Journal of Theoretical and Applied Information Technology
20th May 2013. Vol. 51 No.2
© 2005 - 2013 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

TOWARD A CONCEPTUAL BASE FOR PROTOCOL ENGINEERING

[1]SABAH AL-FEDAGHI, [2]ALAA AL-SAQA

[1]Assoc. Prof., Department of Computer Engineering, KUWAIT UNIVERSITY, KUWAIT

[2]Engr., Department of Computer Engineering, KUWAIT UNIVERSITY, KUWAIT

[1]E-mail: sabah@alfedaghi.com, [2]eng_alaa_alsaqa@hotmail.com

ABSTRACT

A protocol refers to a set of rules governing the exchange of data between entities. Several notations are utilized in the specification of protocols, including different types of diagrams such as flowchart-like depictions, UML sequence diagrams, and state transition diagrams. This paper is a contribution to this area, proposing a diagrammatic methodology for protocol specification. It is based on the notion of flow of "primitive" things in a system with six stages: creation, release, transfer, arrival, acceptance, and processing. The aim is to introduce a conceptual and complete description of basic streams of flow among entities and stages including "crossing points" that need rules of data transfer. The resultant specification is a map over which a protocol can be superimposed.

Keywords: *Protocol, Network, Conceptual model, Communication, Rules*

1. INTRODUCTION

Any communication model comprises a source that generates the data to be transmitted, a transmitter that converts the data into transmittable signals, a transmission system that carries the data, a receiver that converts received signals into data, and a destination that takes incoming data [1]. Communication-related handling of information involves such operations as the release, transfer, arrival, and acceptance of exchanged information. This requires a set of rules governing data exchange, referred to as a *protocol*, which can be defined as a set of rules governing the exchange of data between entities [2]. According to [3],

An *entity* is anything capable of sending or receiving information and a system is a physically distinct object that contains one or more entities… To reduce communication systems design complexity, most systems are organized as a series of layers or levels, each one built upon its predecessor… The purpose of each layer is to offer certain services to the higher layers, shielding those layers from the details of how the offered services are actually implemented. A layer n entity on one system carries on a conversation with a layer *n peer entity* on another system. The rules and conventions used in this conversation are collectively known as the *layer n protocol* [4]. (Italics added).

For example, the OSI Reference Model presents standards for linking computers with seven protocol layers [2]. Each n layer comprises active entities capable of sending or receiving information to/from their layer n peer entities in another system.

The entities in layer (N) implement a service used by layer (N+1). In this case layer (N) is called service provider and layer (N+1) is called service user. Layer (N) may use the services of layer (N+1) in order to provide its service. Services are available at service access points (SAP). The layer (N) SAPs are the places where layer (N+1) can access services. Each SAP has an address that uniquely identifies it. [3]

In OSI Reference Model (Figure 1), layers from 1 to 3 are network dependent and concerned with the protocols associated with the data communication network used to link two communicating systems. Layers 5 to 7 are application oriented and concerned with the protocols that allow two end-user application processes to interact with each other.

© 2005 - 2013 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

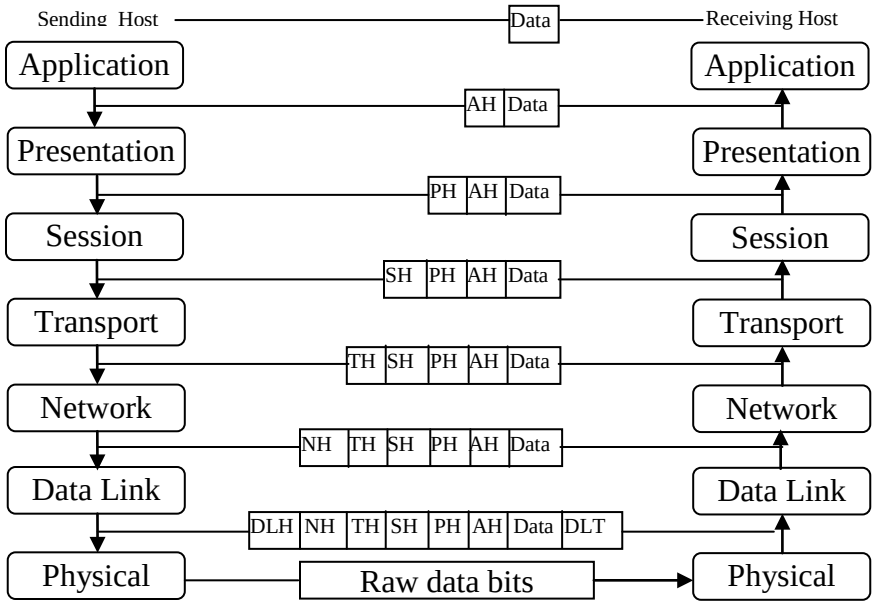


Figure 1. OSI Reference Model (From [7])

The intermediate transport layer (4) hides the detailed operation of the lower network-dependent layers from the upper application-oriented layer [5][6].

Another example of a layered communication system is the TCP/IP model, which consists of the four layers *link* (device driver and interface card), *network* (e.g., IP protocols), *transport* (e.g., the TCP protocol), and *application* (includes FTP and DNS) [8][9][10][11].

When one starts with protocols that work at the upper layers, each set of data is wrapped inside the next lower layer protocol [12]. In Figure 1, every layer adds an additional piece of information to the message it is transmitting. Such an arrangement is referred to as *encapsulation*. The additional information appears as a header (e.g., TH = Transport Header). The data link layer also adds a trailer to its data, so each layer is encapsulated in the next layer. On the side of the receiving host, each layer removes the additional piece of information, and such a process is referred to as *decapsulation* [7][13][14][15].

