Network Infrastructure Technology Supporting Parallelization and Multiplexing of Services

● Yuzuru Iida ● Hiroaki Tanaka

The increasingly fierce competition between information and communications technology (ICT) companies is increasing the need for them to carefully cultivate markets, by first starting small and then later expanding processing power so as not to affect existing services. Parallelizing services is an important technology in this process while multiplexing is an essential technology in building highly reliable systems. Furthermore, services are being added and modified with increasing frequency in order to handle diversifying customer needs in a timely manner, and the number of service users and smart devices, such as smartphones, is increasing rapidly. For these reasons, packet data traffic on IP networks is growing and diversifying more and more quickly. This paper describes network infrastructure technologies for creating scalable infrastructures that can appropriately distribute diverse and high-volume packet data to parallelized and multiplexed server systems providing highly reliable and high-volume services and that enable services to be added and modified frequently. It also discusses initiatives at Fujitsu in this area.

1. Introduction

High-performance processing is increasingly needed by application services to handle the continuing increase in the amounts of data being handled. Furthermore, the growing importance of social infrastructure means that it needs to be highly reliable. Conventionally, it has been possible to increase performance and reliability in a single server, but with services diversifying ever more quickly, the increases in packet data traffic of these services are exceeding the rate at which server performance can be increased. Under such conditions, server parallelization and multiplexing are effective technologies for implementing systems with both high performance and high reliability.

Generally, for systems providing services, servers are multiplexed and run in parallel to maintain quality of service while avoiding over-investing in equipment. In such cases, packet data must be processed to enable it to be distributed to a server (distribution processing). In doing so, service performance and reliability can be improved by modifying the distribution destinations in accordance with the load state of each server and the state of service quality. In the past, this has been done by providing the necessary distribution logic individually, when a service is first provided. However, as services are being added and changed more often, establishing a mechanism that will enable the distribution logic to be changed without interrupting services has become important.

In this paper, we describe technologies for building a scalable network infrastructure for operating highly reliable, high-volume services on parallel, multiplexed server systems. This infrastructure enables effective distribution of packet data, which is ever increasing in diversity and volume, and enables services to be added and modified reliably without interrupting services. We also describe Fujitsu's initiatives in this area.

2. Programmable network infrastructure component technologies

The processing flow for the network infrastructure is shown in **Figure 1**. Four main technologies are used to distribute the packet data to the servers.

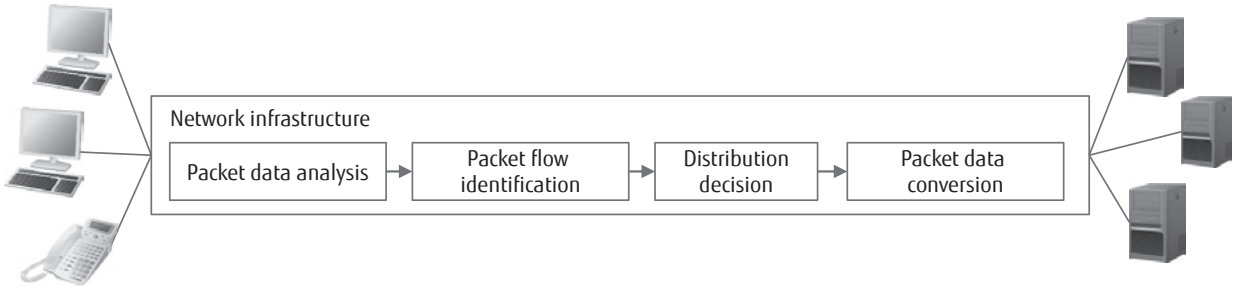

Figure 1
Network infrastructure processing flow.

1) Packet data analysis

Contents of received packets are analyzed, and protocol type to be used for packet data is identified.

2) Packet flow identification

Flows of packet data related to various services are identified on basis of identified protocol type.

3) Distribution decision

Distribution of packet data to servers is decided in accordance with service quality parameters such as workload and response time of each server.

4) Packet data conversion

Contents of received packets are converted so that they can be transferred to target server.

