ДОМЕННИЙ АНАЛІЗ

UDC 004.415.2.045 (076.5)

Chebanyuk O.V.
National Aviation University

DESIGNING OF ONTOLOGY FOR THE APPLICATION DOMAIN “DATA TRANSITION PROCESSES IN NETWORKS”

Article represents a process of designing of the ontology for application domain “Data transmission process in networks”. Designed elements of the ontology can be used for defining components reuse [1], processes of requirement elicitation activities [2], refinement of behavioral software models, that are represented as UML diagrams [2]. and many other tasks in software engineering activities. Also for refinement of process defining interconnection between domain entities the domain analysis with designing behavioral models of business process of data transmission in networks, designed according different technologies.

У роботі представлено процес проектування онтології проблемного домену «Передача даних у мережах». Спроектовані елементи онтології можуть використовуватися для визначення повторно використовуваних компонентів [1], при виконанні завдань управління вимогами [2], уточнення поведінкових моделей програмного забезпечення [2] та багатьох інших завдань у різних процесах розробки програмного забезпечення. Також для уточнення характеру взаємозв’язків між сутностями проблемного домена спроектовано поведінкові моделі бізнес-процесів, що ілюструють механізми передачі даних у різних мережах

В работе представлен процесс проектирования онтологии проблемного домена «Передача данных в сетях». Спроектированные элементы онтологии могут использоваться для определения целесообразности повторного использования компонентов [1], выполнении задач управления требованиями [2], уточнения поведенческих моделей програмного обеспечения [2] и ири выполнении других задач разработки программного обеспечения. Для уточнения характера взаимосвязей между сущнстями проблемного домена в работе представлены модели бизнесс-процессов, которые илюстрируют особенности механизма передачи данных в различных сетях.

Keywords: *problem domain process, ontology, network protocol, business modeling, activity diagram, seuience diagram, technologies of data transition through network*

Introduction

A common definition of the term ontology is that it is a formal, explicit specification of a shared conceptualization [1]. A domain ontology focuses on a specific subject, here the system under construction. For requirements engineering the aspect of sharing is of particular interest, it means that stakeholders agree on terminology and term relations. By making use of this information we lower the risk for requirements ambiguity. Also ontology is useful in Agile software development process, when MDD approach is used. One of the ontology purpose is to provide ways of defining whether software behavioral or static models satisfy Model-Driven Engineering requirements namely: completeness, authenticity and being non contradictory.

Concepts defining ontology

An ontology formal definition could be the following:

$O=\{C, HC, HR, A, I\}$ (1)

where C is the set of concepts in a domain. Ontology concepts are the terms the requirements engineer uses in the requirements. This includes actors, components, events, states, etc. of the system under construction.
HC is the set of the hierarchical relations among concepts.
HR is the set of generic relations among concepts.

Relations are links between concepts. We use two kinds of relations:

– A named relation represents a functional relationship between a subject concept and an object concept. The relation name is expected to be a transitive verb. Named relations are used to derive suggestions.

– Anonymous relations simply indicate that two concepts are related. This is used to prefer semantically related suggestions.
A is the set of axioms.

Axioms are relations between concepts with a special meaning.

• An *equivalence* axiom represents the knowledge that two concepts refer to the same phenomenon in the domain (synonyms). A *subclass-of* axiom states that one concept is a subclass of another one. Both kinds of axioms lead to an inheritance of relations from the equivalent or parent concept. The *part-of* axiom imposes a hierarchical structure on the ontology contents, which facilitates navigation.
I is an ontology instances.

Designing of ontology elements for the application domain "transmitting data in networks"

1. Define a set of concepts, creating the ontology vocabulary

1.1 Internetworking

An internetwork is a collection of individual networks, connected by intermediate networking devices, that functions as a single large network. Internetworking refers to the industry, products, and procedures that meet the challenge of creating and administering internetworks.

1.2. Quality of Service (QoS)

QoS refers to the capability of a network to provide better service to selected network traffic over various technologies, including Frame Relay, Asynchronous Transfer Mode (ATM), Ethernet and 802.1 networks, SONET, and IP-routed networks.

