Design and Implementation of the Operation and Maintenance System in Component-Router∗
Yong Jiang, Jianping Wu
[jyong@cns1.cs.tsinghua.edu.cn]
Department of Computer Science and Technology, Tsinghua University, Beijing, 100084
Phone: +86-010-62785822 Fax: +86-010-62788109

Abstract:
The OAM (Operations and Maintenance) module of a router fulfills the management and manipulation of the router. The OAM is one of the most important sub systems in a router, guaranteeing it to work in going order. With the rapid development of the network protocols, it is increasingly important to dynamically upgrade router's software. In this paper, the authors make a thorough research on the operation and maintenance system in routers. First the CR-OAM (Component-Router-OAM) design requirements and evolutions are introduced. Then the authors present the Component-Router's architecture and the CR-OAM function requirements, that Component-Router is developed by Tsinghua University in China, the design and implementation of the CR-OAM is also described in detail, whose novel feature is the ability to dynamically load and configure extensible component at run time. At last research directions and open problems in this area are discussed.

Keywords: OAM, Extensible component, Router

1. Introduction
The management of the router includes performance management, configuration, traffic measurement, fault tolerance and security. They are defined in reference [1] and [2].

With the development of Internet, routers should incorporate several new services in addition to forwarding packets:

- Integrated/differentiated Services
- Enhanced routing function (level 3 and level 4 routing and switching, QoS routing)
- Security algorithms (e.g. to implement virtual private networks (VPN)) and packets filtering (e.g. Firewall)
- Enhancements to existing protocols(e.g. congestion control algorithm resembling Random Early Detection (RED))

New core protocols (e.g. IPv6)

We call the router, which provides these extended functions, Component-Router (CR), while call the traditional router Best Effort Router.

Tsinghua University designs and implements the modular and extensible Component-Router’s prototype system, on the project of “High Performance and Multiple Protocols Router”[3], which was the National High Technology Research and Development Plan (863) Key Project in 1999. This article introduces the design and implementation of OAM subsystem in the Component-Router in detail. The OAM subsystem is the essential module in a router's software. Component-Router has special requirements on the architecture and functions of OAM subsystem, e.g. supporting modular and extensible components, dynamically loading extensible components, scalability, run-time configuration and so on. We produce our own design thought and discuss related works.

1.1 Related Works
There are many correlative research reports about the similar topic. For example, Microsoft Routing and Remote Access Service (RRAS) [4] used in Window NT 4.0 and 5.0 server is easy to operate and configure. However, RRAS is limited in scalability. ALTQ[5] developed by Sony Corporation is the architecture used for packet scheduling. It has implemented Class Based Queuing (CBQ, [6]) used for bandwidth reservation and DRR module, but it does not allow dynamically loading modules. Extensible router project of Princeton University has implemented Extensible Router Architecture [8] based on Scout Operating System [7]. It has implemented DRR [9], Virtual Clock [10] and WFQ [11] algorithm and so on. The search group in Washington University introduces Router Plugin Architecture [12] used to extend functions of Integrated Services Router. The architecture is based upon NetBSD, and only supports flow configuration. It does not support line card configuration of the distributed router, and the performance of its classifier is poor, so it is difficult to be used in a high performance router.

∗ This research was sponsored by CNSF under grant number 69822002 and HTRDP(863) under grant number 863-306-2D-07-01.
In the rest of the paper, we provide details on both the CR-OAM design and the different components it relies on, and present some initial results on the performance and capabilities of the implementation. In Section 2, we provide a short overview of the Component-Router system architecture, and highlight its main functions and component. In Section 3, we introduce function requirement and total design of CR-OAM system. Section 4 describes the design and implementation of the modular and extensible OAM subsystem. Finally, Section 5 summarizes some of the experiences gathered from this work and lists a number of extensions and enhancements that are currently being worked on.

2. Software and Hardware Architecture of Component Router

2.1 Hardware Architecture
CR employs distributed multiprocessor architecture, which accords with “High Performance Multiple Protocol Router” developed by our group. Figure 1 shows this architecture as follow:

1 Central Processor Module (CPM):
   - Processing routing protocol packets
   - Maintaining global routing table
   - Keeping synchronization of local routing table and global routing table
   - Fulfilling operation and configuration of routers and network management function
   - Maintaining global security data base
2 Extended Processor Module (PM):
   - Packets forwarding with local routing table
   - Implementing packets filtering and scheduling algorithm according to the configuration of the OAM system
   - Supporting various data link protocols
3 Interface Unit (IU):
   - The driver of Physical Communication Unit (PU)
   - Supporting high speed interface

2.2 CR Software Architecture
CR is a kind of high performance backbone router which supports various routing and network protocols and provides various extended service function such as QoS, IP security and so on. Its total function framework (Figure 2) is composed of five main parts:

- OAM Subsystem
- Supporting Subsystem
- Routing Subsystem
- Forwarding Subsystem
- Interface Unit Subsystem

Besides these composing parts hereinbefore, CR also provides extended service module for configuration.