Programmable network infrastructure component technologies are needed for distributing packets to servers, and these components should be implemented as functions. Fujitsu is making the packet distribution process programmable so that the distribution logic can be changed without interrupting services.

Our objective is to develop a network infrastructure in which the packet distribution process is programmable so that the network is not susceptible to service interruptions due to such actions as software rebuilding and equipment restarting and so that the packets can be distributed in accordance with the server load and service quality.

To implement a programmable network infrastructure, we used a structure that divides each function into definition files, which regulate details of operation, and a processing engine, which operates in accordance with these definition files. This structure enables services to be added or modified by simply changing the definition files. Existing services are not affected and continue running, resulting in stable operation.

The four component functions of the programmable network infrastructure are explained below.

2.1 Programmable packet data analysis function

The programmable packet data analysis function uses a "protocol format definition file" to identify the protocol type to be used for the received packet data. This file includes the protocol format information needed to analyze the packet data such as the field starting positions. When a new protocol is defined, protocol information is added to this file to enable the new protocol to be used.

An overview of this function is shown in **Figure 2**.

2.2 Programmable packet flow identification function

The programmable packet flow identification function uses packet flow identification definition files and specifies a flow identifier for the protocol type specified by the packet data analysis function. This file describes identifiers for flows using a protocol. When a new protocol is defined, flow identifier information for the new protocol is added to this file.

An overview of this function is shown in **Figure 3**.

2.3 Programmable distribution decision function

The programmable distribution decision function uses distribution logic definition files and decides where to distribute packets as specified by the packet flow identification function. It also maintains sessions by managing the correspondence between flow identifiers and distributions that have been decided while the flow or session is valid (i.e., the packet data continues

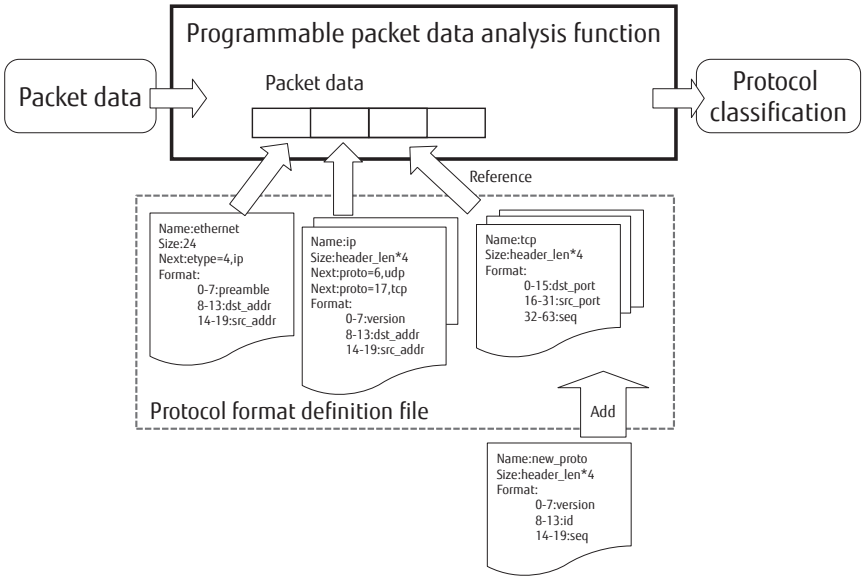

Figure 2
Programmable packet data analysis function overview.

