A Framework for Network State Management in the Next-Generation Internet Architecture

Xin Huang, Sivakumar Ganapathy, and Tilman Wolf
Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, Massachusetts 01003
Email: {xhuang,sganapat,wolf}@ecs.umass.edu

***Abstract*—In the next-generation Internet architecture, more functionality will be placed in the data path of routers in the form of "services" and other packet processing features. This trend will increase the amount of state information that is maintained by system inside the network. In this paper, we present a framework for structuring and managing network state information distributed across the next-generation Internet architecture.**

I. Introduction

The ubiquitous use of networked computer systems in our daily lives has expanded the requirements on the Internet: security and privacy have become important considerations; sensor networks and mobile wireless devices require communication paradigms different from the traditional client-server mode, service providers require mechanisms to differentiate their offerings beyond simply providing raw bandwidth. The current Internet architecture has been vastly successful in providing end-to-end connectivity, but cannot be easily extended to address these new demands. Instead, the networking community is in the process of considering a complete redesign of the Internet architecture [1].

A major trend among the network architecture design is to place more functionality and network state information inside the network [2]–[4]. Instead of constraining routers to simply forward packets and leaving all processing tasks to the end-systems, programmable processing components (e.g., Network Processors) of the routers are used to implement complex packet processing services. This change in philosophy acknowledges an already existing trend towards more functionality in the data path: firewalls, NAT boxes, intrusion detection systems, and similar middleboxes already perform processing tasks, but have been perceived as add-ons that do not match with the current Internet architecture. In the next-generation architecture, these functions exist as explicit services or network processing. On the other hand, an increasing number of stateful applications, such as IP traceback and stateful firewalls, worm detection, and packet inspection, are emerging to meet the security and manageability requirement of the next generation Internet [5].

It is our expectation that the above trend will lead to a situation that more and more network state information exist inside the network. Furthermore, in many cases, the network state information needs to be exchanged between different components of the same network system, or even between different network systems(e.g., between routers, or between router and end system). There are numerous challenges associated with state management in such an environment:

- **Diversity in Hardware Implementation of Local State Access:** Many existing network systems use a variety of different registers, queues, and memory subsystems to store packets, networking state, and processing state. For example, an Intel IXP network processor [6] uses several types of memory in a single embedded packet processing system: general-purpose registers, registers to access SRAM, SDRAM, next-neighbor interconnect, and FIFO buffers for external I/O interfaces. For each type of memory, a programmer needs to use a different assembly instruction or subsets of registers and understand performance implications.
- **Lack of Standardized Mechanisms for State Exchange Across Network Systems:** There are numerous approaches for exchanging state information within a system (e.g., shared memory, FIFO queues, etc.) and across systems (e.g., message passing, RDMA, etc.). The choice of mechanism depends on a number of system parameters. No single access mechanism has crystalized as a general standard.
- **Need for Handling Different State Contexts:** When referencing state, it is necessary to consider the context in which it is accessed. The context may depend on the packet flow, the processing service, or the network node. To simplify access to state information, such context information should be used when referencing state. Ideally, a network system should allow a variety of different context bases.

To avoid complexities associated with numerous point solutions for the above challenges, we believe that it is important to develop a broader state management solution. In this paper, we propose a framework for structuring and managing all network state information spread throughout the Internet architecture. Specifically, the contributions of this work are:

1) Organization of all network state information on network systems in a single three-dimensioned state space. This is the basis of our state information management system.
2) Unification of addressing and access of network state information spread across the network.

The remainder of the paper is organized as follows. Sec-

tion II discusses related work. Section III presents the state space architecture. Section IV discusses state management, and Section V summarizes and concludes this paper.

II. Related Work

In the design of the current Internet, routers perform stateless store-and-forward operations [7]. Over time, state information from the application and transport layer has found its way into the network (e.g., NAT [8]) and new stateful networking features have been proposed (e.g., IP traceback [9] or stateful packet inspection [10]). In the context of next-generation Internet architecture designs, many types of new services in the data path have been proposed [2] and network designs with inherently stateful designs are being explored in ongoing research projects [3], [4]. The use of network virtualization [11] is likely to further increase the diversity of network uses and associated stateful processing.

The processing aspect of such data path functions and services has been studied extensively. Embedded multicore processors called network processors (NP) have been developed specifically to be placed in the data path of networks [12]. These NP designs have been explored with respect to system architectures [13], memory architecture [14], run-time support [15], and alternative processing architectures [16]. Network processors have been proposed as infrastructure components in the experimental next-generation Internet backbone [17].

