers in the Internet of things infrastructure. This scheme describes the simplex half of the duplex communication channel. The second simplex half of the channel is constructed similarly.

On the transmitting side of the channel, the **MUX** multiplexer receives a flow of segments from the Real Time Data Generator, as well as from the IP Packet Generator. From these two flows, the multiplexer forms Raw Socket Ethernet frames. Each successive frame is passed to the Send software module to be sent to the specified network interface (wired Ethernet or WiFi radio interface).

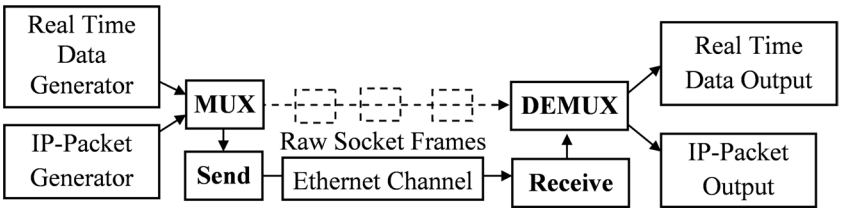

Fig. 4. Functional scheme of the simulator for multimedia data transfer

On the receiving side of the channel, the frame is received by the Receive program module and placed in the data processing buffer. Next, the **DEMUX** inverse multiplexing module parses the frame payload in the buffer. As a result, the buffer content is divided into a real-time data flow (Real Time Data Output) and an IP packet queue (IP-Packet Output). Real-time segments are transmitted over pre-established virtual connections (VC). Each virtual connection has a label, a fixed frequency and size of the segments. A modified Raw Socket Ethernet channel is formed by a pair of *Send/Receive* modules.

Consider a sensor network managing traffic safety at a complex intersection of the city's or megalopolis transport system. We'll assume that this sensor network contains three main segments according to departmental subordination:

– First segment: municipal electric transport (trams, trolley buses);

– Second segment: traffic police of the Ministry of Internal Affairs (manned and unmanned vehicles, motorcycles, bicycles and pedestrians with move control devices);

– Third segment: traffic service (traffic lights, sound signaling devices, surveillance cameras, etc.).

Each of the three sensor network segments is an autonomous system of the access layer, and contains its own controller. The segments associate into a domain via a domain controller, which provides their interaction with the urban traffic management system. The number of sensors and controls in each segment of such an access network may be many dozens, which determines the number of parallel flows and virtual channels for telemetry data transfer. Each flow has its own characteristics in terms of possible rates and admissible delays in the control loop. For example, the speed of cars at the intersection can reach 50 km/h, cyclist's – about 15 km/h, pedestrian's – 5 km/h.

For the model description simplicity, we divide all real-time telemetry flows into three main categories.

The first group is the most dynamic flows (for example, car telemetry). We assume that each frame transmitted between the segment controllers and the domain controller contains telemetry data blocks from all sensors or actuators from the first group of streams for a particular segment of the sensor network. The delay in transmission over the physical link is approximately one duty cycle T of frame circulation. The metropolis scale delay of electromagnetic waves propagation can be neglected.

The second group consists of medium dynamic flows (for example, cyclists' telemetry). The data blocks of these streams will be placed in every even of two successively transmitted frames. The transmission delay will be 2T.

The third group – the least dynamic flows (for example, telemetry of pedestrians and special vehicles); data blocks will be placed in every third of three successively transmitted frames. The transmission delay will be 3T.

For the convenience of visual control in the process of debugging a software simulator CMT, we will assume that all data blocks of the first category of telemetry streams are the same in size and value, and are equal to the character constant '11'. Blocks of the second category take constants '2222'; the third category '333333'. The operation of CMT multiplexer does not depend on the specific values of data blocks. The PAYLOAD size is 24 characters (maximum size is 1,506). In each frame, if there is free space, we will place a fragment of the packet queue (for example, information from surveillance cameras) in order to fully utilize the payload field. For ease of visualization, individual packages and their fragments carry the monotonous characters, for example: XXX, YYY, ZZZ, etc. The payload field formed in this way, containing telemetry data blocks and packet fragments, we designate as conveyor-transporting module (CTM).

