Modeling Theory of Component Based Network Architecture

Feng YanLi

1. Dept. Computer, Shandong Institute of Business and Technology, fengyanli@sdibt.edu.cn
doi:10.4156/jdcta.vol5.issue4.14

Abstract

In this paper, we first provide formal (nominal) definition of component, connector and network architecture, and then use process algebra theory to define component operations on the network architecture. Further, we develop the network architecture’s algebra model and prove the model’s consistency and completeness.

Keywords: Component, Network Architecture, Process Algebra

1. Introduction

Computer network is a distributed and concurrent complex system. Logically, its architecture must be specified, verified, designed, implemented and evaluated formally and correctly. As the logic framework of the network system, network architecture specifies and describes the network. Network system description can be classified into two levels: system architecture level and protocol level. At the system architecture level, the concepts and principles for network system design and construction are described, the logical relations between system components, functional constraints and modulars are specified, and the system behavioral structures are defined. At the protocol level, system component operations and data structure are designed and described, and thus the protocol description refines the functions and services of the network architecture. Network architecture is the highest level conceptual description of a computer network system, and together with the design description at the protocol level it depicts the various levels of the network system. For this reason, the formal network architecture consists mainly of the specification and verification, and it is about the high level configuration of a network system not about the implementation details of the system itself. As the design, implementation and evaluation of a network system are mostly related to the network protocols, they belong to protocol engineering. Through an in-depth analysis of a typical network architecture, this paper characterizes the basics of various network architectures, formally defines the components and connectors of a network architecture and the architecture itself. The paper also conceptualizes component operations and develops an algebra system for the network architecture. Further, by incorporating process algebra into the network architecture modeling, we create an algebra model for the network architecture[1].

2. Network Architecture framework

In the history of computer network development, network architecture has played an important role. As the logical framework of a computer network, network architecture is directly related to the functions and capacity of the network. Network architecture design covers the overall configuration, control and definitions of the network, and specifies the concrete function of each system component and the communication protocols between components.

In the beginning stage of computer network development, complete and special network architectures are usually provided independently by computer systems manufacturers. Examples include SNA[2] by IBM and DNA[3] by DEC. Driven by product development, these kinds of network architectures provides concrete definition of network system components, the available functions and interface regulations for the components, and the communication protocols between the components. And these components form up a network architecture. In the second stage of network development, i.e., open network integration stage, open system integration reference model was developed by ISO so that networks manufactured by different companies and of different network configurations can be integrated. Designed for the integration of network systems with different configurations, these kinds
of network architectures have specified the component entities of a standard open network system, and the services each entity can provide and can be made available to other entities, and the communication protocols between those entities. So the network architecture is a model of system integration. In the third stage of network development, i.e., network application stage, research on network architecture has come to a stage of multiple orientations, since the onset of the network architecture of Clark and Tennenhouse[4]. The most representative network architectures of this stage are the Active Network Architecture (ANA)[5] and TINA[6]. In an ANA that specifies the execution sequence of every system component and their communication protocols, the combined services are provided by the resident components which resident at active network nodes and through the embedded components that are actively embedded onto network nodes. Using object oriented method, TINA proposes that distributed telecommunication application be composed of a group of computing objects and these computing objects interact through interfaces. In addition, in the computing modeling of TINA, definitions for invoking oriented operation interface and for information stream oriented stream interface are provided.

It can be seen from the analysis of different network architectures that components, entities and objects are all the basic elements of a network system. Thus they can all be called components. They communicate with each other through various logical connections (such as layer interface and connection protocols). We will conceptualize the environment for the interactions between multiple components as a component or a connector. In the following we will formally describe the components based network architecture, and develop a formal visual chart for the network architecture that can be expanded and whose system is composed of subsystems that are formed up with components and connectors. We also consistently define and describe the characteristics of components, connectors and network architectures and their dynamic behavior. We further classify the various different ways of connections into different categories, discuss the interactive relations between various network architectures, and put forward a general network architecture model that will provide a solid theoretical foundation for further research in network architecture.

Zhao[7] proposes a mathematical model theory for software architecture that is built upon set theory and process algebra. We will formally describe and analyze the network architecture model by using the framework composed of components and connectors.