The aspect of state – which is closely tied to processing – has received less attention. System design tradeoffs for network processors change if stateful processing is performed [5]. Network protocols can be distinguished by the use of state information in the network (i.e., hard state vs. soft state) [18]. Protocol state is complemented by processing state, which is used to store temporary information. Ephemeral state processing uses small amounts of such temporary state on routers to implement advanced network services [19]. It has also been proposed to carry temporary processing state within packets [20]. While these studies address state in the context of network systems and protocols, our work address the issue of network state at a network architecture level. Our contribution is an overall framework for state management across the entire network and a discussion on how state access can be implemented across different network system components.

In a broader context, there is a discussion if data path processing should really be part the next-generation Internet architecture. Alternate visions propose an all-optical network [21] or high-speed routers that can process packet quickly and thus require hardly any buffer space [22]. However, even if these technologies are used for the core of a future Internet, we still expect the edge network to become a place where services and network processing tasks dominate. Apart from the technical need for these services, there is also a business reason. Network service providers need to be able to distinguish their offerings from competitors through services [23] rather than simply providing raw bandwidth.

III. State Space Architecture

Before presenting the state space, we give a brief overview on where state appears in the network.

A. State in Network Systems

Network state information in our work refers to the state information that is maintained in the memory of the network systems for packet processing. Some illustrative examples of state information inside routers are:

- **Control Path Router State:** routing table, forwarding information base, NetFlow measurement data, etc.
- **Data Path Router State:** packet queues, packet counters, flow table for NAT, etc.
- **Data Path Processing State:** flow classification result of packet, temporary data structures for IP prefix lookup, etc.
- **End System Protocol State:** TCP connection state, application layer state, etc.

All this state information is important and relevant for the network to operate correctly. Since state information is fragmented across a number of different nodes and system components, there is no single, unified mechanism to access all of it. Currently, different techniques are used to exchange different types of information between components (e.g., direct memory access to move forwarding information base from the router's control processor memory to the port memory, or a TCP connection to read NetFlow measurements from a system). It would be beneficial to have a single system that manages all state across network nodes and system components. Such a system could hide the complexities of accessing and exchanging state information between entities when implementing protocols and service.

B. State Space

To be consistent with the Internet structure, we organize the varieties of state information across the network into a three-dimensioned state space, as illustrated in Figure 1. The three dimensions in the figure are:

- **Context Dimension**: State can be associated with different services. Traffic belong to different services have different requirements (e.g., service quality, resource reservation), go through different processing functions, and thus need different state information to be maintained over the network.
- **Space Dimension**: From the source, packets travel through different routers to reach the destination. Different routers or end systems along the path may maintain their own state and thus needs to be considered as separate nodes in our framework.
- **System Dimension**: A router system (or an end-system) may consist of different levels of components that maintain their own memory and network state. The number of levels and the type of system components vary with the system architecture of each node.

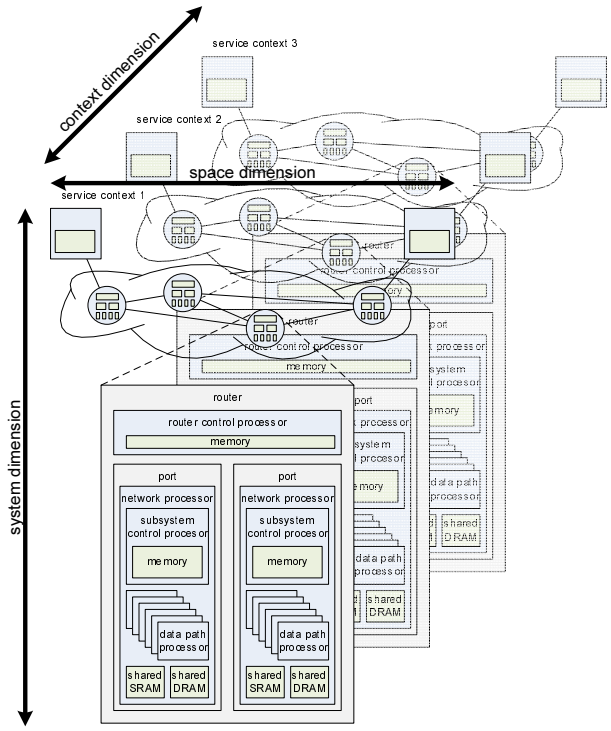

Fig. 1. Structure of Network State Space.

IV. State Management

Our proposed state space architecture can be roughly compared to a virtual memory system on a workstation computer. Where virtual memory hides details of the physical location of data, our state space architecture hides the access mechanism that are necessary to access data from different systems and system components. Our proposed state information management framework includes three major components: state reference, address mapping, and access mechanism.