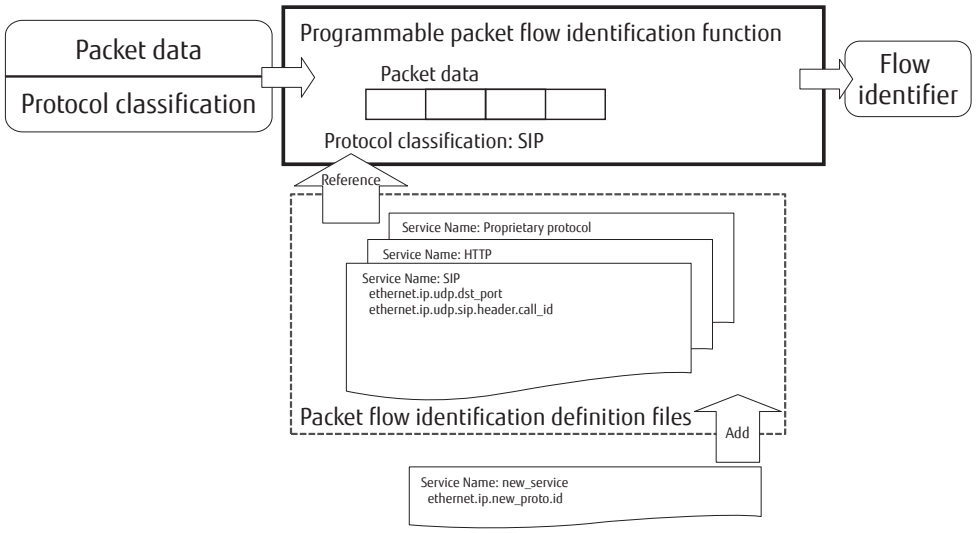

Figure 3
Programmable packet flow identification function overview.

to be distributed). The programmable distribution decision function first checks this managed data, and, if a received flow identifier is already being managed (i.e., the received flow identifier has already been registered in the database), it uses the distribution already under management.

The distribution logic definition files describe distribution logic such as how to distribute the load for each flow identifier in accordance with the server loads. Optimal packet distribution can be achieved after adding servers for a new service by changing the logic in the distribution logic definition files. If the available servers become congested, the number of servers used for the service can be increased by, for example, changing the distribution logic definition files from using an N+M redundancy control multiplexing scheme to using an N+1 scheme.

An overview of this function is shown in **Figure 4**.

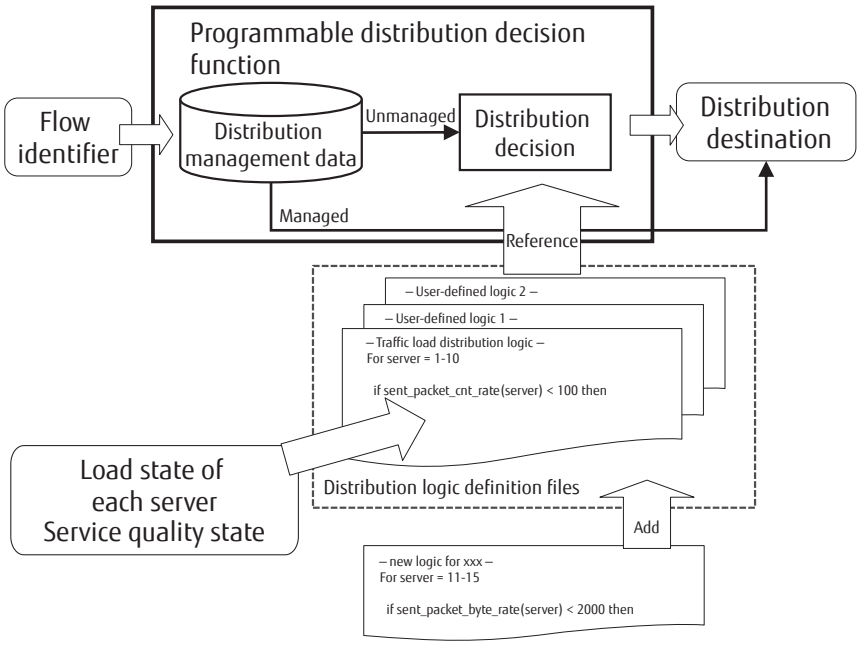

Figure 4
Programmable distribution decision function overview.

2.4 Programmable packet data conversion function

The programmable packet data conversion function uses packet data conversion logic definition files and performs the packet data conversion necessary to send the packets to the target server.

The packet data conversion logic definition files describe the rewriting logic (i.e., which fields to rewrite and how) for each protocol type needed to convert packet data to be sent to the target server and the logic for other modifications necessary due to the conversion of the packet data. When a new protocol is defined, packet data rewriting logic is added to this file.