Multiplexing telemetry blocks and fragments of a packet queue in the PAYLOAD field will be performed using markup tags, each of which is formed by a combination of the reserved character "C" (command) and other symbols:

C0 is the separator between the real-time data area (the initial part of the CTM module) and the packet data area (the remaining part of the CTM module);

CN is the label of the telemetry data block for the Nth category of real-time streams (C1, C2, C3, etc.);

CA is the beginning of the package; CC is a substitution command for transferring any tag as a data byte.

Suppose the packet queue is:

– XXXXXX;
– YYYYYYYYY;
– ZZ;
– HHHH;
– PPPP;
– QQQQQQQQQ;
– WWWWWW;
– RRRRRR;
– TTTTTT.

Then the first six CTM modules will take the form shown in Fig. 5.

The program code of the simplified version of the MUX multiplexer is shown in Fig. 6. The MUX module is designed as a standalone program that can be used independently of the *Send* and *Receive* modules for verifying the principle of multiplexing based on the entered markup tags. The working version of the MUX module is implanted in the Send module as subroutine based on an extended list of markup tags. The

first three operators in the MUX module determine the size of the CTM module, the frequency of telemetry data blocks appearance, as well as the number of characters in each data block. The operator fout = open ("/ home / user / Desktop / OutPack.txt", 'w') opens the OutPack.txt file for recording, which displays the results of the MUX module. Operator x = fin.read (1) sets the read pointer on the first character of the fin file. The DEMUX software module is in many respects similar to the MUX module and performs inverse multiplexing operation on the receiving side of the channel.

CTM-1	C111C0CAXXXXXXCAYYYYYYYY
CTM-2	C111C22222C0YYCAZZCAHHHH
CTM-3	C111C3333333C0CAPPPPCAQQ
CTM-4	C111C22222C0QQQQQQQQCAWW
CTM-5	C111C0WWWWCARRRRRRCATTTT
CTM-6	C111C22222C3333333C0TTCA

Fig. 5. Data structure in conveyor transporting modules CTM

```
# Begin MUX module
CTM=24
K1, K2, K3 = (1, 2, 3)
D1, D2, D3 = (2, 4, 6)
n=1
fin = open("/home/victor/Desktop/InPack.txt", 'r')
fout = open("/home/user/Desktop/OutPack.txt", 'w')
x = fin.read(1)
while n<=6:
  L=2
  print '\n CTM='"%2d"%(n),
        fout.write('\n CTM='"%2d"%(n),)
  if (n%K1 == 0):
        L=L+2+D1
        print '\t C1'+'11',
        fout.write ('\t C1'+'11',)
  if (n%K2 == 0):
     print 'C2'+'2222',
     fout.write ('C2'+'2222',)
     L=L+2+D2
  if (n%K3 == 0):
     print 'C3'+'333333',
     fout.write ('C3'+'333333')
     L=L+2+D3
  print 'C0',
  fout.write ('C0',)
  Freeslot = CTM-L
  if Freeslot >= 1:
     x = fin.read(Freeslot)
     print x
     fout.write(x)
  n=n+1
fin.close()
fout.close()
# End MUX module
```

Fig. 6. The program code of the data multiplexing module MUX

7. Discussion on computer simulation results

The sensor networks interoperability is a great challenge for different equipment manufacturers on the way to the Internet of Things. One of the possible approaches to sensor networks compatibility provision was proposed in the Odesa national academy of telecommunications (Ukraine) based on the conveyor-modular data transfer (CMT) at the OSI network layer. To verify the CMT method, *three main tasks* were set in this work.