2.1. Formal Definition of Component

As one of the basic constructing elements for network architecture, a network component is generally believed to possess certain functions and is a recognizable network unit. And components interact with each other through interfaces. In network architecture, components can be of various sizes, and a number of components can be combined into another component which is more complex and has more powerful functions. Components are independent from each other so that they can be separately used as independent units in different architectures. To their outside environment, components conceal their concrete implementation details and provide services only through interfaces. Components are the logical units of a network system and not related to any specific implementation methods.

As a logical unit of a network system, a component interacts with its outside environment through its interface only. A component interface consists of a group of terminal interfaces where each terminal interface represents an interactive port between a component and its outer environment. It thus can be followed that a component description is mainly for that of the component’s port.

According to their characteristics, components in network architecture can be classified into two types: network components and user components. Network components provide network services and network protocol behavior. At the same time, all the network components include channel processes for interaction with their surroundings. User component is a process set for communications between users, and it can transfer the service functions provided by the network components. But user component is not used for transmitting messages. Given that network component has multiple properties, we will summarize a few of those important properties and provide formal definition of two kinds of components.

Definition 2.1.1: Network Component
NC=(Serv, Stat, Chan, Beha, Cons) is a quintuple,
where Serv is the service process set for the network component, Stat the statement process set, Chan the channel process set, Beha the description of network component behavior, and Cons is the set that defines the constraints for network components and the relations between composing units.

Definition 2.1.2 User Component
UC = (Serv, Stat, User, Beha, Cons) is also a quintuple,
Where Serv the communicating process set for the service syntax between user component and network component, Stat is the user service control process set for transferring services, User is the visiting control process set for the user component’s behavior, Cons is the set that defines the constraints for user components and the relations between composing units.

In the following, we will use component name and element name to describe the elements in the component quintuple. For example, UC.Serv represents the Serv element in the component UC.

Definition 2.1.3: Given two components C1 and C2, if C1 and C2 satisfy the following conditions:

1. C1.Serv = C2.Serv
5. C1.Cons = C2.Cons

Then components C1 and C2 are equal. And the equality can be written as C1 = C2.

2.2. Formal Definition of Connector

A connector is used to establish the interactions between components and is a composing unit of the network architecture that dictates the interaction principles. The concept of connector breaks away from the traditional held view that network architecture can only use direct interaction model to describe the relations between individual entities. This conceptual breakthrough has helped not only to describe and analyze the characteristics for the interaction settings between entities, but also describe the interactions among multiple entities, and at the same time ensure that we can use mathematical model to formally describe network architecture.

Connector is a special type of component, and components interact with each other through connectors. The interfaces of connectors are composed of all the interaction points that connect the connector and its components. These interaction points are called roles. Each role of a connector indicates the interaction participants for this connector. If a connector has multiple roles, then this connector can be connected with multiple components. In another word, this connector can describe the interactive relations among multiple components. To describe a connector is mostly to describe the roles of the connector [8].

According to their functions, connectors in network architecture can be classified into two categories. One category is called interaction connectors, which are used to describe the service interface models and represent the interactions between service providing components and user components. This kind of connector usually represents the interaction settings of “request/service”(service interface model)” or “request-service/request-service (internet interconnection model).” Another category is the transfer connector, which is used to describe various network system interaction models and usually represent the interaction settings for “receive-filter-send.” Network connector also has multiple properties, we will summarize a few of those important properties and provide formal definition of the two kinds of connectors.

Definition 2.2.1 Interaction Connector
IC = (Role, Brok, Beha, Cons) is a quadruple
Where Role is the interaction set between the interaction connector and its connecting components, Brok is the acting process set for processing and transferring request and service messages for the connector, Beha is the syntax description for the interaction connector’s behavior, and Cons is the set describing the relations between the interaction connector’s constraints and its composing elements.
Definition 2.2.2: Transfer Connector

$TC = \langle \text{Role, Filt, Data, Cons} \rangle$ is a quadruple

Where $\text{Role}$ is the interaction set between the transfer connector and its connecting components and is mainly used as the interface for receiving and sending messages, $\text{Filt}$ is the process set of the transfer connector for processing and transferring input messages and output results, $\text{Beha}$ is the syntax description for the transfer connector’s behavior, and $\text{Cons}$ is the set describing the relations between the transfer connector’s constraints and its composing elements.