1.3. Open System Interconnection model

The Open System Interconnection (OSI) reference model describes how information from a software application in one computer moves through a network medium to a software application in another computer. The OSI reference model is a conceptual model composed of seven layers, each specifying particular network functions.

1.4 Local Area Network and Wide Area Network (LAN&WAN).

A LAN is a high-speed data network that covers a relatively small geographic area.

It typically connects workstations, personal computers, printers, servers, and other devices. LANs offer computer users many advantages, including shared access to devices and applications, file exchange and communication between connected users.

1.5. Wide Ares Network (WAN) is a data communications network that covers a relatively broad geographic area and that often uses transmission facilities provided by common carriers.

WAN technologies generally function at the lower three layers of the OSI model: the physical layer, the data link layer, and the network layer.

1.6. Multiplexing is a process in which multiple data channels are combined into a single data or physical channel at the source (can be implemented at any of the OSI layers).

1.7. Demultiplexing is the process of separating multiplexed data channels at the destination.

One example of multiplexing is when data from multiple applications is multiplexed into a single lower-layer data packet.

1.8. Virtual Private Network

A VPN (Virtual Private Network) is a generic term that describes any combination of technologies that can be used to secure a connection through an otherwise unsecured or untrusted network

1.9 Carrier sense

Each station continuously listens for traffic on the medium to determine when gaps between frame transmissions occur.

1.10 Multiple access

Stations may begin transmitting any time they detect that the network is quiet (there is no traffic).

1.11 Collision detect

If two or more stations in the same CSMA/CD network (collision domain) begin transmitting at approximately the same time, the bit streams from the transmitting stations will interfere (collide) with each other, and both transmissions will be unreadable. If that happens, each transmitting station must be capable of detecting that a collision has occurred before it has finished sending its frame. Each must stop transmitting as soon as it has detected the collision and then must wait a quasirandom length of time (determined by a back-off algorithm)

The worst-case situation occurs when the two most-distant stations on the network both need to send a frame and when the second station does not begin transmitting until just before the frame from the first station arrives. The collision will be detected almost immediately by the second station, but it will not be detected by the first station until the corrupted signal has propagated all the way back to that station. The maximum time that is required to detect a collision (the collision window, or "slot time") is approximately equal to twice the signal propagation time between the two most-distant stations on the network.

2. Analysis of business processes in ontology "Data transmition protocol in networks"

Principles of data transition by network protocols without connection establishment

The internet protocol implements two basic functions: addressing and fragmentation [3]. The internet modules use the addresses carried in the internet header to transmit internet datagrams toward their destinations, to fragment and reassemble internet datagrams when necessary for transmission through "small packet" networks.

The internet protocol treats each internet datagram as an independent entity unrelated to any other internet datagram. There are no connections or logical circuits (virtual or otherwise).

The internet protocol uses four key mechanisms in providing itsservice: Type of Service, Time to Live, Options, and Header Checksum.

The internet protocol does not provide a reliable communication facility. There are no acknowledgments either end-to-end or hop-by-hop. There is no error control for data, only a header checksum. There are no retransmissions. There is no flow control.

Investigation

2.1. The mechanism of data transition by network protocols without connection establishment

1. The sending application program prepares data and calls on to its local internet module in order to send data as a datagram. The destination address and other parameters are passed as arguments of the call.

2. The internet module prepares a datagram header and attaches the data to it. The internet module determines a local network address for this internet address, in this case it is the address of a gateway.

3. It sends this datagram and the local network address to the local network interface.

4. The local network interface creates a local network header, and attaches the datagram to it, then sends the result via the local network. Packet is compressed in order to reduce the size and improve its quality.

5. The datagram arrives at a gateway, the local network interface strips off this header, and turns the datagram over to the internet module. The internet module determines from the internet address that the datagram is to be forwarded to another host in a second network. The internet module determines a local net address for the destination host. It calls on the local network interface for that network to send the datagram.

6. This local network interface creates a local network header and attaches the datagram sending the result to the destination host.

7. At this destination host the datagram is stripped of the local net header by the local network interface and handed to the internet module [4].

8. The internet module determines that the datagram is for an application program in this host. It passes the data to the application program in response to a system call, passing the source address and other parameters as results of the call.