As a result of solving *the first task* in Section 4 of this work, a compromise choice of the algorithm for multi-channel transmission of multimedia data and digital telemetry in sensor networks of the Internet of Things is justified. This compromise is achieved through a technically simple modification of the logical link control (LLC) sublayer in the network interfaces of one of the most popular local area network technologies Ethernet. Such a modification at the first stage of its implementation can be carried out consolidated within the framework of separate associations of autonomous systems, and does not require a complex procedure for the harmonization and adoption of international standards. The essence of the selected CMT algorithm (Odesa national academy of telecommunications) is the implementation of one additional interworking protocol (VLC virtual connection protocol) in addition to the conventional IPv4 and IPv6 protocols. The choice of such a solution allows combining two difficultly compatible methods of processing and transmitting data in one physical communication channel – circuit switching (with deterministic delay) and packet switching (with high channel resource efficiency).

The result of solving *the second task* (Section 5) is the development of a software interface for conveyor-modular transfer of multimedia data via the Ethernet link or WiFi radio channel. This result was achieved through the creation of two software modules in the Python language in the Linux operating system (modules *Send* and *Receive*), as well as through the use of a low-level Raw Socket Ethernet frame shaping mode.

The result of solving *the third task* (Section 6) is a software simulation of multimedia data transfer processes that combines the processing of real-time data (telemetry of sensor networks) and IP packets. This result was achieved by creating two software modules in the Python/Linux language (MUX and DEMUX). The MUX module multiplexes data on the transmit side of the channel, i.e. carries logical coding of telemetry blocks and fragments of IP packets in the form of formal grammar text using markup tags. The DEMUX module performs data demultiplexing, i.e. parsing of the formalized text on the receiving side of the channel. After demultiplexing, the input data stream is divided into an IP-packet queue and a multi-channel telemetry queue. The packet queue is processed by the router, and the telemetry queue is processed by the virtual connection switch.

The advantage of modeling the conveyor-modular transfer (CMT) compared to previously published results is the creation of the first working version of the software simulator, in which the processes of data multiplexing and demultiplexing at the logical link layer (LLC) are separated from the processes of cyclic data transfer via the Ethernet frames.

From the point of view of the problem formulated in Section 2, this work is the next stage in the complex of systemic research on verification of the theoretical method of conveyor-modular transfer (CMT). The results of the work confirm the logical correctness of the selected basic algorithm with one additional interworking protocol VCP (virtual connection protocol). The developed software simulator is a practical tool for a new spectrum of model experiments and specification of individual parameters of the VCP protocol.

The main limitations of the simulation of the interworking processes in the IoT networks described in the software simulator and based on it modeling results are as follows. The first limitation is a simplified visual user interface that does not allow for cognitive tracking of rather complex information processes in sensor networks. The second aspect is a small set of markup tags, which limits the number of possible channels for parallel transmission of telemetry data (up to

10 channels). The third limitation is the absence in the simulator of a built-in generator of random telemetry streams and IP packets with given statistical characteristics. In addition, the simulator described is intended only for network adapters and Ethernet interfaces.

The above limitations can be considered as shortcomings of this work from the point of view of further research. The most urgent tasks at the next stages are the improvement of the interactive graphical user interface, the extension of the logical coding language of the frame payload, the development of a methodology for modeling the information load of communication channels. Also of interest is the scaling of the principles of conveyor-modular transfer (CMT) to other local area network technologies.

8. Conclusions

1. In the work, the rationale and specification of the algorithm for multi-channel data transmission in a real-time transport management system at the aggregation layer of individual domains of sensor networks are carried out. The proposed algorithm is based on the sharing of packet data transmission based on the current IP protocol and the new method of conveyor-modular transfer (CMT), the protocols of which are under development.

2. To build a multiplex channel in the real-time systems of the Internet of Things, wired and wireless interfaces were developed using the Raw Socket Ethernet technology in the Python programming language in the Linux Ubuntu operating environment. The implementation of such a communication channel involves the modification of the logical link control (LLC) sublayer in Ethernet technology.

3. Software modules MUX and DEMUX in the Python/Linux environment for simulating multi-channel data transfer using the conveyor-modular transfer method were developed. Preliminary testing of these modules was carried out, which confirms the logical correctness of the proposed algorithm. Based on the simulation results, the directions for further research in the field of interconnection of sensor networks in the architecture of the Internet of Things are formulated.