A. State References

The proposed framework for state management provides access to all this state through a common interface. When accessing any state, it is important that a processing component can specify which state information it needs to access. Thus, the virtual address needs to contain sufficient information to identify the network node and system component to which the access is directed. In addition, it is necessary to identify the packet, protocol, and service context. This virtual address is then translated into (1) a physical address on the system where the data is stored and (2) a mechanism for accessing this data from the system that is requesting the access. We first briefly describe the abstractions we use in the virtual address space. The translation to physical address and possible access mechanisms are described in Sections IV-B and IV-C.

1) Network Nodes: The distinction between different nodes follows the typical concepts used in networking. There are different types of references that are allowed in order to provide convenient reference mechanisms:

- **Absolute Reference:** A system can reference a node by name.
- **Topology-Based Reference:** A system can reference other nodes based on topological properties (e.g., a neighbor, or a node in the same administrative domain).
- **Connection-Based Reference:** A system can reference other nodes based on characteristics of a particular connection (e.g., previous hop, or destination end-system).

The mapping process described later ensures that the references are resolved in the correct context.

2) System Components: Implementations of router systems can be as simple as a workstation computer with multiple line cards and as complex as multi-rack systems with numerous line cards, port processors, and control processors. In order to capture this diversity of systems, we use a hierarchical structure (the system dimension in Figure 1) to represent the key processing components of a router architecture and their associated memories. This hierarchy can be adapted to different specific implementations by adjusting the number of levels in the hierarchy and the number of components on each level. The hierarchy of system components is:

- **Router Control Processor and Memory:** There is at least one control processor in each router, which implements routing and other control path functions. The associated memory presents the top level of the system hierarchy.
- **Subsystem Control Processor and Memory:** Scalable router designs employ control processors on line cards, router ports, etc. Therefore we allow multiple levels of subsystems (e.g., blade, port, interface, etc.) with control processors and associated memory.
- **Data Path Processor and Memory:** Each router has to have a processing system that handles data path operations. Any associated memory is represented in this level. There may be multiple processors that operate in parallel on the data path.

When referencing a particular memory, a component may be addressed either directly or relatively to another (e.g., one level up in the control path).

3) Contexts: To make network-wide state management practically useful, it is important to provide access control according to context. For example, protocol operations may need to access state that is provided in the context of this protocol, but cannot or do not need to access any other state. Requiring references in the context of all state in the system may be too complex. Instead, the system automatically maps state accesses into the context of the protocol. Potential contexts for this purpose are:

- **Packet Context:** When processing packets on a network processor or other data path processing system, many state accesses are performed on the packet data itself. There is also temporary processing state that is associated with the packet that that may be created during packet processing (e.g., flow classification result).
- **Protocol Context:** Many protocols have state information

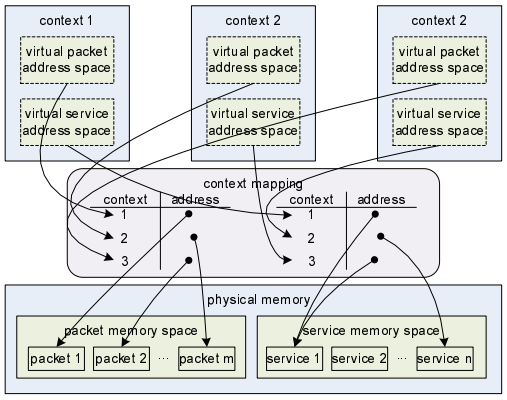

Fig. 2. Context Mapping in State Management Framework.

associated with them (e.g., routing tables, lookup data structures, or measurement statistics). This context contains such protocol-level information that is independent of any particular packets.

- **Service Context:** This context represents state that is associated with more complex processing tasks that may span multiple nodes and the end-system. It is possible to see protocol and service context as one type of context.

When referencing state from packets, protocols, or services, the system automatically provides the context in which the request is performed.

B. Address Mapping

The use of references that refer to a particular context (e.g., packet) can lead to a scenario where two identical access requests (e.g., read header of packet) can point to two different physical locations if the respective contexts differ (e.g., two different processors forwarding packets on a port). This means that a system that implements our state management system must be able to perform suitable demultiplexing operations to ensure that the correct memory location is found. This functionality can be achieved by using a mapping process similar to how it is done for segments in virtual memory. The only difference is that a mapping table exists for each context (e.g., packet context, protocol context, etc.). This concept is illustrated in Figure 2.