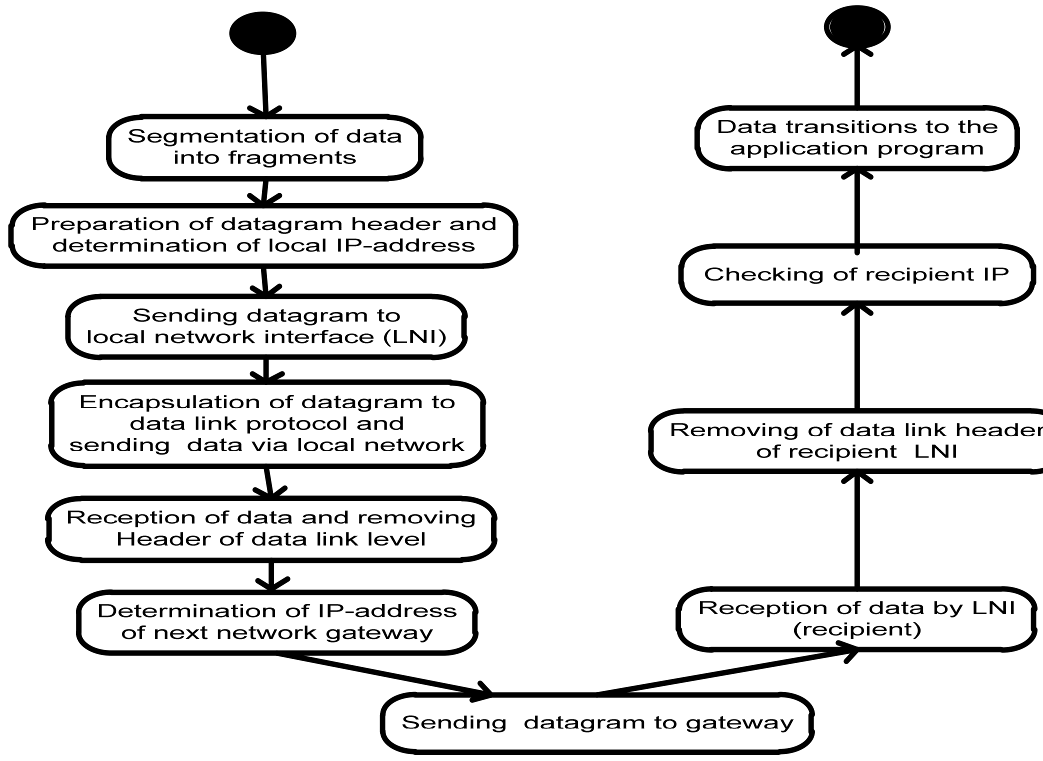

Segmentation of data into fragments
Preparation of datagram header and determination of local IP-address
Sending datagram to local network interface (LNI)
Encapsulation of datagram to data link protocol and sending data via local network
Reception of data and removing Header of data link level
Determination of IP-address of next network gateway
Sending datagram to gateway
Reception of data by LNI (recipient)
Removing of data link header of recipient LNI
Checking of recipient IP
Data transitions to the application program

Fig 1. Activity diagram of data transition in network protocols without establishment connection

2.2 Principles of data transition by network protocols with connection establishment

TCP is a connection-oriented, end-to-end reliable protocol designed to fit into a layered hierarchy of protocols which support multi-network applications. The TCP provides for reliable inter-process communication between pairs of processes in host computers or communication networks. TCP assumes it can obtain a simple, potentially unreliable datagram service from the lower level protocols.

The TCP fits into a layered protocol architecture just above a basic Internet Protocol which provides a way for the TCP to send and receive variable-length segments of information enclosed in internet datagram "envelopes".

A TCP connection is managed by an operating system through a programming interface that represents the local end-point for communications, the *Internet socket*. During the lifetime of a TCP connection the local end-point undergoes a series of state changes (on the diagram states are marked in bold):

LISTEN server represents waiting for a connection request from any remote TCP and port.

SYN-SENT client represents waiting for a matching connection request after having sent a connection request.