Component-Router (CR) providing extended function and traditional Best Effort Router are very different in software architecture. Figure 3 contrasts the two routers.

Comparing to Best Effort Router, CR kernel has such increased components: packets scheduler, packets classifier, security mechanism, routing and switching based upon QoS, firewall and congestion control mechanism.

In routers, different algorithms are distinct in the facet of performance and system overhead. Most algorithms are refreshed and must be replaced or upgraded frequently, so such router software components have relative “Fluidity”.

Figure 1. Component-Router hardware architecture

Figure 2 CR Software Architecture
In this paper, we propose a CR software architecture, which meets following requirements:

- **Modularity**: Implementation of specific algorithms comes in the form of modules called Extensible Components (EC), and programming interface is provided, so network administrator can use EC language to make new EC.
- **Extensibility**: New components can be dynamically loaded at run time.
- **Flexibility**: Instances of EC can be created, configured and bound to specific flows. EC can be all software modules, or they can be software drivers for specialized custom hardware. We design one CR Configuration Language (CRCL), and network administrator can use it to define new flow pattern and configure ECs to deal with the flow.
- **Performance**: The system should provide a very efficient data path, with no data copying, and no additional interrupt processing. The additional overhead of system should not seriously impact performance.
- **Distributed process**: The system supports distributed router hardware architecture, and can be configured based on both per flow and per port.

### 3. Function Requirements

To be brief, the fundamental function of OAM subsystem is to maintain and manage router.

In CR, OAM subsystem lies in the highest layer and is also the earliest starting task of the system. And then it starts and loads supporting subsystem, routing subsystem, interface unit and other modules. It initializes and configures other modules and allocates system resources. So the function of this module is very complex, and plays an important part in the security, reliability and steady running of routers and network [13], [14], [15], [16].

#### 3.1 Function Requirement of OAM Subsystem

The goal is to establish an extensible modular router system, which must have the concept of network flow and can achieve component selection based upon per flow. So we think that OAM subsystem should include such enhanced functions besides those defined in [1], [2]:

1. **Dynamic loading/unloading of Extensible Component (EC) at run time into/from the router subsystem.** EC is a code module which implements a specific extended function. And it is required to glue the individual component to the system and to provide a control interface. There must be an EC Control Management (ECM). ECM hides EC's specific implementation details and allows them to access the whole system in a simple and flexible method.

2. **Creating individual and flexible instances of Extensible Component (EC).** An instance is a run-time entity of an individual EC.

3. **Flexible configuration and management.** Configuration language CRCL is used to define new flow patterns and configure the EC modules to process the flow.

4. **Efficient mapping of data packets to flows, and the ability to bind flows to EC components.** Sets of flows are specified using filters. Filters are specified as six-tuples:

   `<Source address, destination address, protocol, source port, destination port, incoming interface>`

   Any of the fields in the six tuples may be wildcard.

5. **Guarantee high performance.** We guarantee the smallest additional system overhead by fewest data copying, quick and efficient classifying strategy, stream-lined data path, and so on. Classifying algorithm uses Recursive Flow Classification (RFC) [17] and combines cache memory mechanism. The data transmission of packet flows binds EC only for the first packet of a burst. Subsequent packets
can be flow-like transmitted quickly.

6. Supporting distributed processing platform. Before this, almost all of extensible router architecture can not support distributed hardware environment, and this limited their application scope. But our software architecture is based on distributed hardware (Figure 1), the forwarding performance is awfully increased, so it can be used as high performance backbone router.

3.2 EC and CRCL

In Router Plugin architecture [12], it only dynamically extended the service through the IP processing path in router, and we think that is far from enough. So after thorough analyzing, we divide all router EC into four types of component.

1. Node component: node component has nothing to do with IP processing path, and it belongs to upper application in router, e.g. TCP, BGP. It can work on any nodes as configuration.

2. Port component: each port component lies on one port, input or output port. This kind of components are mainly classifier or scheduler. Classifier commonly deploys on input port, and scheduler can be deployed on both input and output port.

3. Component based on network flow: These components are similar with Plugin in Router Plugin architecture, used to extend services for IP packets processing.

4. Cache component: They are special, because they are used to connect different components to implement full data path.

In order to configure system easily, we develops a CR configure language CRCL, which is a text model language. It can be used like DOS’s batch file. Its main functions include flow pattern description, component definition, component operation definition. We introduce it as follow examples.