Clearly, an important practical question in this context is that of access control. For reasons of security and safety is may not be desirable that a packet or a service can access state information from other packets or services. This isolation can be provided elegantly through this mapping process. By enforcing one level of indirection to memory accesses, it is possible that the system can check each memory access. Memory that should not be accessed by a component simply is not mapped into the range of accessible state space, and any attempted access fails. This solution, of course, does not address the issue of policies on access control (i.e., what system component should be allowed to access what state and what context). It simply provides a mechanism to limit the state space that is accessible to a particular system component.

C. Access Mechanisms

When a physical memory address has been identified, the memory access is performed. Accessing local memory clearly is a straightforward process, but accessing memory on a non-local system component is a challenge. Different system components provide different access mechanisms to their memories and common mechanisms need to be identified.

We show a list of existing memory access mechanisms and their characteristics in Table I. They can be classified into the following categories:

- **Shared Memory**: Shared memory refers to a block of random access memory that can be accessed by several different processors in a multiprocessor system. Thus, a processor can access the memory space that also belongs to another processor. It can be further divided into two types: Centralized Shared Memory (CSM), also called Uniform Memory Access (UMA) [24], and Distributed Shared Memory (DSM), also called non-uniform Memory Access (NUMA) [25].
- **Direct Memory Access**: Direct memory access refers to an access to memory that belongs to another subsystem via a suitable interconnect. Direct Memory Access (DMA) is typically used for I/O devices to access main memory space within the same system. Remote Memory Access (RMA) [26] and Remote Direct Memory Access (RDMA) [27] allow memory accesses on different processor systems that are connected via a network (e.g., in massively parallel computer clusters).
- **Message Passing**: Message passing (MP) is a form of communication and state information exchange done by exchanging messages [28]. Message Passing can be used for interprocess communication in uniprocessor system, interprocessor communication in multiprocessor system, and intercomputer communication in computer cluster.

Table II shows which mechanism is most suitable when performing accesses between different system components.

V. Summary

In this paper, we have presented a framework for managing network state across different systems and system components. We have shown how references to different state can be used and how the system can map these accesses to physical memories. We also present an overview on potential access mechanisms for non-local memory.

We believe that finding a good approach to managing state in networks is important in order to manage the complexities of next-generation network services. There are still many open questions about the design and implementation of such a state management framework. Our proposed architecture is a first step in this direction and a basis for continued research and discussion.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant Nos. CNS-0447873 and CNS-0626690.

TABLE I
MECHANISMS FOR ACCESSING MEMORY ACROSS SYSTEMS.

Characteristics	Shared Memory		Memory Access			Message Passing
	CSM	DSM	DMA	RDMA	RMA	MP
Interconnect (IC)	local IC	local IC	local IC	network	network	local IC or network
Visibility	yes	yes	limited	limited	limited	no
Access performance	high	high	high	low	low	high or low
Range	short	short	short	long	long	long
Coherence issues	if cached	if cached	none	none	if cached	none
Example system	network processor		NIC	parallel cluster		grid computing

TABLE II
ACCESS MECHANISM CHOICE BETWEEN DIFFERENT SYSTEM COMPONENTS.

	Interface	DPP	SCP	RCP	End system
Interface	[DMA]	DMA	-	-	-
Data path processor (DPP)	DMA	SM/DSM/ MP	DSM/MP	DSM/MP	-
Subsystem control processor (SCP)	-	DSM/MP	DSM/MP	DSM/MP	-
Router control processor (RCP)	-	DSM/MP	DSM/MP	RMA/RDMA/ MP	MP
End system	-	-	-	MP	MP

REFERENCES

[1] A. Feldmann, "Internet clean-slate design: what and why?" *SIGCOMM Computer Communication Review*, vol. 37, no. 3, pp. 59–64, Jul. 2007.

[2] T. Wolf, "Service-centric end-to-end abstractions in next-generation networks," in *Proc. of Fifteenth IEEE International Conference on Computer Communications and Networks (ICCCN)*, Arlington, VA, Oct. 2006, pp. 79–86.

[3] R. Yates, D. Raychaudhuri, S. Paul, and J. Kurose, "Postcards from the edge: A cache-and-forward architecture for the future internet," Rutgers WINLAB, University of Massachusetts, Tech. Rep., 2006, http://www.nets-find.net/Postcards.php.

[4] D. Boneh, D. Mazieres, M. Rosenblum, A. Akella, and N. McKeown, "Designing secure networks from the ground-up," Stanford University, University of Wisconsin-Madison, Tech. Rep., 2006, http://www.nets-find.net/DesigningSecure.php.