SYN-RECEIVED - server represents waiting for a confirming connection request acknowledgment after having both received and sent a connection request.

ESTABLISHED (both server and client) - represent an open connection, data received can be delivered to the user. The normal state for the data transfer phase of the connection.

FIN-WAIT-1 (both server and client) - represents waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.

FIN-WAIT-2 (both server and client) - represents waiting for a connection termination request from the remote TCP.

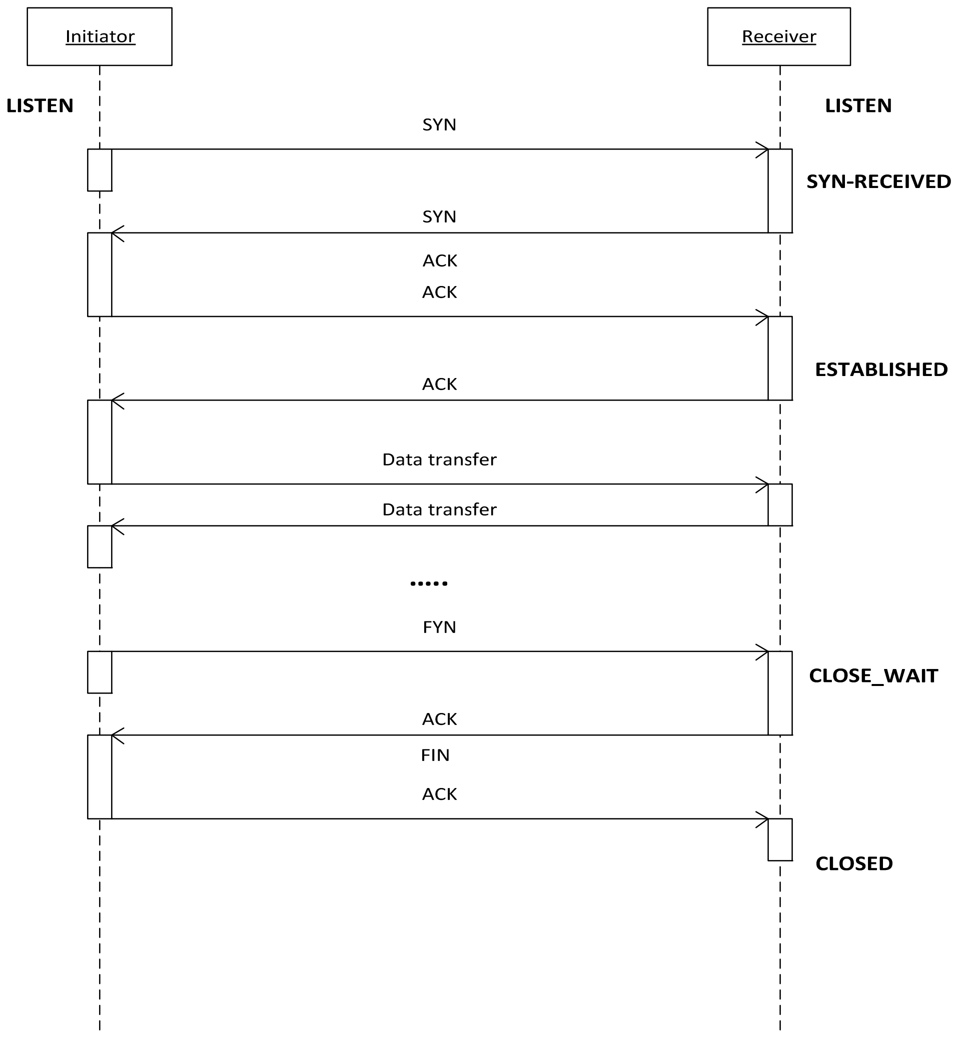
CLOSE-WAIT (both server and client) - represents waiting for a connection termination request from the local user.

CLOSING (both server and client) - represents waiting for a connection termination request acknowledgment from the remote TCP.

LAST-ACK (both server and client) - represents waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request).

TIME-WAIT (either server or client) - represents waiting for enough time to pass to be sure the remote TCP received the acknowledgment of its connection termination request. [According to RFC 793 a connection can stay in TIME-WAIT for a maximum of four minutes known as a MSL (maximum segment lifetime).]

