TNC-compatible NAC System implemented on Network Processor

An'an Luo[1], Chuang Lin[1], Zhen Chen[1,2], Xuehai Peng[1], Peter D. Ungsunan[1]
[1]Department of Computer Science and Technology, [2]Research Institute of Information Technology
Tsinghua University
Beijing, P.R. China
{laa, clin, zhenchen, xhpeng, hongsunan}@csnet1.cs.tsinghua.edu.cn

Abstract—**In this paper, based on the Trusted Network Connect architecture, we designed a novel TNC-compatible Network Access Control System which ensures that network administrators enforce security policies on endpoint connection and communication with corporate network depending on the endpoint integrity and security status. The platform framework is built on the Intel IXP2400 network processor and a set of network access control mechanisms is implemented. The paper introduces the system design and implementation based on hardware characteristic of the IXP2400 Architecture, presents emulation performance results of the system, and then proposes systemic performance optimizations, especially cryptographic performances, according to IXP2400 shared memory hierarchy and access latency, which averagely boost the throughput more than 25%. The novelty of system design is the utilization of IXP2400 multi-core and multi-thread network processor's software and hardware platform to implement the NAC system framework through secure and reliable communication to ensure endpoint integrity and platform-authentication, which is compatible with Trusted Network Connect.**

Keywords- TNC; network access control; network processor; AES algorithm

I. INTRODUCTION

Network Access Control (NAC) is defined as some control mechanisms which are enforced by network administrators at routers or proxies as to authenticate users and machines to authorize them to access the network [1]. Network access control technology plays an increasingly important role in computer network security, especially against security challenges brought by ubiquitous applications on the heterogeneous networks. A small quantity of unauthorized or unsecured network accesses can easily get across security mechanisms, like firewalls and authorization proxies, to compromise the whole network.

Traditionally, NAC can be classified as ingress NAC and egress NAC. Ingress NAC protects internal servers from external attackers and intruders while Egress NAC prevents internal users accessing undesired or dangerous external servers [2]. Both of the above NAC systems protect the internal network from being compromised from external network.

However, local network sometimes acts as a greenhouse of insecure network factors, due to its weakness in ensuring the security of the internal endpoint hosts' accesses. If any host computer without authorization or insecurity connects and accesses the external network without any protection, it could causes serious problems, such as worms, virus, Trojan horses, DOS attacks and other malicious activities, which could compromise many more hosts, or even the whole network. So mechanisms to keep every computer's security before connecting to the network will cut down the probability of virus bursts for the network.

Therefore, network access control technology puts increasing demands on endpoint integrity protection which is paid more and more attentions by network technical companies and research organizations. Trusted Network Connect (TNC) Architecture proposed by Trusted Computing Group (TCG) stresses endpoint integrity, and released the TNC Architecture for Interoperability Specification in May 2006 [3].

In the paper, we built a novel TNC-compatible Network Access Control system using the Intel IXP2400 network processor and its development platform to implement network access control mechanism which ensures that network administrators enforce security policies for endpoint host connections on corporate networks depending on the integrity status of the endpoint host. Based on multi-core and multi-thread network processor platform, we implement efficient cryptographic processing to strengthen security, the system architecture we present emphasized platform-authentication and secure communication, as well as endpoint integrity, which is not only compatible to TNC specification but also more flexible to system functional extend implemented on IXP2X00. To alleviate the memory-wall bottleneck and enhance system performance, we made optimization on cryptographic performance by adopting more effective memory allocation and utilization.

The rest of the paper is organized as follows. Section 2 provides a brief background introduction about the architecture of TNC and IXP2400 network processor, Section 3 and section 4 introduce system design and implementation in detailed. Section 5 we analyze the systemic performance bottlenecks in memory hierarchy and access latency and optimize the system design and cryptographic algorithm according to the hardware characteristic of IXP2400 network processor. The final section concludes the paper and our future work.

IEEE computer society

Authorized licensed use limited to: Tsinghua University Library. Downloaded on October 7, 2008 at 8:45 from IEEE Xplore. Restrictions apply.