An overview of this function is shown in **Figure 5**.

3. Platform

We implemented the four functions described above for the programmable network infrastructure and used the FlowEngine data plane software as a platform for this implementation. FlowEngine runs on Linux and was developed to support a variety of customer needs related to traffic control. It supports flexible business development in stages, taking services from a small start-up to full-fledged operation. It also makes it easy to build in the particular processing required for a variety of services.

FlowEngine also supports flexible hardware updates, enabling developed software assets to be utilized effectively. Another reason for using FlowEngine is that allocation of network functions across CPU cores can be optimized in accordance with performance requirements.

FlowEngine comprises three components. The internal structure of FlowEngine is shown in **Figure 6**, and an overview of the processing is shown in **Figure 7**.

1) FlowEngine framework

This component has functions for controlling execution of data plane processing, such as running the engine thread and pipelining add-ons, as well as I/O functions.

2) Included add-ons

These are extension functions provided by FlowEngine. They include standard add-ons, which provide standard network processing, and service add-ons, which execute service functions.

3) Add-on API

This is the application programming interface (API) provided for installing user add-ons. With FlowEngine, customer requirements are implemented by installing user add-ons that match particular

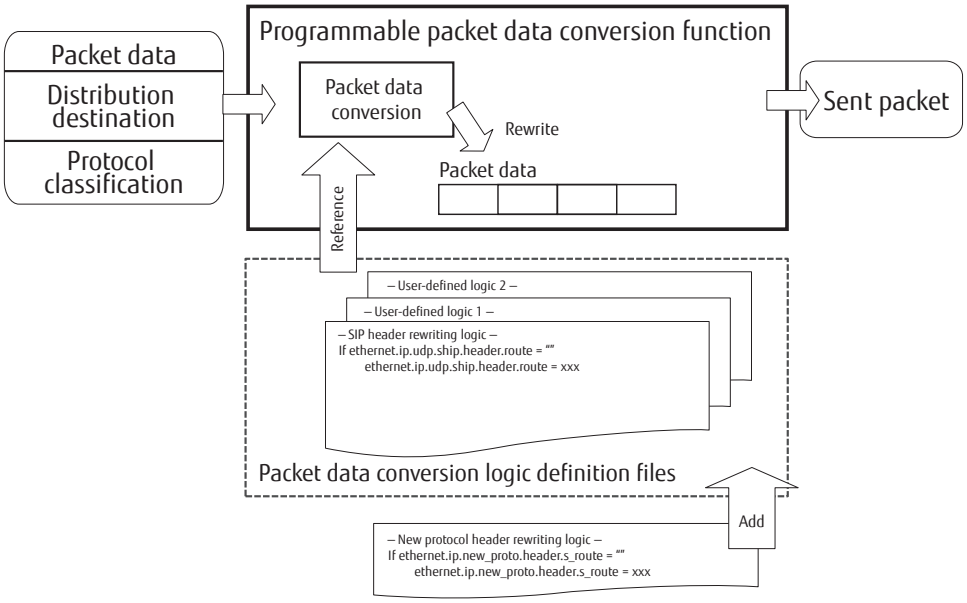

Figure 5
Programmable packet data conversion function overview.

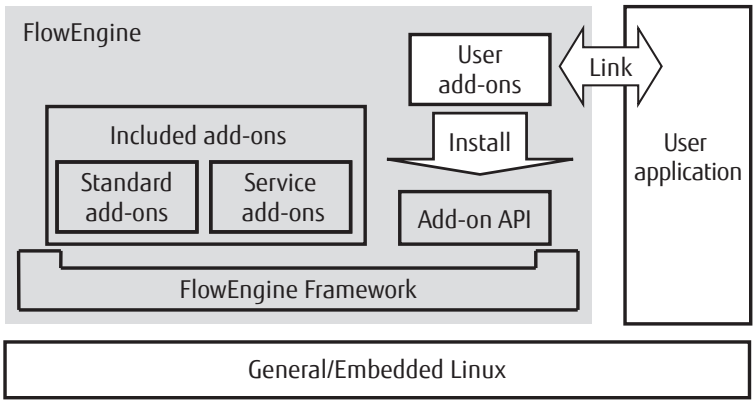

Figure 6
FlowEngine internal structure.

requirements to FlowEngine and by linking their operation to user applications.