References

1. Porkodi R., Bhuvaneswari V. The Internet of Things (IoT) Applications and Communication Enabling Technology Standards: An Overview // 2014 International Conference on Intelligent Computing Applications. 2014. doi: https://doi.org/10.1109/icica.2014.73
2. Internet of Nano-Things, Things and Everything: Future Growth Trends / Miraz M., Ali M., Excell P., Picking R. // Future Internet. 2018. Vol. 10, Issue 8. P. 68. doi: https://doi.org/10.3390/fi10080068
3. Machine-to-Machine (M2M) communications: A survey / Verma P. K., Verma R., Prakash A., Agrawal A., Naik K., Tripathi R. et. al. // Journal of Network and Computer Applications. 2016. Vol. 66. P. 83–105. doi: https://doi.org/10.1016/j.jnca.2016.02.016
4. Time-Delay Systems: Modeling, Analysis, Estimation, Control, and Synchronization / Boubaker O., Balas V. E., Benzaouia A., Chaabane M., Mahmoud M. S., Zhu Q. // Mathematical Problems in Engineering. 2017. Vol. 2017. P. 1–3. doi: https://doi.org/10.1155/2017/1398904
5. Yu W., Cao J., Chen G. Stability and Hopf Bifurcation of a General Delayed Recurrent Neural Network // IEEE Transactions on Neural Networks. 2008. Vol. 19, Issue 5. P. 845–854. doi: https://doi.org/10.1109/tnn.2007.912589
6. Bharathidasan A., Sai Ponduru V. A. Sensor Networks: An Overview. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.5089&rep=rep1&type=pdf
7. Zheng J., Jamalipour A. Introduction to Wireless Sensor Networks // Wireless Sensor Networks. 2008. P. 1–18. doi: https://doi.org/10.1002/9780470443521.ch1
8. Doyle P. Introduction to Real-Time Ethernet I // The Extension. A Technical Supplement to Control Network. 2004. Vol. 5, Issue 3. URL: http://www.ccontrols.com.cn/pdf/Extv5n3.pdf
9. Lammermann S. Ethernet as a Real-Time Technology. Leipzig, 2008. 21 p. URL: http://www.lammermann.eu/wb/media/documents/real-time_ethernet.pdf
10. EtherNet/IP Programmer's Guide // Parker Hannifin Corporation. 2009. URL: https://www.naic.edu/~phil/hardware/byuPhasedAr/floor/Parker_EthernetIP_UG.pdf
11. Cao J. PROFINET. URL: http://www.cs.wayne.edu/~hzhang/courses/8260/Lectures/Chapter%2012%20-%20PROFINET.pdf
12. The Ethernet Fieldbus // EtherCAT Technology Group. 2009. URL: https://www.ethercat.org/pdf/english/EtherCAT_Introduction_0905.pdf
13. EPSG Draft Standard 301. Ethernet POWERLINK Communication Profile Specification. Version 1.3.0 // Ethernet POWERLINK Standardisation Group. 2016. URL: https://www.ethernet-powerlink.org/fileadmin/user_upload/Dokumente/Downloads/TECHNICAL_DOCUMENTS/EPSG_DS_301_V-1-3-0__4_.pdf
14. Sercos III Communication Development Platform // Texas Instruments. 2015. URL: http://www.ti.com/lit/ug/tidu534a/tidu534a.pdf
15. IEEE 1588-2008 – IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems // IEEE Standard Association. 2008. URL: https://standards.ieee.org/standard/1588-2008.html
16. Hibbard J. 5 Real-Time, Ethernet-Based Fieldbuses Compared. 2016. URL: https://www.manufacturingtomorrow.com/article/2016/05/5-real-time-ethernet-based-fieldbuses-compared/8044/
17. A Language-based Approach for Interoperability of IoT Platforms / Gabbrielli M., Giallorenzo S., Lanese I., Zingaro S. P. // Proceedings of the 51st Hawaii International Conference on System Sciences. 2018. doi: https://doi.org/10.24251/hicss.2018.714
18. Integrated and Differentiated Services. URL: https://users.ece.utexas.edu/~ryerraballi/MSB/pdfs/M5L4.pdf