The following general example, summarized from [12], assumes that the protocol stack being used is TCP/IP with an FTP client program for files transferring from/to an FTP server.

1. The FTP client program is started on the sending computer.
2. The address and port of the server are selected.
3. A request is made to connect to the server.
4. The application layer sends information through the presentation layer to the session layer to open a connection.
5. The session layer negotiates for a connection. There are several synchronization signals to establish the connection:
   a. The session layer of the client sends a data packet signal to the transport layer.
   b. The transport layer adds a header to the packet indicating the source and destination ports.
   c. The network layer adds source and destination IP addresses.
   d. The datalink layer determines the hardware address of the computer to which the data are being sent.
   e. The information is transmitted across the hardware layer.
   f. The FTP server sees the ethernet frame matching its address and strips the ethernet header information and sends it to the network layer.
   g. The network layer examines the IP address information, strips the IP header, and sends the information to the transport layer.
   h. The transport layer strips the TCP header and sends the information to the appropriate program servicing the requested port.

© 2005 - 2013 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

i. The session layer in the FTP program conducts a series of data exchanges through all the lower layers to the client computer until a session is established.

6. At this point information may be sent through several FTP commands between the client and the server. Every transmission passes through the network layers from the application layer down to the hardware layer, then back up through the layers on the receiving computer.
7. When the client decides to terminate the session, the layer is informed by the higher layers and negotiates for the closing of the connection.

Several notations are utilized in the specification of protocols, including different types of diagrams such as flowchart-like depictions, UML sequence diagrams, and state transition diagrams [3][16][17]. This paper is a contribution to this area, proposing a diagrammatic methodology for protocol specification. It is based on the notion of flow of "primitive" things in a system with six stages: creation, release, transfer, arrival, acceptance, and processing.

The aim of the paper is to introduce a conceptual and complete description of basic streams of flow among entities and stages including "crossing points" that need "protocolization", not in the sense of notarization, but to mean "providing rules of transfer." It is a map over which a protocol can be superimposed.

2. MOTIVATIONAL EXAMPLES

This section displays a sample diagrammatic description used in modeling protocols. While we are focusing here on a specific model, it is not our observation that protocol specification, in general, lacks the presence of a conceptual framework for expressing basic primitives and operations involved in any convention that facilitates transactions. While describing the sample standard model discussed in this section in detail is beyond the scope of this paper, the discussion aims at demonstrating the need for a more precise methodology of description that is beyond the level of narratives and sketches.

In this context a *primitive* is a unit of information that is sent from one layer to another [18]. There are four *classes of primitives*: Request, Confirm, Indication, and Response, as shown in Figure 2. *Request* is a request for services from another layer, *Confirm* is the acknowledgment, *Indication* is notification of the information to the layer requesting the service, and *Response* is acknowledgment of the indication. The primitive includes protocol layer ID, protocol ID, primitive class, primitive name, and parameters. The protocol ID specifies to which protocol entity this primitive should be sent, e.g., IEEE 802.11 or IEEE 802.3 [19][20][21].

To base the description on a firm conceptual basis, the notion of *unit of information*, described as "sent from one layer to another," seems in need of further elaboration. In our proposed framework, this "unit of information" is "a thing that flows" (denoted as a *flowthing*) with six basic *exclusive* operations: created, released, transferred, arrived, accepted, and processed, as shown in Figure 3. Exclusiveness here indicates that if the unit of information is in one of these six states (also called stages), then it is not in any of the other five states. The "flow system" of a type of flowthing (denoted as *flowsystem*) represents the legal sequence of stages that a process can exhibit.

"Things that flow" are the units in a Flow Model (FM) that has been utilized in many applications [23][24][25][26][27]. Accordingly, Figure 3 can now be presented as shown in Figure 4.

In Figure 4, each of the two layers (called *spheres* in FM) has four flow systems that correspond to the flowthings: Request, Confirmation, Indication, and Response. First, a request is created in layer n (circle 1 in Figure 4), then it is released and transferred (circle 2) to layer n-m.

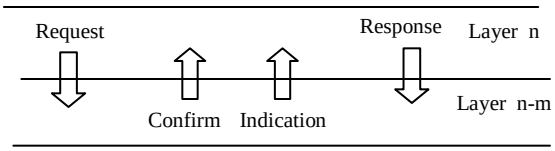


Figure 2. Primitives (from [22])

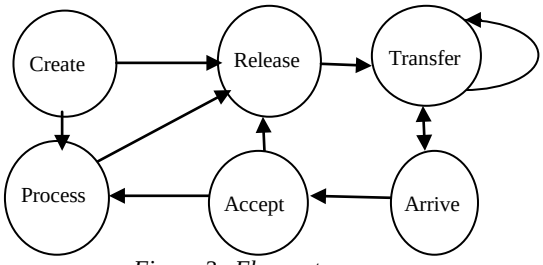


Figure 3. Flowsystem

Journal of Theoretical and Applied Information Technology
20th May 2013. Vol. 51 No.2
© 2005 - 2013 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