II. BACKGROUNDS

A. *Trusted Network Connect*

Trusted Network Connect (TNC) is a new concept as well as an open architecture solution which was defined and promoted by the Trusted Computing Group (TCG). The TNC architecture will leverage and integrate with existing network access control mechanisms such as 802.1X [4] and others.

The goal of Trusted Computing as defined by the Trusted Computing Group (TCG) is to improve the trustworthy behavior of platforms and to permit trustworthy verification [5]. Through trusted network connection protocols and trusted platform mechanisms, entities seeking connectivity can be platform-authenticated and authorized (against some network policy) before being given full network connectivity.

As shown in Figure 1. The TNC Architecture adopts server-client mode, consisting of three entities which is Access Requestor (AR), Policy Enforcement Point (PEP) and Policy Decision Point (PDP). The AR is the entity seeking access to a protected network while the PDP is the entity performing the decision-making regarding the AR's request, in light of the access policies. And the PEP is the entity that enforces the decisions of the PDP regarding network access. The details of the Architecture and Specification of TNC can be found in TCG TNC Architecture for Interoperability Specification Version 1.1 Revision 2.

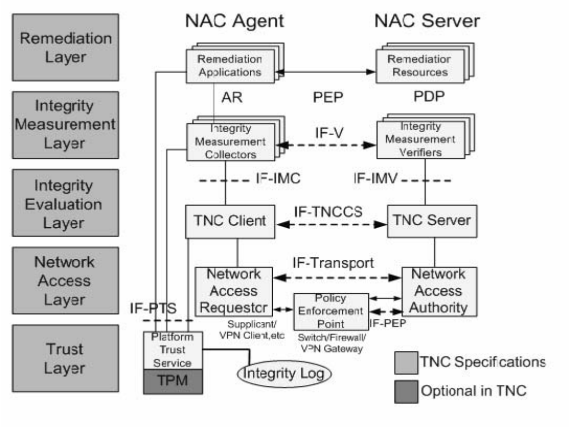

Figure 1. TNC Architecture

The TNC authorization process can be summarized to five stages as described in Figure 2. The AR entity collects the endpoint devices' integrity information about anti-virus software, firewall software, OS patch, and sends it to Server's verifiers on PDP through the PEP; PDP makes decisions and sets access control policies depending on the integrity aspects and security status. PEP enforces policies for endpoint host connections to their corporate networks and restricts the environment in accordance with access rights. Finally, PEP provides quarantine and remediation for endpoint devices to satisfy the security policy and provide eligibility for connection.

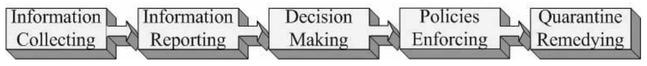

Figure 2. TNC Authorization Process

The TNC Architecture is not a totally new design which has many differences from other network architectures, but a complete set of platform authentication and security network connect mechanisms integrating several mechanisms and protocols of network access and connect control. Therefore, it is not necessary to design all the interfaces and protocols for specification and entities, which is much more convenient for us to implement a system platform.

As an industrialization specification, TNC is supported and promoted by more and more networking industry companies up to now, such as HP, IBM, Juniper Networks, Meetinghouse, Nortel, Symantec and Wave Systems, some of which have already developed products supporting the TNC specifications for network security and endpoint integrity.

B. *IXP2400 Network Processor*

The Network Processor Unit [6]-[7] enables network researchers and developers to add the latest network services while maintaining high throughput and low latency.

The system platform is implemented on the Intel IXP2400 network processor, which is suitable for integrated software and hardware multi-threaded and multi-core parallel computing. It consists of 8 multi-threaded MicroEngines (ME), XScale Core, 4K 32-bit word Scratchpad Memory, 2 QDR SRAM Interface controllers, 1 DDR DRAM Interface controller, Hardware Hash Unit, Media and Switch Fabric(MSF) Interface, Half Duplex OC-48/2.5 Gbps Ethernet Interface, and other Peripherals Components.

TABLE I. MEMORY HIERARCHY IN IXP2400

Memory Type	Logic Word Length (byte)	Capacity (byte)	Access Delay (clock cycle)
Local Memory	4	2560 per ME	3
Scratchpad	4	16K	60
SRAM	4	128M	90
DRAM	8	1G	120

System programming focuses primarily on writing data plane components utilizing MicroEngines, and application programs running on XScale core. As network processor's shared memory hierarchy increases difficulties in providing a balance between performance and flexibility while designing and programming, we attempt to design a more efficient memory allocation scheme. TABLE I. shows memory resources in one ME or shared by all the MEs described in [8].