2.3. Components based the Network Architecture Framework

As the logical framework of a network system, network architecture can be used to guide the network system analysis and design, including the description of the composing elements (components and connectors) of the network system, definition of the services provided by those elements, and the interactive relations and the interaction protocols between those elements. Here is the definition of network architecture.

Definition 2.3.1: Network Architecture Model

$NA := (CP, CC, CS)$ is a triple

Where $CP$ represents the component set of the network architecture and is the basic algorithm logic unit and data logic unit of the network architecture, $CC$ represents the connectors set of the architecture and is the connecting unit among components, $CS$ represents the constraints of the network architecture and thus describes the relations between components and connectors and shows how the components and connectors are combined together.

According to definition 2.3.1 and the definitions for components and connectors given previously, the basic concept of a network architecture model can be described as follows:

Network architecture model :: = (Components, Connectors, Constraints)

Component :: = \{terminal1, terminal2, …, terminal n\}

Connector :: = \{role1, role2, …, role m\}

Constraint :: = \{(terminal i, role j) …\}

Definition 2.3.2 Given two respective network architectures $NA := (CP, CC, CS)$ and $NA’ := (CP’, CC’, CS’)$, if they satisfy the following:

$CP \supseteq CP’$

$CC \supseteq CC’$

$CS \Rightarrow CS’$

Then $NA’$ is the sub-architecture of $NA$.

Network architecture has the following main properties:

1) Interactive Operations: The components of the network architecture can via connectors carry out direct and indirect operations toward other components.

2) Expandable: Network architecture can through connectors dynamically change the set of the connected components, and the interaction relations among those components can be modified accordingly.

3) Combinable: Network architectures can be combined into sub-architectures through connecting their components, and those sub-architectures can be combined into more complex architectures.

Components and connectors are the most important composing parts of network architecture, the internal relations between components and connectors directly demonstrate the behavior model of the network system. In the following, we will mainly discuss the algebra model of network architecture where components and connectors (as components operations) are the object. We denote $NA = (CP, CC)$.

3. The Algebra Model of Network Architecture

Process algebra has defined many combination operations for process, and those operations demonstrate interaction methods and principles of process. Complex architecture can be constructed through combination of processes. Process combination is the most distinguishing feature that differentiates process algebra from other formal procedures (methods). In the context of process
algebra, components of network architecture can all be represented as processes, system behavior can be described by the processes that compose the components, and the system itself can be represented by a process. Connectors help to establish the interactions between components and the interaction principles, and thus help to realize the interactive operations between components. In fact, connectors can be viewed as the implementation of components operations and thus help to set up a network architecture algebra model.

3.1. Component Operations

Definition 3.1.1: Component Concurrent Operations

Given a network architecture NA and its components set CP, C1, C2 ∈ CP, then the concurrent operations of components C1 and C2 can be written as C1 ‖ C2, C1 ‖ C2 becomes a new component, and C1 and C2 carry out their concurrent operations independently.

Definition 3.1.2: Component Synchronous Operations

Given a network architecture NA and its components set CP, C1, C2 ∈ CP, then the synchronous operations of components C1 and C2 can be written as C1 ⊕ C2, C1 ⊕ C2 becomes a net components, C1 and C2 must be executed synchronously or cooperatively.

Definition 3.1.3: Component Selective Operations

Given a network architecture NA and its components set CP, C1, C2 ∈ CP, then the selective operations of components C1 and C2 can be written as C1 □ C2, C1 □ C2 becomes a new component, execution of C1 and C2 will be carried out according to their exterior environment.

Component selective operations can be used to represent the system behavior of the traditional design model of branching computing procedure.

Theorem 3.1.1: Given a network architecture NA and its components set CP, C1, C2, C3 ∈ CP, then the components connecting operations ‖, □, ⊕ are of reflective, commutative, associative properties. For example, for concurrent operation ‖, the following equalities hold.