CLOSED (both server and client) - represents no connection state at all [3].

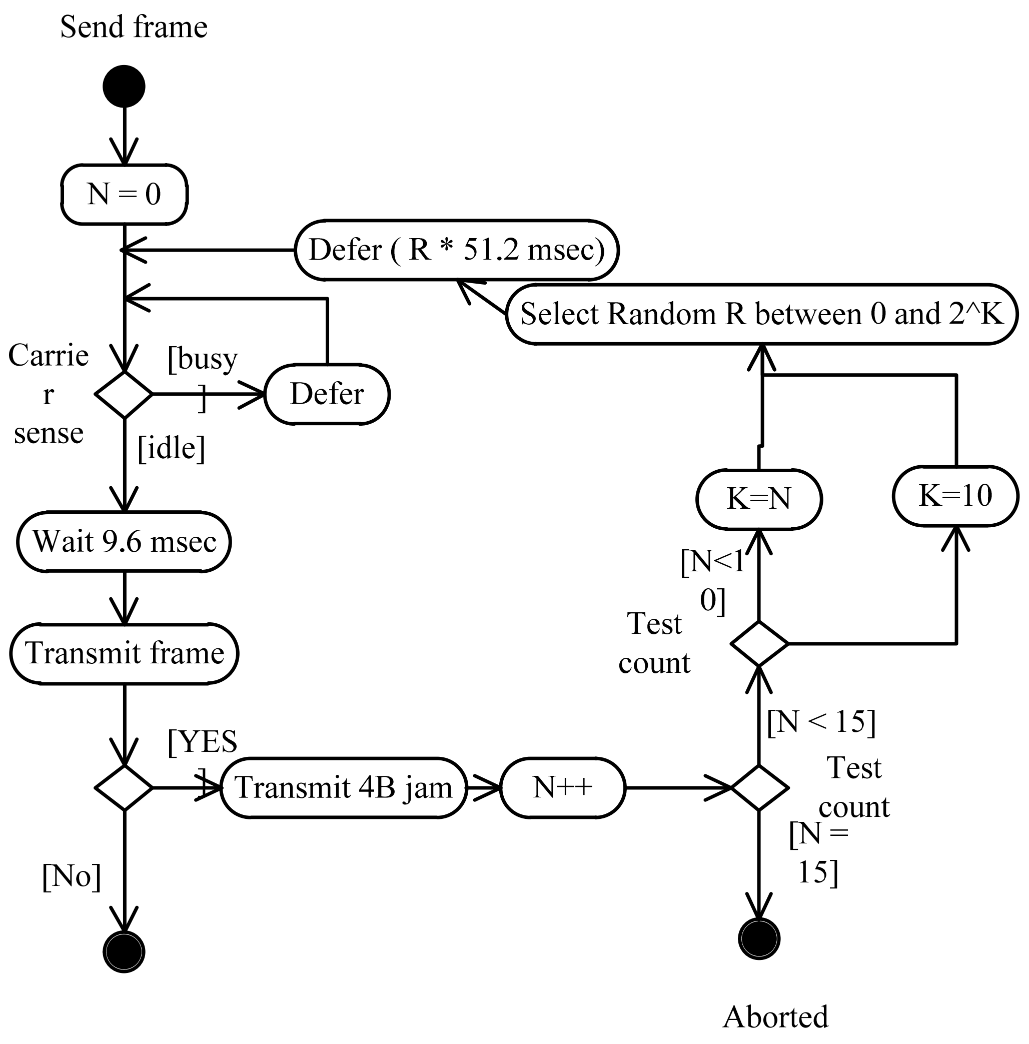


Pict. 2 Sequence diagram of messages exchanging in TCP protocol

Consider the peculiarities of data transmition process an Local Area Network (LAN). Main problem of LAN is organizing data transmition process without collisions. Different technologies propose different methods. We will consider Ethernet and Token Ring