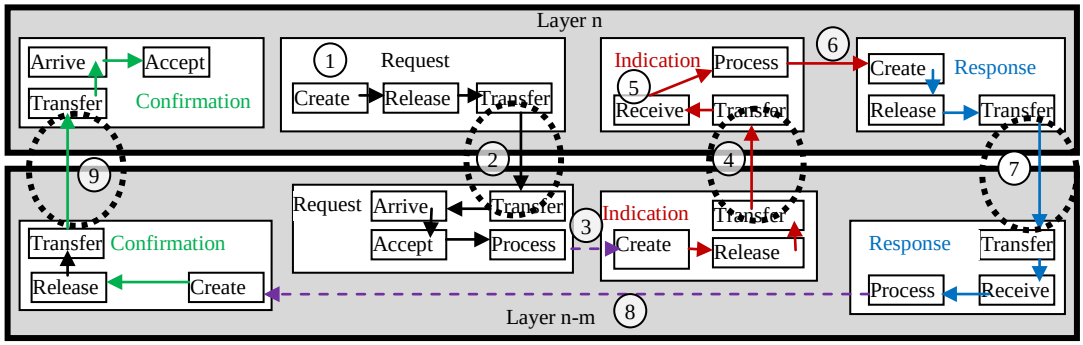


Figure 4. FM-Based Description That Corresponds To Figure 2

Conceptually, stage Released is different from stage Transfer, as in the case of sending an email that is not actually transferred but is waiting for channel availability (e.g., in an output buffer). Transfer is the interface module in the sphere that interacts with the channel.

In layer n-m, arrived requests may not be accepted. If accepted, the request is then processed; this triggers (circle 3) the creation of a confirmation. In FM, flows of data are represented by solid arrows, while triggering is indicated by dashed arrows. Indication is transferred to layer n (circle 4). When arriving flowthings are always accepted, Arrival and Acceptance stages can be combined in one state called Receive, as shown in the Indication flowsystem of layer n (circle 5). Processing Indication triggers (circle 6) the creation of Response, which flows (circle 7) to layer n-m, and this in turn triggers (circle 8) the creation of Confirmation, which flows (circle 9) to layer n.

Figure 4 is certainly more complete than the sketch of Figure 2. It represents what we call a conceptual map that can provide a base for identifying "protocolization" crossing points (the four dotted ovals in Figure 4, which are roughly sketched by the four thick arrows in Figure 2).

"Protocolization" points are important for building "rules of transferring", but also, "rules of communication" need identification of creation points since the language and formatting of the "units of information" are decided at these points. FM-based description draws the entire lifecycle of these units that are to be shipped out or obtained from different spheres. The methodology portrays whole supply chains ready to embrace all types of rules of communication or otherwise. Along with basic drawing items such as arrows and flowsystems, we can also superimpose other tools such as timing and synchronization, logic (e.g., AND, OR), and so forth that can be borrowed from current diagrammatic methods.

3. SPHERES AND SUB-SPHERES

Teraoka et al. [21] also introduced Abstract Entity (AE) to "achieve link independency of the link indications" in addition to the Protocol Entity (PE) that processes a specific protocol used in the conventional protocol-layering model (see Figure 5) showing AEs and PEs using primitives. Conceptually, this introduces the notion of "entities" in layers. Figure 6 shows the FM description that includes two AEs. Multiple levels can appear in such type of modeling.

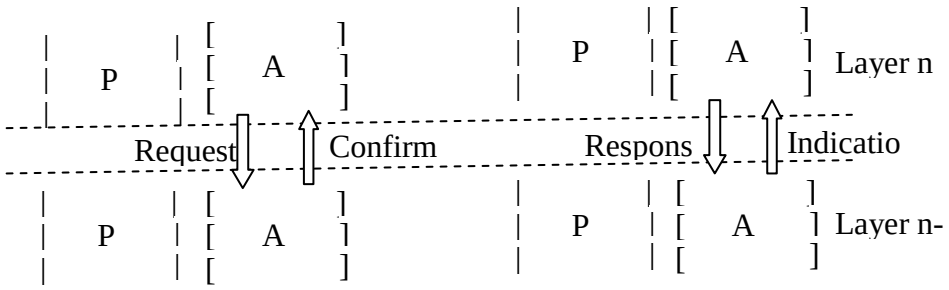


Figure 5. AE And PE With Primitives (From [21])

Journal of Theoretical and Applied Information Technology
20th May 2013. Vol. 51 No.2
© 2005 - 2013 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

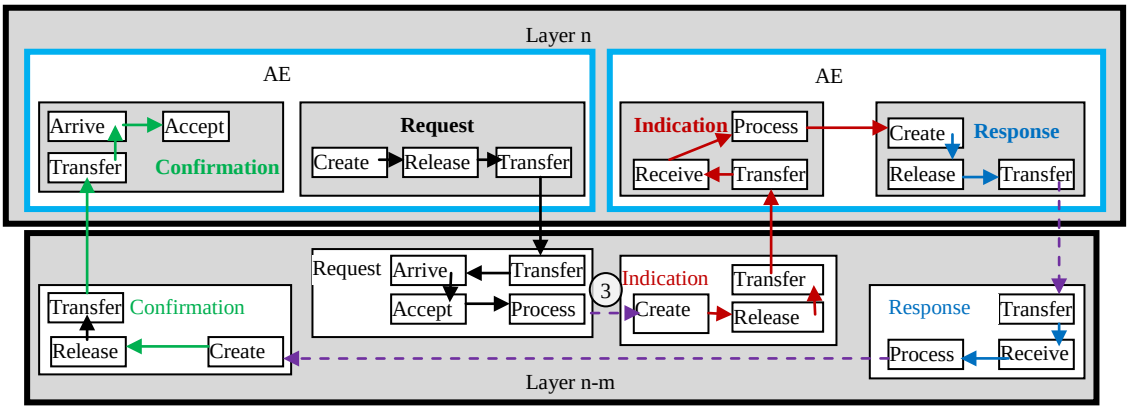


Figure 6. Sub-Spheres

4. A COMPLETE PICTURE OF PROTOCOL OPERATION: SEQUENCE DIAGRAM AND FINITE STATE

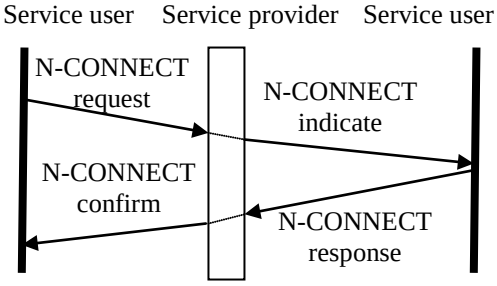
Hekmat, of PragSoft Corporation [7], uses a sequence diagram and final state diagram to draw a complete picture of the way a protocol operates.