19. Towards Multi-layer Interoperability of Heterogeneous IoT Platforms: The INTER-IoT Approach / Fortino G., Savaglio C., Palau C. E., de Puga J. S., Ganzha M., Paprzycki M. et. al. // Internet of Things. 2018. P. 199–232. doi: https://doi.org/10.1007/978-3-319-61300-0_10
20. Overcoming the Heterogeneity in the Internet of Things for Smart Cities / Kazmi A., Jan Z., Zappa A., Serrano M. // Interoperability and Open-Source Solutions for the Internet of Things. 2017. P. 20–35. doi: https://doi.org/10.1007/978-3-319-56877-5_2
21. OpenFlow-enabled SDN and Network Functions Virtualization // Open Networking Foundation. 2014. URL: https://www.opennetworking.org/wp-content/uploads/2013/05/sb-sdn-nvf-solution.pdf
22. Keyzer M., Loutas N., Goedertier S. Introduction to RDF & SPARQL // Open Data Support. 2014. URL: https://joinup.ec.europa.eu/sites/default/files/document/2015-05/d2.1.2_training_module_1.3_introduction_to_rdf_sparql_v1.00_en.pdf
23. Introduction to Web Ontology Language (OWL) // University of Dublin, Trinity College. URL: https://www.scss.tcd.ie/Owen.Conlan/CS7063/06%20Introduction%20to%20OWL%20(1%20Lecture).ppt.pdf
24. Sousa P. T., Stuckmann P. Telecommunication network interoperability // Telecommunication Systems and Technologies. Vol. II. URL: http://www.eolss.net/sample-chapters/c05/e6-108-22.pdf
25. The internet of things: mapping the value beyond the hype / Manyika J., Chui M., Bisson P., Woetzel J., Dobbs R., Bughin J., Aharon D. // McKinsey & Company. 2015. URL: https://www.mckinsey.com/~/media/mckinsey/business%20functions/mckinsey%20digital/our%20insights/the%20internet%20of%20things%20the%20value%20of%20digitizing%20the%20physical%20world/the-internet-of-things-mapping-the-value-beyond-the-hype.ashx
26. Tikhonov V. I., Taher A., Tykhonova O. Conveyor module resource scheduling in packet based communication channel // Bulletin of the National Technical University "KhPI". A series of "Information and Modeling". 2016. Issue 21 (1193). P. 152–161. doi: https://doi.org/10.20998/2411-0558.2016.21.17
27. Tikhonov V. I., Taher A., Tykhonova O. V. Simulation the algorithm of multimedia data integration in packet based digital channel // Measuring and Computing Devices in Technological Processes. 2016. Issue 2. P. 151–155.
28. Developing the architecture of integrated 5G mobile network based on the adaptation of LTE technology / Tikhonov V., Nesterenko S., Babich Y., Taher A. Q., Berezovsky V. // Eastern-European Journal of Enterprise Technologies. 2017. Vol. 5, Issue 2 (89). P. 42–49. doi: https://doi.org/10.15587/1729-4061.2017.111900
29. Tykhonova O. V. The Ethernet based method of interoperability scope extension in a converged network // Information and Telecommunication Sciences. 2017. Vol. 8, Issue 2. P. 11–17.
30. Vorobiyenko P. P., Tykhonova O. V., Tikhonov V. I. Interoperability Scope Extension in Converged Packet Based Network // The 2nd IEEE International Conference on Information and Telecommunication Technologies and Radio Electronics (UkrMiCo'2017). 2017. P. 497–500.
31. Elg L. Innovations and new technology – what is the role of research? // VINNOVA. 2014. URL: https://www.vinnova.se/contentassets/e5fe05cb13604be7b221f3ddbecb41c3/va_14_05.pdf
32. Tikhonov V. I., Vorobiyenko P. P. Integrated telecommunication technology for the next generation networks // Proceedings of the ITU Kaleidoscope Academic Conference "Building Sustainable Communities". 2013. P. 187–193.