2.3 The CSMA/CD Access Method collision detection in Ethernet

The CSMA/CD protocol was originally developed as a means by which two or more stations could share a common media in a switch-less environment. The protocol does not require central arbitration, access tokens, or assigned time slots to indicate when a station will be allowed to transmit. Each Ethernet MAC determines for itself when it will be allowed to send a frame. before attempting to retransmit the frame. The CSMA/CD access rules are summarized by the protocol's acronym[4,5]:



Pict. 3 Activity diagram of resolving situation of collision in Ethernet network

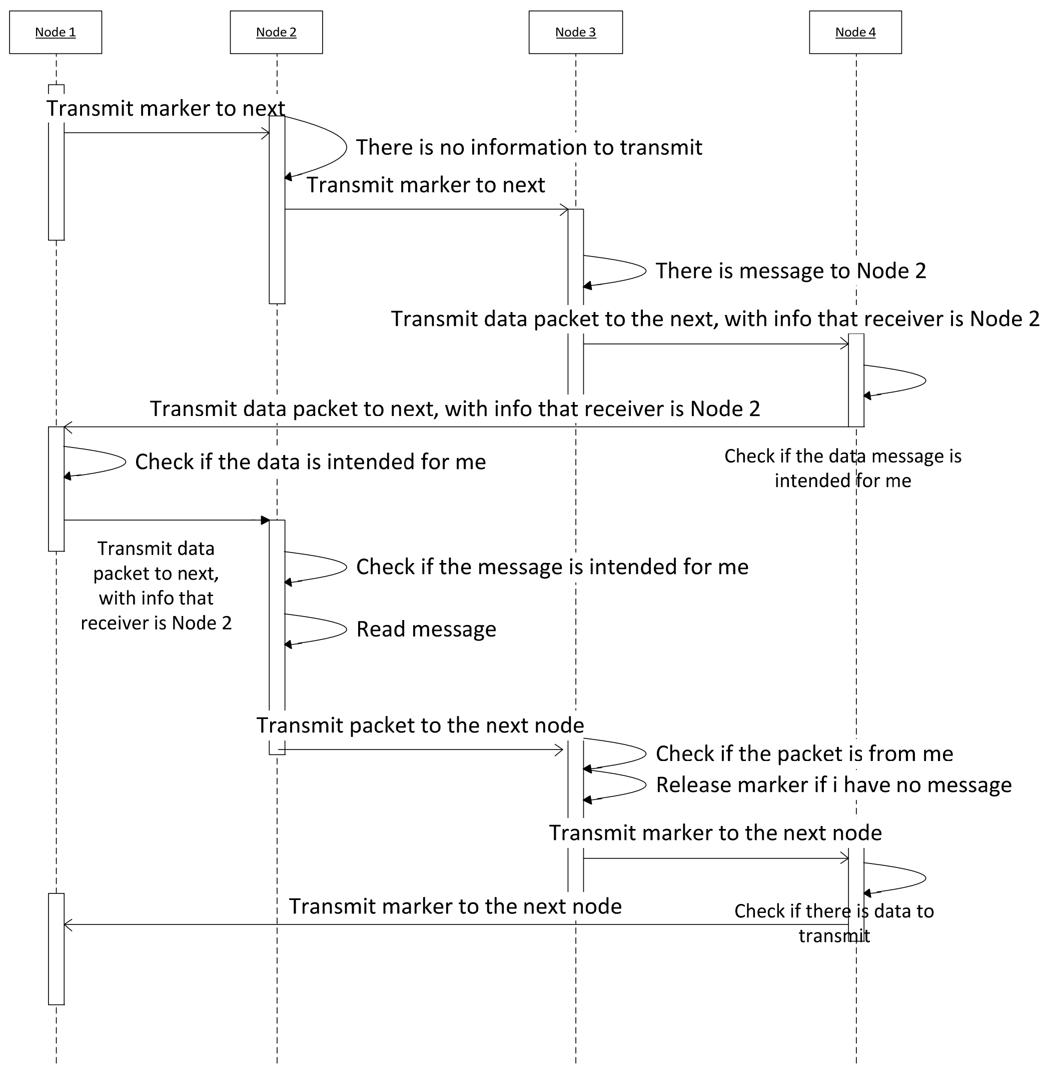
2.4 Token Ring

Token Ring and IEEE 802.5 are two principal examples of token-passing networks. Token-passing networks move a small frame, called a token, around the network. Possession of the token grants the right to transmit. If a node receiving the token has no information to send, it passes the token to the next end station. Each station can hold the token for a maximum period of time.