4. Scripting engines

The programmable network infrastructure functions consist of definition files that prescribe the operations, and scripting engines (processing engines) that operate in accordance with the definition files. Use of the scripting language to specify a definition file enables the creation of functions that support new services and new protocols through simple editing of the appropriate definition file, without the need to re-build the entire program. The scripting engines are built into FlowEngine using the add-on functionality, creating a programmable network infrastructure.

An image of the programmable network infrastructure with embedded scripting engines is shown in **Figure 8**. The packet data analysis, packet flow identification, and packet data conversion functions execute using the scripting engine in the corresponding FlowEngine add-on. The distribution decision function forwards distribution requests from the distribution

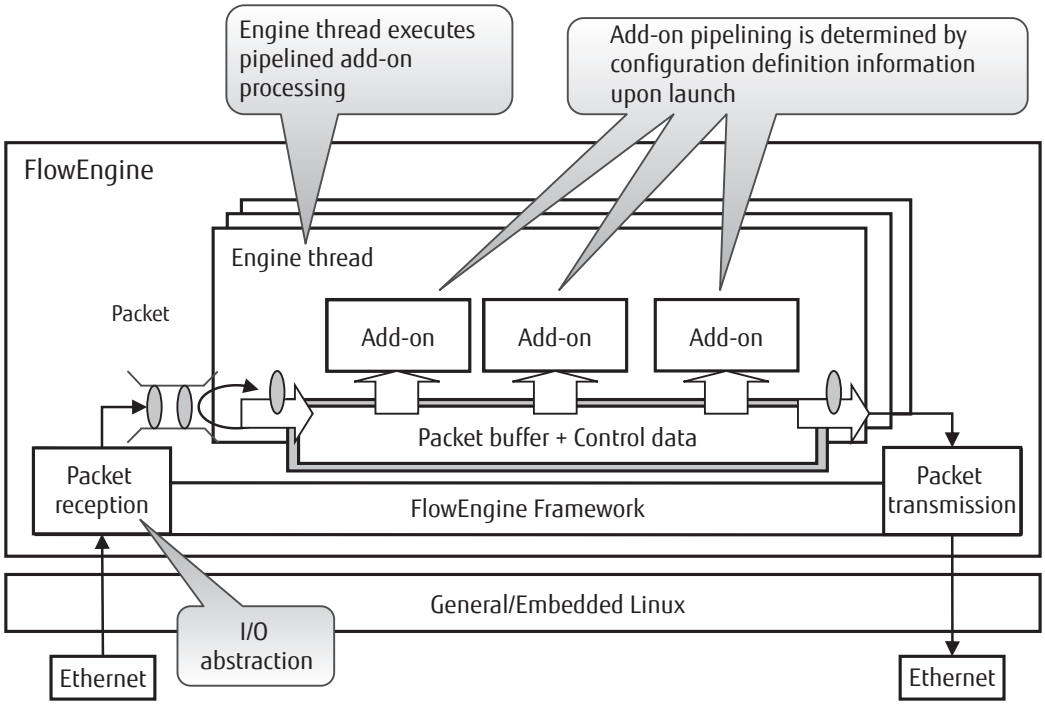

Figure 7
FlowEngine processing overview.