> It is worth noting the complementary nature of sequence diagrams and state transition diagrams. The former specifies a service protocol from an outside observer's point of view, while the latter describes the same protocol from a station's point of view. The two notations, combined, provide *a complete picture* of how a protocol operates. (Italics added) [7]

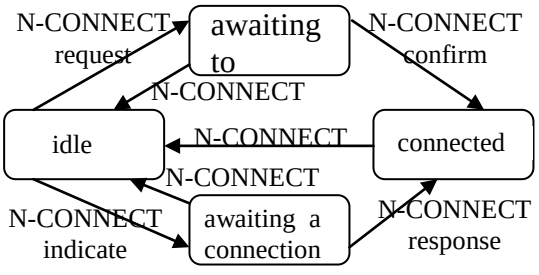
Figure 7(a) shows an example of a sequence diagram for a request for connection at the network layer [7]. A service user issues a request for a connection to a peer service user. The service provider sends a connection indication to the peer service user. The peer user responds to the service provider, confirming the cycle with the original user.

In Figure 7(b), Hekmat [7] provides a finite state diagram that describes the states of "a station at the network layer".

> According to the diagram, assuming that a station is in the idle state, if it issues a connection request to another station, it enters the attempting to connect state where it waits for a connection to be confirmed, in which case it moves to the connected state, or disconnected, in which case it returns to the idle state. A similar scenario applies to an incoming connection which starts with the station receiving a connection indication. [7]



(b) Sequence diagram



(a) FSM diagram

Figure 7. A Complete Picture Of Protocol Operation Using Sequence Diagram And Finite State (From [7])

Journal of Theoretical and Applied Information Technology
20th May 2013. Vol. 51 No.2
© 2005 - 2013 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

We can argue that Figures 7 (a) and (b) do not represent a complete picture. Additionally, the picture is fragmented and vague. Figure 8 shows the corresponding FM-based specification. A request is created (circle 1), and transferred (circle 2) to the service provider, who processes the request, triggering (circle 4) the creation of indication. Since a user can create requests and respond to requests from other users, the sequence indicated in circles 1, 2, 3, and 4 from a user are mirrored by the sequence indicated by circles A, B, C, and D from the other user. Thus, an indication flows to the user (circles 5 and E), causing the creation of a response (circles 6 and G). The response, in turn, flows to the provider (circles 7 and F) to cause the creation of a confirmation that flows to the requester (circles 8 and I).

While Figure 8 provides a complete picture of "how a protocol operates," it lacks the notion of *state*, which may be needed. Additionally, it is possible to represent the service user as one sphere since flowsystems in different users' spheres mirror each other.

transferred, arrive, and be received. "Flow" in FM is not a mere movement in space or time; rather, it can mean a change, a transformation in appearance or condition. A state itself can only be *created* or *processed* (= changed. For example, a door's state can be Open or Closed when it appears as a component in a sphere. Creation, here, is a conceptual phenomenon within a sphere and not necessarily an ontological existence. For example, states can be created or processed (e.g., an open door is changed to half-open) but cannot directly flow to other spheres.

In the example under consideration, the modeler specifies the state's "values" as Idle, Waiting to connect, Waiting for connection, and Connected. Different stages of flow in the FM representation cause the *creation* of these states in the flowsystem of states in the service user's sphere.