[5] J. Verdu, J. Garcia, M. Nemirovsky, and M. Valero, "Architecture impact of stateful networking applications," in *Proc. of ACM/IEEE Symposium on Architectures for Networking and Communication Systems(ANCS)*, Princeton, NJ, Oct. 2005, pp. 11–18.

[6] *Intel Second Generation Network Processor*, Intel Corporation, 2005, http://www.intel.com/design/network/products/npfamily/.

[7] D. D. Clark, "The design philosophy of the DARPA Internet protocols," in *Proc. of ACM SIGCOMM 88*, Stanford, CA, Aug. 1988, pp. 106–114.

[8] K. B. Egevang and P. Francis, "The IP network address translator (NAT)," Network Working Group, RFC 1631, May 1994.

[9] A. S. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer, "Hash-based IP traceback," in *Proc. of ACM SIGCOMM 2001*, San Diego, CA, Aug. 2001, pp. 3–14.

[10] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, and J. Turner, "Algorithms to accelerate multiple regular expressions matching for deep packet inspection," in *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, Aug. 2006, pp. 339–350.

[11] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, Apr. 2005.

[12] T. Wolf, "Challenges and applications for network-processor-based programmable routers," in *Proc. of IEEE Sarnoff Symposium*, Princeton, NJ, Mar. 2006.

[13] D. E. Comer, *Network System Design Using Network Processors*, 1st ed. Prentice Hall, 2003.

[14] J. Mudigonda, H. M. Vin, and R. Yavatkar, "Overcoming the memory wall in packet processing: Hammers or ladders?" in *Proc. of ACM/IEEE Symposium on Architectures for Networking and Communication Systems (ANCS)*, Princeton, NJ, Oct. 2005, pp. 1–10.

[15] T. Wolf, N. Weng, and C.-H. Tai, "Run-time support for multi-core packet processing systems," *IEEE Network*, vol. 21, no. 4, pp. 29–37, Jul. 2007.

[16] J. W. Lockwood, N. Naufel, J. S. Turner, and D. E. Taylor, "Reprogrammable network packet processing on the field programmable port extender (FPX)," in *Proceedings of the 2001 ACM/SIGDA Ninth International Symposium on Field Programmable Gate Arrays (FPGA '01)*, Monterey, CA, Apr. 2001, pp. 87–93.

[17] J. S. Turner, "A proposed architecture for the GENI backbone platform," in *Proc. of ACM/IEEE Symposium on Architectures for Networking and Communication Systems (ANCS)*, San Jose, CA, Dec. 2006, pp. 1–10.

[18] P. Ji, Z. Ge, J. Kurose, and D. Towsley, "A comparison of hard-state and soft-state signaling protocols," in *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, Karlsruhe, Germany, Aug. 2003, pp. 251–262.

[19] K. L. Calvert, J. Griffioen, and S. Wen, "Lightweight network support for scalable end-to-end services," in *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, Pittsburgh, PA, Aug. 2002, pp. 265–278.

[20] J. T. Moore, M. Hicks, and S. Nettles, "Practical programmable packets," in *Proc. of IEEE INFOCOM 2001*, Anchorage, AK, Apr. 2001, pp. 49–59.

[21] C. Qiao and M. Yoo, "Optical burst switching (OBS) – a new paradigm for an optical Internet," *Journal of High Speed Networks*, vol. 8, no. 1, pp. 69–84, Mar. 1999.

[22] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Part III: Routers with very small buffers," *SIGCOMM Computer Communication Review*, vol. 35, no. 3, pp. 83–90, Jul. 2005.

[23] L. He and J. Walrand, "Pricing and revenue sharing strategies for Internet service providers," *IEEE Journal on Selected Areas of Communications*, vol. 24, no. 5, pp. 942–951, May 2006.

[24] J. L. Hennessy and D. A. Patterson, *Computer Architecture – A Quantitative Approach*, 2nd ed. San Mateo, CA: Morgan Kaufmann Publishers, Inc., 1995.

[25] D. E. Culler, J. P. Singh, and A. Gupta, *Parallel Computer Architecture: A Hardware/Software Approach*, 1st ed. Morgan Kaufmann Publishers, Inc., 1999.

[26] A. V. Gerbessiotis and S.-Y. Lee, "Remote memory access: A case for portable, efficient and library independent parallel programming," *Scientific Programming*, vol. 12, no. 3, pp. 169–183, Aug. 2004.

[27] A. Romanow and S. Bailey, *An Overview of RDMA over IP*, Cisco System and Sandburst Corporation.

[28] A. Silberschatz, J. L. Peterson, and P. B. Galvin, *Operating System Concepts*, 3rd ed. Addison-Wesley Publishing Company, 1991.