Flow pattern description:

**Definition 1:** A cell is the fundament element, and one cell is a tetrad <offset, length, mask, value>. One packet is matching with one cell, if and only if the result of the packet’s length bytes from offset AND mask equals to value.

**Definition 2:** A line is a set of cells. One line is matching with packets, if and only if each cell in line is matching with this packet.

**Definition 3:** A flow pattern is a duality <head_len, line>, head_len is the length of the protocol header, line is corresponding line.

In figure 4, TCP traffic is described with CRCL, and it defines TCP connection from source port 1111 to destination port 2222. Among it, # denotes notation. Only fixed TCP header is described because of simplicity.

**Component definition:**

**Definition 4:** Component definition is a group of five elements <name, class, input_interface, output_interface, configuration_string>, denote component’s name, type, input interface, output interface and configuration string.

**Component operation definition:**

Component operation includes three classes:

1. Running node component, such as defining which node running RIP. One of important things is to assure all correlated components at the same node. The CRCL compiler can check it, if error found, system return to last correct state.

2. Running port component. Port component should bind with certain port, so port must be specified. The detail will be discussed in CCU.

3. Component control path based on network flow. The control path defines the full path processing the flow.

In figure 5, a simple component operation definition is shown. “->” denotes calling for the component at the same node, while “=>” denotes calling for the component at other node. The definition of component is ignored because of simplicity.

- # define two bytes length source port
tcp_cell1 = (0, 2, 0xffff, 1111)
- # define two bytes length destination port
tcp_cell2 = (2, 2, 0xffff, 2222)
tcp_line = (tcp_cell1, tcp_cell2)
- # 20 bytes fixed header information
tcp_header = (FIXED, 20 , 1)
tcp_pattern = (tcp_header, tcp_line, null)

Figure 4 TCP traffic defined with CRCL

- # run RIP at node 1
  node1: RIP -> UDP -> IP
- # run RR scheduler in port 1 at node 2
  node2, port1: input_buff -> RR_send
- # audit TCP traffic in Figure 4
  tcp_pattern: eth_buff -> Counter ->
  output_buff => input_buff -> FIFO_send

Figure 5 Component operation definition
4. Design and Implement of CR-OAM

4.1 Overall Design of OAM Subsystem

Management commands from a network manager are converted into the format of management protocol request message, translated by OAM command interpreter and handed by OAM management module. OAM management module communicates with executive module through backplane bus or switch fabric to achieve the goal of network observation, configuration and management. Figure 6 shows the architecture of OAM.

For supporting distributed hardware platform, our software architecture is distributed, composed with one center processing node and several periphery nodes. Center node include:

- **Extensible Component Library (ECL):** The ECL stores all EC code modules.
- **Extensible Component Manager (ECM):** The ECM is the core of overall OAM system. It manages EC, including receiving configuration information from ECCI, upgrading new ECL, FPL and CIB. It is responsible for working with ECMA in peripheral nodes, including maintaining synchronization between nodes, forwarding messages to individual EC reliably at run time from other kernel components and subsystem.
- **EC Configure Interface (ECCI):** ECCI is a user interface unit, used to configure the system. It controls and configures Configuration Information Base (CIB) of EC through command line and configuration file based on CRCL language. Then the system configures EC bound to flows according to the data of EC CIB.
- **Flow Pattern Library (FPL):** The FPL records user defined flow pattern. The structure is \{pattern, index\}, pattern is flow pattern, introduced in last section, index is one unique value, which is used to look up configuration information according this pattern in CIB.
- **Configure Information Base (CIB):** The CIB records system configure information, includes EC information according to per flow pattern and information of node component and port component.

Periphery node include:

- **Extensible Component Manage Agent (ECMA):** The ECMA cooperates with ECM in center node to fulfill components’ installing, loading and execution. When system boots, ECMA communicates with ECM by synchronization protocol, guaranteeing every node work in order. Then ECMA downloads FPL and CIB from ECM to local database. After that, it downloads and executes node component and port component according CIB information, and sends downloading request to ECM according flow pattern at run time.
- **Classify Control Unit (CCU):** The CCU implements a packet classifier and builds the glue between the flows and EC. It maintains three data structures:
  - Local Configure Information Base: Local CIB in periphery node;
  - Local Flow Pattern Library: Local FPL in periphery node;
  - Local Flow Cache: This is a kind of cache mechanism, used to fast look up flow pattern.