III. T-NAC SYSTEM DESIGN

A. *Architecture of T-NAC System*

The framework of the TNC-compatible NAC system is illustrated in Figure 3. We call it T-NAC system. The prototype system's targeted usage scenario is Ethernet LAN. Local hosts act as Access Requestors in the internal network connecting to the IXP2400 through gigabytes optical bandwidth. Consisting of fast data plane and control plane processors, IXP2400 plays the role of Proxy Gateway as well as Authentication Server, which integrate PDP and PEP entities into one proxy.

Authorized licensed use limited to: Tsinghua University Library. Downloaded on October 7, 2008 at 8:45 from IEEE Xplore. Restrictions apply.

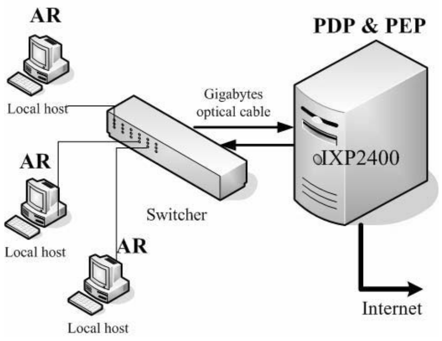

Figure 3. T-NAC System Framework

The architecture of the T-NAC System includes AR, PDP and PEP entities; each entity runs procedures in different logical layers as the TNC specification. The AR entity on a local host runs Linux OS, and the PDP entity is implemented in VxWorks OS running on a high-performance general purpose XScale CPU on the IXP2400, while the PEP runs on the MicroEngines.

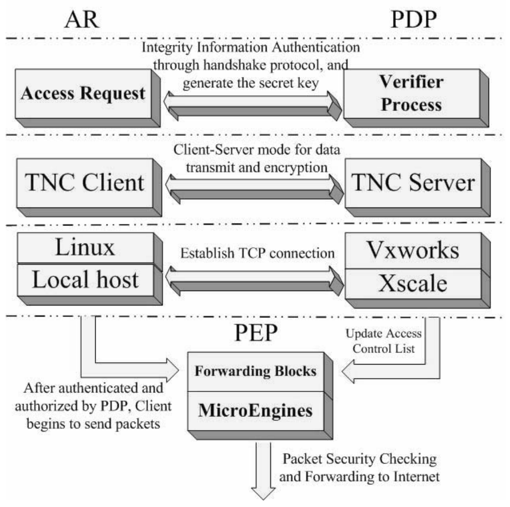
AR
PDP
Integrity Information Authentication through handshake protocol, and generate the secret key
Access Request
Verifier Process
Client-Server mode for data transmit and encryption
TNC Client
TNC Server
Linux
Local host
Establish TCP connection
Vxworks
Xscale
PEP
Update Access Control List
Forwarding Blocks
MicroEngines
After authenticated and authorized by PDP, Client begins to send packets
Packet Security Checking and Forwarding to Internet

Figure 4. The architecture of T-NAC System

As the goal of T-NAC system is security, compatibility and flexibility, it is compatible with the TNC architecture but not the same as the specification released by TCG, which is still being constituted as an industrial specification. However we strengthen protection of data integrity, platform-authentication, and secure communication between AR, PDP and PEP besides endpoint integrity by implementing efficient cryptographic processing using Advanced Encryption Standard (AES) algorithm with 128 bits block sizes and key sizes [9]. We ignored asymmetric-key ciphers which is too computationally expensive for MicroEngines. And both as a symmetric block cipher algorithm, AES algorithm are much more secure than DES. We populated AES encryption algorithm using open source code from OpenSSL[1] for reference.

[1] OpenSSL: http://www.openssl.org/source/openssl-0.9.7c.tar.gz

The total design of the T-NAC system includes two phases: Endpoint Integrity Measurement and Evaluation, and Access Policies Making and Enforcement.