Unlike CSMA/CD networks (such as Ethernet), token-passing networks are deterministic, which means that it is possible to calculate the maximum time that will pass before any end station will be capable of transmitting. This feature make Token Ring networks ideal for applications in which delay must be predictable and robust network operation is important. Factory automation environments are examples of such applications.

Token Ring networks use a sophisticated priority system that permits certain user-designated, high-priority stations to use the network more frequently. Token Ring frames have two fields that control priority: the priority field and the reservation field.

Only stations with a priority equal to or higher than the priority value contained in a token can seize that token. After the token is seized and changed to an information frame, only stations with a priority value higher than that of the transmitting station can reserve the token for the next pass around the network. When the next token is generated, it includes the higher priority of the reserving station. Stations that raise a token's priority level must reinstate the previous priority after their transmission is complete[4,5].


Node 1
Node 2
Node 3
Node 4
Transmit marker to next
There is no information to transmit
Transmit marker to next
There is message to Node 2
Transmit data packet to the next, with info that receiver is Node 2
Transmit data packet to next, with info that receiver is Node 2
Check if the data is intended for me
Check if the data message is intended for me
Transmit data packet to next, with info that receiver is Node 2
Check if the message is intended for me
Read message
Transmit packet to the next node
Check if the packet is from me
Release marker if i have no message
Transmit marker to the next node
Transmit marker to the next node
Check if there is data to transmit

Pic. 4 Sequence diagram of data transmition with token usage

2.5 General algorithm of ATM networking (Fig. 5)

1. Connect to ATM network.
2. Get an identifier number of the nearest ATM switch.
3. Form one's own address beginning with the identifier number of the switch.
4. The switch enters data into its table.
5. Divide data into segments.
6. Pack data.
7. Send an initializing request to the nearest switch. Get answer from the nearest switch about initiation. If initiation is possible then switch sends a request to the receiver, otherwise go to the item 2.
8. Get the request that the user accepted the connection.
9. Send the acknowledgment that the connection is accepted
10. Data is transmitted.
11. The connection is closed.
12. The subscriber sends a message about breaking the connection.
13. Another subscriber confirms breaking the connection [6].

In the ATM network switches have identifying numbers and the addresses of the subscribers connected to these switches have prefixes of the identifying numder of the switch they are connected to. Thus the switch contains identifying numbers of other switches and addresses of subscribers connected to it. Such hierarchical approach to assigning addresses allows to reduce the number of records in the switch tables (fig 2.3).