After system reboots, configuration and selection of EC set must be done before transmitting packets. Because not all packets bind the same EC set, it requires configuring EC instances for per flow. Configuration can be implement through network manager or description files. The steps involves the following sequence of events and actions:

- Center node and periphery nodes start on order according to synchronization protocol, The detail of the protocol will be ignored because of space limitation.
- Periphery nodes download local FPL and local CIB from center node, and clear flow cache.
- Center node loads EC according to CIB: Using the load() function, component module codes are loaded.
into memory. Then EC are loaded into the kernel as function calling. On loading, they register
themselves with the ECM by providing a callback function. This function is used to send messages to
the EC. There are messages for creating and freeing instances of the EC and for binding EC instances to
flows. Once the callback function for an EC has been registered, the ECM can forward these
configuration messages to the EC.

- Periphery node loads EC according to local CIB.
  The EC needs to register with the ECMA.

Now the system is ready to process data packets.

### 4.2 Extended Component Management (ECM)

The ECM manages a table to store the EC’s type, name and
callback function. Once loaded into the kernel, EC registers
its callback function in the ECM. All control paths to EC go
through the ECM. The ECM is responsible for dispatching
these messages to the target EC, and handling exceptions.

Besides managing Extended Component, ECM can complete basic operations of OAM subsystem. It provides
the ability to control any system module directly and
improves the system’s ability to counteract serious network
faults or attacks efficiently.

### 4.3 Classify Control Unit (CCU)

The CCU is the most important component to implement
modular and extended service in OAM subsystem. It
implements a packet classifier, fast flow detection, and
provides the binding between EC instances and filters. To do
so, it manages two main data structures: filter table and flow
table.

Flow table is implemented using hashing. We use the five
tuple of header fields \(<source address, destination address, protocol, source port, destination port>\)
from the packet to calculate the hash index. The array for the hash table is
allocated at system boot time. Its size is dependent upon the
environment in which the router is used; the default value
used in our kernel is 32768.

Next we will introduce the implement of RFC [17]
algorithm for filter table. A filter table is used to classify
packets. Several generic packet-filtering algorithms
[18,19,20] are very powerful and flexible, but their
implements are very complex and show a poor performance
at high speed. However, our goal is to find a fast lookup
algorithm for matching the six-tuple \(<source address, destination address, protocol, source port, destination port, incoming interface>\).

We have implemented best-matching filter lookup using
RFC algorithm. RFC includes pretreatment and lookup.
Pretreatment is implemented with software, while lookup
can not only be implemented with software, but also with
hardware. Of course their performance are very different. To
guarantee the generality and consider that software
implementation can also provide the required performance,
we choose software to implement the whole RFC.

Under the worst condition (using three-period lookup), the
code executive time of total lookup process is \((140\text{ clks}+9\text{ t_m})\),
in which clks is clock circle and \(t_m\) is memory access time, if
RFC algorithm is implemented using software.

### 4.4 Data Packet Processing

In our CR prototype system, packets are sent to the EC
instance for implementing specific function. Because
packets processing impacts system performance directly,
next we will introduce the data packet flow process in detail
with examples.

We specify the processing steps of an IP packet. If the flow
does not exist in flow table: first, hardware interface receives
the packet and sends it to IP core; IP core finds appropriate
EC instance and calls CCU, while parameters are array
pointer and EC identifier; CCU looks up in flow table at first,
and if fails, looks up in filter table (because lookup steps are
the great overhead of system, quick and efficient lookup
mechanism is needed); after finding, CCU stores instance
pointer in flow table and return the handle to IP core; call
EC instance and process packet; repeat above steps and go
on.

After processing the first non-cache-memory packet, there is
a faster path for processing because of flow table: the packet
gets flow index which records relative EC instance for
processing from flow table, then gets EC instance to handle.

The implement mechanism has the characters of high
modularization and the smallest system overhead. In
addition, because system overhead is only related to the
lookup of first packet, the extensibility is very good.

### 5. Conclusions and Future Work
The main contribution of this paper is the design and implementation of OAM based upon the requirement of Component-Router modular architecture, the run-time loading of modules, system extensibility, run-time configuration and so on. The system is extensible and flexible very well, and implements the quick classification of packets. Under the condition of a large number of filters, the classification of an IPv4 packet only needs 9 memory accesses at most. Smaller the number of filters is, fewer it needs to access memory. It can provide the general management and maintaining to Component-Router, guaranteeing the dynamic extensibility.

OAM system is very important to the running and management of routers. At the same time, the appearance and application of new network technologies must require more powerful OAM. To consider the above, OAM can do more researches at following aspects:

1. Adapt the requirements of new network protocols. With the appearance of new network technologies and protocols, new requests to OAM are emerging.

2. Increase network security performance. The development of network brings the security problems, which are not only the technology ones but also become the society ones increasingly. The improving method usually takes advantage of communication cryptology, then needs to maintain and manage the secret key. In addition, authentication mechanism must be imported to implement the safe OAM.

3. Combine the management function of operating system to router OAM.

4. Import active network mechanism [21, 22] to OAM.

REFERENCES