1) *Endpoint Integrity Measurement and Evaluation*

The Endpoint Integrity Measurement and Evaluation process begins with AR trying to establish a connection with Authentication Server on PDP when the local host in the LAN requests access to the corporate network. PDP will verify and evaluate endpoint integrity information collected and sent by AR.

To protect encrypted communication between PDP and AR from any interception or modification by a third party, we adopted a Diffie-Hellman [10] secret key exchange to generate two cryptogrammic keys, which is the Session Key (SK) for message encryption/decryption during endpoint integrity authentication, and the Payload Encrypted Key (PEK) for policy enforcement on PEP[2]. The sequence diagram during Integrity measurement is illustrated in Figure 5.

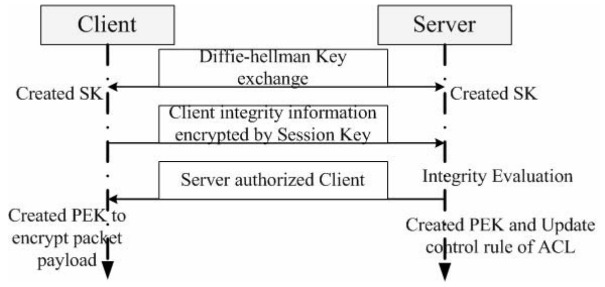
Client
Server
Diffie-hellman Key exchange
Created SK
Created SK
Client integrity information encrypted by Session Key
Server authorized Client
Integrity Evaluation
Created PEK to encrypt packet payload
Created PEK and Update control rule of ACL

Figure 5. Sequence Diagram during Integrity Measurement

2) *Access Policies Decision and Enforcement*

Through endpoint integrity measurement and evaluation, the T-NAC system is able to establish a "level of trust" and access policy of each authenticated endpoint before allowing connection to the network.

Network administrator usually describes a control policy based on an access control list (ACL), which is a list of filtering rules for packet forwarding. For the purpose of strengthening integrity and authentication of packet payloads during the transmission, the T-NAC system extends the function of ACL and adds security encryption mechanism and packet authentication to eliminate security threats, such as IP forgery and packet modification.

TABLE II. CONTROL RULE OF ACL

IP Address	Unique ID of AR	Payload Encrypted Key (PEK)	Access Policy

We design an ACL to store basic information including IP address, Unique ID of AR, Payload Encrypted Key and enforced access policy of local hosts, as described in TABLE II.

The ACL is stored in DRAM of IXP2400 and updated by PDP. An Enforcement policy contains the privilege description

[2] Payload Encrypted Key is used for PEP to decrypt packets' payload, which will be discussed in the next part.

Authorized licensed use limited to: Tsinghua University Library. Downloaded on October 7, 2008 at 8:45 from IEEE Xplore. Restrictions apply.

of access, including four types: complete access (default), partial access, restricted access and forbidden access. Payload Encrypted Key is different from Session Key; Session Key is used by AR and PDP in process of integrity authentication while Payload Encrypted Key is stored in ACL and encrypted/decrypted during Policy Enforcement between hosts and PEP. The IP address and Data Transmit Key have one to one correspondence, but the same IP can correspond to several ID of AR process. Through the ACL, the PEP could enforce different control policies according to different local hosts where the packets flow is generated from.

When AR has passed authentication and has been authorized for access, Server will send the access policy to the PEP and update the ACL shared with the PEP, then PEP will authenticate and forward packets by enforcing control policies. For the purpose of securing transmission, AR will add its ID into the packet payload, and use AES encryption on the payload with PEK. Due to the confidentiality of ID and TK, it enhances the difficulty in any IP forgery and payload modification.