Conclusion elements of ontology

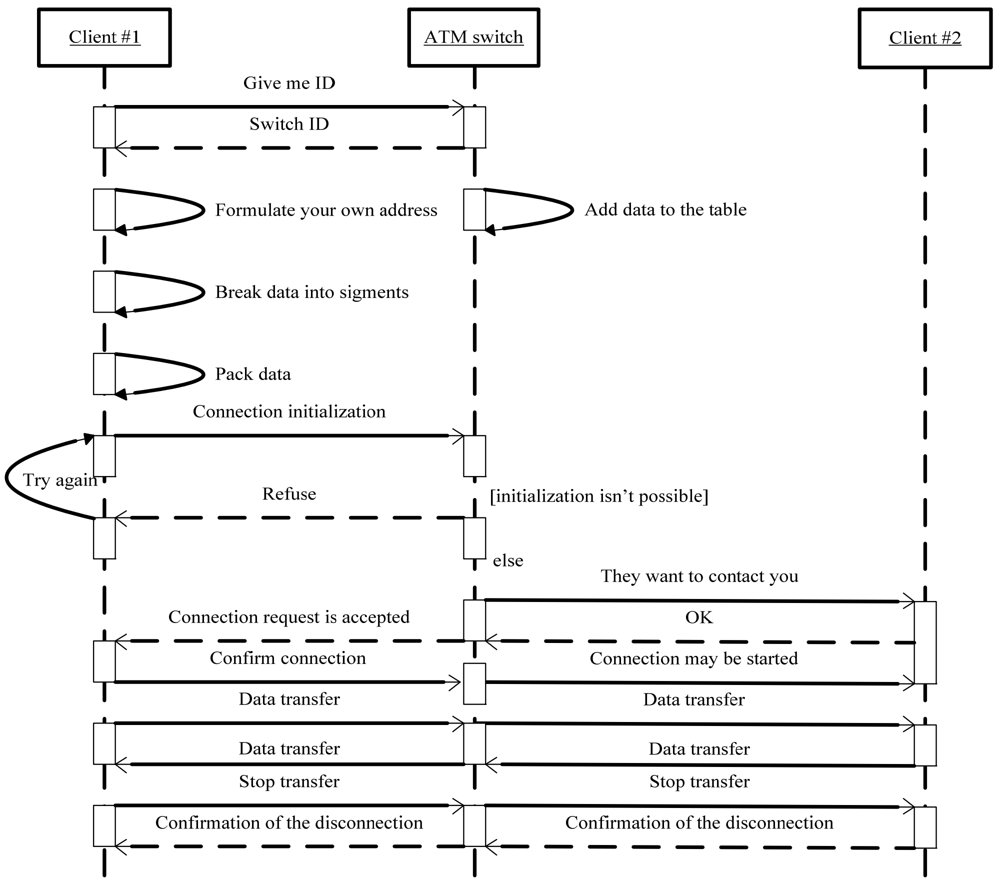


Fig. 5. Sequence diagram: ATM networking

Data are divided into segments and grouped into packets containing 53 octets having 48 bytes of data and 5 bytes of a header.

When the logical connection is estabished a special packet is sent to the switch. The packet contains data about the recipient address, quality of service (jitters, available resources) and so on. If the switch can serve the connection it initiates a link and sends a request to the recipient, otherwise it breaks the connection. Then the receiver informs about getting a request and the sender informs him about handshaking (when the recipient received a packet the sender may have changed his mind). Duplex data transmission is produced. Subscribers may be disconnected on the request of one of them. The other subscriber receives his message and sends a request to acknowledge the disconnection.

Conclusions Material to design an ontology "Data transmission in network" is represented in this article. Behavioral software models of main processes in application domain, allows to increase the effectiveness of software development process according to Agile methodology namely:

- activity diagram of data transition in network protocols without establishment connection;
- sequence diagram of messages exchanging in TCP protocol;
- activity diagram of resolving situation of collision in Ethernet network;
- sequence diagram of data transition with token usage;
- sequence diagram: ATM networking.

Using of these diagrams allows to refine both behaovioral and static software models while software development process. Also ontology vocabulary helps to elicitate requirements, define synonims and improve documentation. One more function of these diagram is to define dependency components in frameworks or namespaces.

References

1. An Approach for Software Component Reusing based on Ontological Mapping / Shi-Kuo Chang, Francesco Colace, Massimo De Santo, Emilio Zegarra1, Yong Jun Qie1 // The Twenty-Fourth International Conference on Software Engineering & Knowledge Engineering. – 2012. – 68 p.

2. Activities and Tecniques for a requirement conceptualization process / Alejandro Hossian, Ramon Garcia-Martinez Phases // STE. – 2013. – 38 p.

3. U.S. Patent No. 8.634.716 "Data transmission network" / Tichborne, Franklin. – 21th of January 2014.

4. U.S. Patent No. 8.625.419.7 "Method and device for adjusting transmission of transport network data" / Ding Chiwu, Huaping Qing. – January 2014.

5. U.S. Patent No. 8.631.450. "Broadband local area network" / Bernath Brett. – 14th of January 2014.

6. U.S. Patent No. 8.630.302. "System and network for deriving voice channels on a broadband communication line" / Bossemeyer Jr, Robert Wesle. – 14th of January 2014.

Information about author:

Chebanyuk Elena Viktorivna – PhD, Software Engineering Department of the Institute of Computer Information Technologies of the National Aviation University. Scientific interests: domain analysis, computer networks.

E-mail: chebanyuk.elena@gmail.com