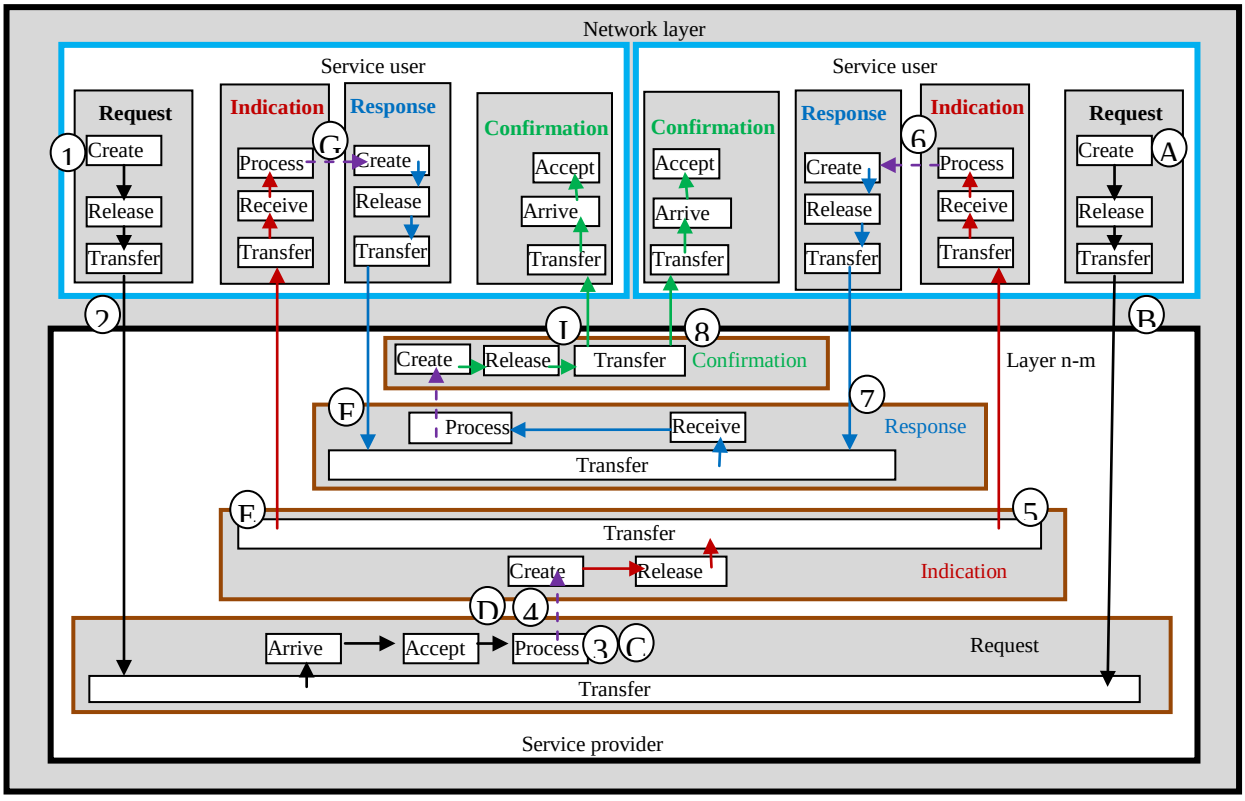


Figure 8. FM Description That Corresponds To The Given Example

In FM a *state* is a flowthing. A flowthing is a thing that can be created, processed, released, and

Accordingly, we redraw Figure 8 as shown in Figure 9. We now have a sphere of one user who plays both roles, those of requester and of one who

© 2005 - 2013 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

responds to requests. As we did previously, we start by creating a request (circle 1). Assuming an original state of Idle, release of the request means requiring transfer by the user (the state of "waiting to connect", circle 2). The different states are denoted by ellipses. Note that the state's flowsystem in the service user's sphere comprises a single stage: creation.

Actual transfer of the request changes the state to Connected (circle 3). When transfer is finished, the state becomes "waiting for connection" (waiting for a response) (circle 4).

The service provider may send an indication (circle 5) that is facilitated by connection to the user, which makes the user's state connected (circle 6).

The user may create a response (circle 7) and release it, changing the state to "waiting to connect" (circle 8). Transferring the response changes the state to "Connected" (circle 9). Transferring the response to the provider (circle 10) causes confirmation to be sent to the user (circle 11).

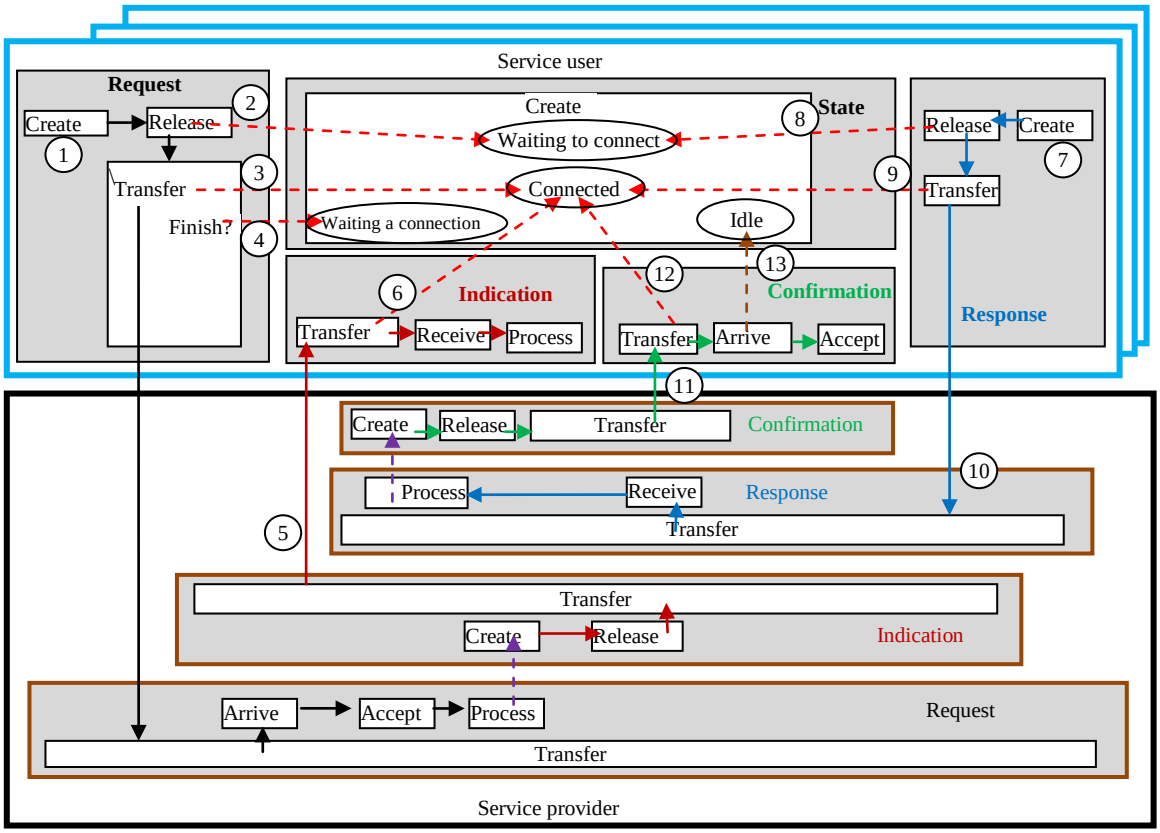


Figure 9. FM Description Incorporating The Notion Of State

This transfer changes the state to "Connected" (circle 12), and when this connection is finished and confirmation arrives at the user, the state changes to "Idle" (circle 13).