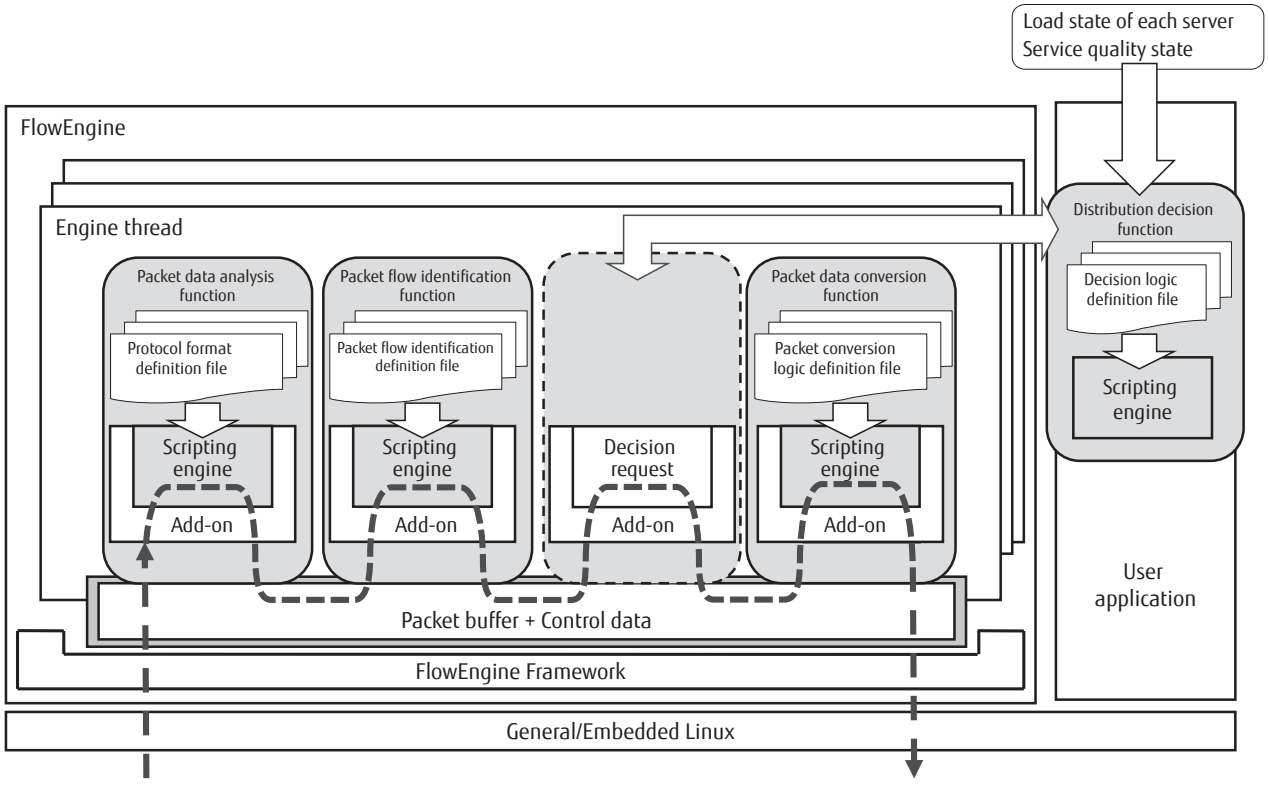

Figure 8
Programmable network infrastructure with embedded scripting engines.

request add-on within FlowEngine to the distribution decision function within the user application, and the distribution decision is performed by the scripting engine within the user application. The result is then returned to the add-on that sent the distribution request. When deciding packet data distribution, service quality information such as the load state and response time of each server is used, but this information must be gathered from the available servers as it is not managed by FlowEngine. Thus, the distribution decision function is positioned as a user application.

5. Future issues

Since the programmable network infrastructure is implemented with definition files run by scripting engines, the quality and performance of distribution processing is greatly affected by the content of these files. Issues needing to be addressed include simplifying the syntax of the definition files to reduce errors, creating a system for verifying the correctness of the logic described in the files, and coding the files so that they are resistant to degradation in performance.

To further increase capacity, we plan to scale out the programmable network infrastructure itself.

6. Conclusion

We have described networking infrastructure technologies needed to build a scalable platform for implementing high-capacity, highly reliable services as well as initiatives at Fujitsu in this area.

Since services will continue to diversify, we will continue our technology and product development efforts and actively promote creation of high-capacity, highly reliable services.

Yuzuru Iida
Fujitsu Ltd.
Mr. Iida is currently engaged in the development of network solutions for communications operators.

Hiroaki Tanaka
Fujitsu Ltd.
Mr. Tanaka is currently engaged in the development of network solutions for communications operators.