L1 : C1 ‖ C1 = C1 (reflective)
L2 : C1 ‖ C2 = C2 ‖ C1 (commutative)
L3 : (C1 ‖ C2) ‖ C3 = C1 ‖ (C2 ‖ C3) = C1 ‖ C2 ‖ C3 (associative)

Corollary 3.1.1: The equalities in theorem 3.1.1 can be extended to the case of finite number of components, that is, C1 ‖ C2 ‖ C3 ‖ … ‖ Cn.

Theorem 3.1.2: Given a network architecture NA and its components set CP, C1, C2, C3 ∈ CP, then the components connecting operation ‖ or ⊕ is of distributive property with respect to operation □. So the following equality holds.

L4 : C1 ‖ (C2 □ C3) = (C1 ‖ C2) □ (C1 ‖ C3)
( C1 □ C2 ) ‖ C3 = ( C1 ‖ C3 ) □ ( C2 ‖ C3 )
C1 ⊕ (C2 □ C3) = (C1 ⊕ C2) □ (C1 ⊕ C3) (distributive)
( C1 □ C2 ) ⊕ C3 = (C1 ⊕ C3) □ (C2 ⊕ C3)

Definition 3.1.4: Component Sequence Operations

Given a network architecture NA and its components set CP, C1, C2 ∈ CP, then the sequence operation for components C1 and C2 can be written as C1;C2, C1;C2 becomes a new component. C1;C2 means that first execute C1; and if succeed, execute C2.

Component sequence operations are used to represent the system behavior of the traditional design model of sequencing computing procedure.
Definition 3.1.5: Component interruption Operation

Given a network architecture NA and its components set CP, C1, C2 ∈ CP, then the interruption operation for components C1 and C2 can be written as C1 ∆ C2, C1 ∆ C2 becomes a new component. C1 ∆ C2 means that C1 is executed first, and start to execute C2 before C1 is successfully completed. Component interruption operations are used to represent the system trouble shooting and recovery behavior.

Theorem 3.1.3: Given a network architecture NA and its components set CP, C1, C2, C3 ∈ CP, then the components sequence operation ; and component termination operation ∆ are of reflective and associative properties. For example, for sequence operation ;, the following equalities hold.

L1 : C1 ; C1 = C1        (reflective)
L2 : ( C1 ; C2 ) ; C3 = C1 ; ( C2 ; C3 ) = C1 ; C2 ; C3   (associative)

Corollary 3.1.2: The equalities in theorem 3.1.3 can be extended to the case of finite number of components, that is, C1 ; C2 ; C3 ; ... ; Cn.

3.2. Network Architecture Algebra Model

Network architecture is mainly composed of components, while connectors are the implementation of component operations. Accordingly, we present the following definition of network architecture model.

Definition 3.2.1: Given a network architecture NA and its components set CP of C1, C2, ..., Cn, and NA = (CP, CC), where CP represents the components set, and CC the operations set, i.e., CC = { ‖ , □ , Q , ; , ∆ } ∈ CP. Applying a finite number of components operations to Ci (i=1, .., n), we will have a new network architecture algebra model. Let the new architecture be noted as NAA model.

Theorem 3.2.1: Given a network architecture NA=(CP, CC), where CP represents the components set, and CC components operation set, i.e., CC= { ‖ , □ , Q , ; , ∆ }, then the following holds.

1 ) ( CP , ‖ ) , ( CP , □ ) , ( CP , Q ) constitute commutative respective semi-groups
2 ) ( CP , ; ) , ( CP , ∆ ) constitute semi-groups.

Proof:
We’ll only provide proof for the operation of ‖ , because the conclusions for other operations can be proved similarly. Suppose ∀ C1 , C2 ∈ CP, then we have

a ) According to definition 3.1.1, we know that CP is closed to operation ‖ , that is, C1 ‖ C2 ∈ CP ;

b ) According to theorem 3.1.1, we know that CP is of associative property with respect to operation ‖ , that is, ( C1 ‖ C2 ) ‖ C3 = C1 ‖ ( C2 ‖ C3 ) ;

c ) According to theorem 3.1.1, we know that CP is of commutative property with respect to operation ‖ , that is, C1 ‖ C2 = C2 ‖ C1 ;

Thus, NA=(CP, ‖ ) is a commutative semi-group. QED.