5. SIGNALING

Signaling refers to the exchange of control information between the components of a network in order to establish, manage, and disconnect calls [7]. Subscriber signaling refers to the signals exchanged between a subscriber and a local exchange. Reference [7] gives as an example the sequence diagram shown in Figure 10 illustrating the signals exchanged for establishing a call between two subscribers. The following description is closely summarized from [7].

Assuming that initially both subscribers have their phones on-hook, the calling subscriber sends an off-hook signal to the local exchange by lifting the receiver. The switch activates an audible dial tone. The subscriber dials a number, and each dialed digit is signaled to the local exchange.

© 2005 - 2013 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

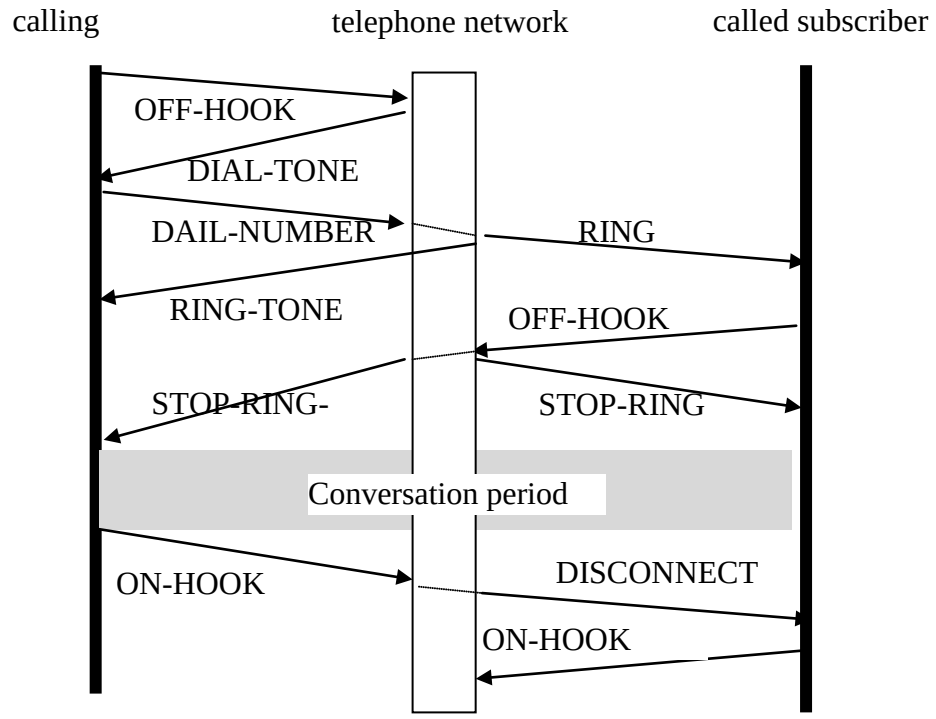


Figure 10. Sample Subscriber Signaling Scenario (From [7])

The local exchange applies a ringing signal to the called subscriber's loop. When the called subscriber lifts the receiver, this causes the local exchange to stop the ringing, and it is propagated back to the calling subscriber which in turn stops the ring tone. Then the two parties connect, and they may engage in conversation. Either subscriber can terminate the call by pressing the hook switch.

Still, the sequence diagram is sketchy and contains many "narrating" gaps that appear as discontinuity in the succession of events. Take, for example, RING and its consequences: OFF-HOOK. There are many threads that are missing here. Ringing involves the following sequence:

- The network sends a ringing signal
- The ringing signal leads to the following alternatives:
  - If the hook is Off, busy signals are sent to the network to arrive at the calling subscriber and activate a busy sound
  - If the hook is On, a ringing sound is activated that stops when the hook is lifted, which in turn triggers conversation

By contrast, the FM representation is characterized by continuity of different threads, making it possible to have a tight series of superimposed protocol rules.

Figure 11 shows an FM-based model of this subscriber signaling scenario. It starts at the top left corner when the hook is lifted off (circle 1). This triggers the creation of an off signal (circle 2) that flows to the network, The signal is processed (circle 3) and triggers the creation of a dial signal (circle 4). The dial signal flows to the calling subscriber, where it is processed (circle 5) to trigger the dialing sound (circle 6).

At this point, we expect the user to dial numbers (circle 7) that are transferred to the network (circle 9), where the called telephone number is processed to trigger (circle 10) ringing that is transferred to the called subscriber, where it is processed (circle 12). Then, according to the state of the hook, we expect the following:

- The hook is off (circle 13), hence a busy signal is created and sent to the network, then to the calling subscriber (circle 15) to trigger a busy sound (circle 16).

Journal of Theoretical and Applied Information Technology
20th May 2013. Vol. 51 No.2
© 2005 - 2013 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

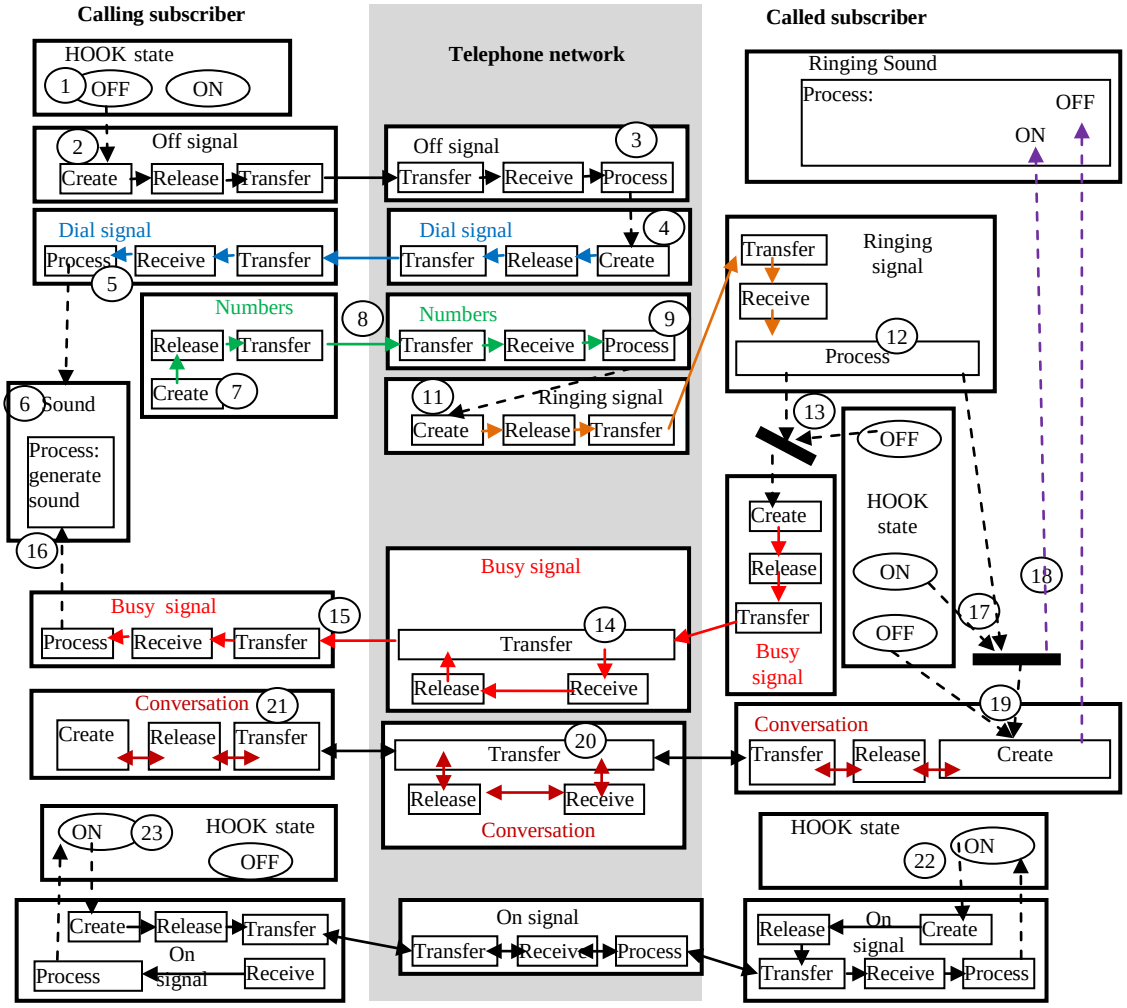


Figure 11. FM Representation Of The Subscriber Signaling Scenario

- The hook is on (circle 17), when a ringing sound is activated (circle 18), and simultaneously, a conversation is created when the hook is lifted (circle 19). Notice that for simplicity's sake we have repeated drawing OFF in the Hook state. Also, we have removed the name of the stage since it is understood from previous discussion. Additionally, we represent the sphere and the flowsystem by one box when there is only one flowsystem in the sphere.

The conversation flowsystems (circles 19, 20, and 21) in the three spheres of calling subscriber, called subscriber, and network are bi-directional in their flows. The conversation continues until one of the two subscribers hangs up (ON) (circles 22 and 23). In this case, an On signal triggers Hook ON on the other side. Here we can modify this such that the ON signal triggers the generation of a sound on the other side.

It may be mentioned in this context that such FM representation is characterized by complexity. Nevertheless, it is a complexity of completeness of details that is specified in a systematic way. It is analogous to the complexity of electrical schematic circuits.

6. CONCLUSION

This paper proposes a diagrammatic methodology for protocol specification. Currently, several notations are utilized in the specification of protocols, including different types of diagrams such as flowchart-like depictions, UML sequence diagrams, and state transition diagrams. The new flowthing model presents a conceptual and

© 2005 - 2013 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

complete description of basic streams of flow among entities and stages including "crossing points" that need rules of data transfer. The resultant specification is a map over which a protocol can be superimposed. Several examples have shown the viability of the model as a tool that can be supplemented by other notations of logical connection and synchronization. Further research could experiment with applying the proposed representation to specific protocols.

REFERENCES:

[1] Pallapa, V. (2012). Introduction to Basics of Communication Protocol, Department of Electrical Communication Engineering, Indian Institute of Science, 2012 (access). http://pet.ece.iisc.ernet.in/course/E2223/Cha1.pdf

[2] Stallings, W. (1994). *Data and Computer Communications*, 4th edition, MacMillan, 1994.