Definition 3.2.2: Suppose NA=(CP, CC) is a network architecture, where CP represents components set, CC the operations set, that is, CC= { ‖ , □ , Q , ; , ∆ }. If the non-empty set CP’ of CP is closed to all operations, then (CP’, CC) can be called the semi-group of (CP, CC).
The Consistency and Completeness of Network Architecture Algebra

The network architecture algebra model given in the above is mainly composed of formal components, formal connectors and the component operations. So it is only a formal definition of network architecture algebra. The definition generalizes the network architecture definition to a general algebra structure by using traditional abstract algebra theory (In fact, it is a semi-group. If we define additional special components, we can prove that it is a semi-group with unitary element, i.e., an odd point.) Consequently, we can use the abstract algebra theory and methods to study the corresponding network architecture on the one hand, and on the other hand can ensure the abstractness, compactness and correctness of the constructed network architecture.

The component operations or executions defined on components set of the network architecture reflect the basic interaction methods between components, and network architecture model can be represented via components and by a series of components operations. Now the questions come to these two. First, are the components operations in the network architecture algebra model consistent? That is, are the components expressions meaningful? Second. Are the components operations in the network architecture algebra model complete? That is, can the previous defined components operations be able to describe all the interactions of components in the network architecture? The following results ensure in theory that the network architecture algebra model is consistent and complete.

Theorem 3.3.1: The network architecture algebra model NA=(CP, CC) is consistent and complete.

Proof: We will use the component operation ‖ as an example to prove the consistency of NA.

Suppose C1, C2 ∈ CP are two network components, from definition 2.1.2, C1 and C2 can be written as

\[ C1 = (\text{Serv1}, \text{Stat1}, \text{Chan1}, \text{Beha1}, \text{Cons1}) \]
\[ C2 = (\text{Serv2}, \text{Stat2}, \text{Chan2}, \text{Beha2}, \text{Cons2}) \]

From the definition for operation ‖ it can be followed that C=C1 ‖ C2 satisfies the following properties:

(1) C . Serv=C1 . Serv1 ∪ C2 . Serv2

From the above properties, it is known that a new component C is constructed from applying component operation ‖ to C1 and C2, and the composing elements for component C can all be drawn from the composing elements of C1 and C2. So C is still a component of the nominal network architecture. Therefore, the simultaneous operation ‖ is defined for any component of the components set. That is, consistency is achieved.

By the same reasoning, all components operations of the NA model satisfy the consistency property. Thus the components expressions constructed from components operations or the network architecture algebra model satisfy the consistency property.

Now we prove the completeness of NA.

According to the results from the communications computing system (CCS) in [8], we know that the basic behavior character of a network system is concurrent, and the concurrent operation ‖ has the capacity of expressing concurrent operations. Also, any methods that are good for formal network architecture requires not only adopting Turing model based sequence computing model, but also concurrent computing model. Because the concurrent operation model can not be derived from improving or expanding the sequence operation model, so the concurrent operation ‖ is necessary.

With respect to sequence operation framework model, the results in [9-12] tells us that any program structure can be represented by sequence, branching and rolling structures. It is obvious from definition 3.1.4 that the component sequence operation ‖ can be able to represent sequence structure, while
definition 3.1.3 tells that component selective operation \( □ \) is of the ability of branching and the rolling structure can be expressed through carrying out component sequence operations for \( n \) times. So, the components operations in the NA model can sufficiently describe the sequence operation structure. The other component operations can also be derived from the above operations.

So it can be followed that the network architecture algebra model NA is complete.

4. Conclusions

In this chapter, by the process algebra as a theory foundation we use formal methods to describe the components of network architectures and the concepts of connectors, and to establish the algebra models of network architectures. This has laid a foundation for the further research on network architectures.

5. Acknowledgment

This work is partially supported by National Natural Science Foundation(61070175). The author also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

6. References