[3] Pärssinen, J., Turunen, M., Heinonen, J., von Knorring, N., Kvist A., and Jäppinen, P. (1999). *Protocol Engineering Concepts and Patterns using UML*, 25 November 1999. http://edu.pegax.com/lib/exe/fetch.php?media=csa:pecp-uml.pdf

[4] Tanenbaum, A. S. (1989). *Computer Networks*, 2nd edition, Prentice-Hall International, 1989.

[5] ITU-T, Information Technology - Open Systems Interconnection - Basic Reference Model: The Basic Model, Recommendation X.200, ITU, 1994.

[6] Dostálek L., and Kabelová, A. (2007). Understanding TCP/IP: A Clear and Comprehensive Guide to TCP/IP Protocols, Packt Publishing, 2007.

[7] Hekmat, S. (2005). *Communication Networks*, PragSoft Corporation, 2005. http://78.159.163.139:9205/1496EDD13DBE130BE31FD1336A62C4E5DF51A2FD773580F06CA13A8327E8D022D2F222DE0C158FAA3C32F4A40/www.pragsoft.com/books/CommNetwork.pdf

[8] Parziale, L., Britt, D. T., Davis, C., Forrester, J., Liu, W., Matthews C., and Rosselot, N. (2006). *TCP/IP Tutorial and Technical Overview*, Eighth Edition, IBM, December 2006. http://www.redbooks.ibm.com/redbooks/pdfs/gg243376.pdf

[9] Simoneau, P. (2011). The TCP/IP and OSI Models, Global Knowledge, White Papers, 25 February 2011. http://www.globalknowledge.com/training/whitepaperdetail.asp?pageid=502&wpid=825&country=United+States

[10] Kazierok, C. M. (2004). The TCP/IP Guide - Version 2.0, 2004. http://www.tcpipguide.com/TCPIPGuide_2-0_s2.pdf

[11] Kazierok, C. M. (2005). The TCP/IP Guide Version 3.0, 20 September 2005. http://www.tcpipguide.com/free/t_NetworkLayerLayer3.htm

[12] The Computer Technology Documentation Project, Network Layers. http://www.comptechdoc.org/independent/networking/protocol/protlayers.html

[13] OmniSecu.com, TCP/IP Encapsulation and Decapsulation. http://www.omnisecu.com/tcpip/tcpip-encapsulation-decapsulation.htm

[14] Wetteroth, D. (2011). *OSI Reference Model for Telecommunications*, McGraw-Hill, 2011.

[15] Networking in IPV6, IPV4, Switching, Routing, IPV6Araea.com, 2011. http://www.ipv6area.com/2011/12/what-is-tcpip-layers-and-architecture.html

[16] Kaliappan P. S., and Koenig, H. (2011). "An Approach to Synchronize UML-Based Design Components for Model-Driven Protocol Development", 2011 *IEEE 34th Software Engineering Workshop*, Limerick, Ireland, 20-21 June, pp. 27-35. ISBN: 978-0-7695-4627-8

[17] Smith,S., Beaulieu A. and Phillips, W. G. (2011). "Modeling and Verifying Security Protocols using UML 2", *IEEE International Systems Conference* (SysCon), 4-7 April 2011, pp. 72-79.

[18] Teraoka, F., Gogo, K., Mitsuya, K., Shibui R., and Mitani, K. (2007). Unified L2 Abstractions for L3-Driven Fast Handover, IRTF MobOpts RG, Internet-Draft, 2007, http://tools.ietf.org/id/draft-irtf-mobopts-l2-abstractions-02.txt

[19] IEEE, "802.11-2007 IEEE Standard for LAN/MAN – Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", 2007.

[20] IEEE, "802.3, 2000 EDITION ISO/IEC 8802-3:2000 (E) Information Technology - LAN/MAN - Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications", 2000.

© 2005 - 2013 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[21] Teraoka, F., Gogo, K., Mitsuya, K., Shibui R., and Mitani, K. (2008). Unified L2 Abstractions for L3-Driven Fast Handover, IRTF MobOpts RG, Internet-Draft, 2008. http://www.citeulike.org/user/shtrom/article/9500320

[22] Teraoka, F., Gogo, K., Mitsuya, K., Shibui R., and Mitani, K. (2007). Unified L2 Abstractions for L3-Driven Fast Handover, IRTF MobOpts RG, Internet-Draft, 2007. http://www.ietf.org/proceedings/67/slides/MobOpts-6.pdf

[23] Al-Fedaghi, S. (2008). "Scrutinizing the Rule: Privacy Realization in HIPAA", *Int. J. Healthcare Inf. Syst. Informatics* (IJHISI), Vol. 3, No. 2, 2008.

[24] Al-Fedaghi, S. (2008). "Software Engineering Interpretation of Information Processing Regulations," *IEEE 32nd Annual International Computer Software and Applications Conference* (IEEE COMPSAC 2008), Turku, Finland, 28 July–1 August 2008.

[25] Al-Fedaghi, S. (2009). "Flow Based Description of Conceptual and Design Levels," Proceedings of the *IEEE International Conference on Computer Engineering and Technology*, Singapore, 22–24 January, Vol. 1, pp. 16-20, 2009.

[26] Al-Fedaghi, S. (2008). "Systems of Things that Flow," Proceedings of the *52nd Annual Meeting of the International Society for Systems Sciences* (ISSS 2008), University of Wisconsin, Madison, USA, 13–18 July 2008.

[27] Al-Fedaghi, S. (2012). "Conceptual Framework For Recursion In Computer Programming", *Journal of Theoretical and Applied Information Technology*, Vol. 46 No. 2, 2012.