B. Design of PEP

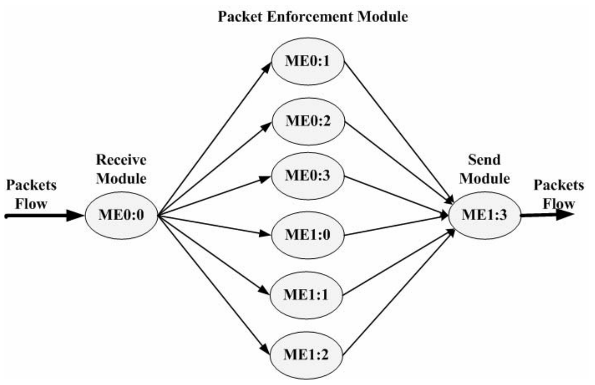

Figure 6. MicroEngine Dispatch Solution of PEP

PEP is the policy enforcement entity of the T-NAC system, based on the IXP2400 Architecture; we implement PDP in the XScale core and PEP in MicroEngines. The MicroEngines execute the PEP code module downloaded from the XScale core. The main function of PEP has two parts: one is as data stream path, which stores and forwards the packets; while the other is a policy enforcement module for the packets. The allocation of MicroEngines is shown as Figure 6.

To enforce access policies and secure communication, each ME will inspect packets to make sure they have not been forged or modified through the security mechanism and secret algorithms. ME seeks the ACL by source IP of packets to get source ID of AR and Payload Encrypted Key, and does AES decryption on packet payloads with the Payload Encrypted Key to get the unique ID, and a ME confirms the packet integrity and validity by comparing the ID in the ACL and the ID decrypted in payload. Finally, a ME enforces the corresponding access policy and forwards packets.

IV. T-NAC System Implementation

A. AR entity

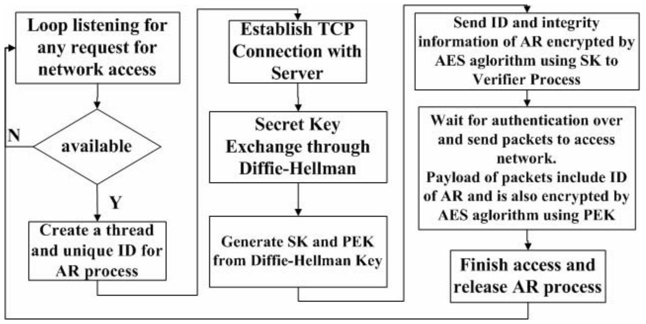
Loop listening for any request for network access
available
N
Y
Create a thread and unique ID for AR process
Establish TCP Connection with Server
Secret Key Exchange through Diffie-Hellman
Generate SK and PEK from Diffie-Hellman Key
Send ID and integrity information of AR encrypted by AES aglorithm using SK to Verifier Process
Wait for authentication over and send packets to access network. Payload of packets include ID of AR and is also encrypted by AES aglorithm using PEK
Finish access and release AR process

Figure 7. Flowchart of AR procedure

Functions of the AR entity include: establish a connection with the server, generating a Session Key and Payload Encrypted Key, collecting and transmitting integrity information. Integrity Information of the AR includes information such as firewall version, anti-virus database update time, OS patch version etc. The program flow chart is shown as Figure 7.

B. PDP entity

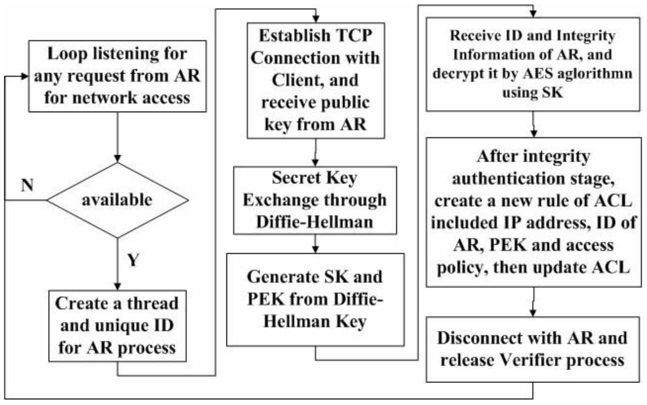
Loop listening for any request from AR for network access
available
N
Y
Create a thread and unique ID for AR process
Establish TCP Connection with Client, and receive public key from AR
Secret Key Exchange through Diffie-Hellman
Generate SK and PEK from Diffie-Hellman Key
Receive ID and Integrity Information of AR, and decrypt it by AES aglorithmn using SK
After integrity authentication stage, create a new rule of ACL included IP address, ID of AR, PEK and access policy, then update ACL
Disconnect with AR and release Verifier process

Figure 8. Flowchart of PDP procedure

PDP flow chart is illustrated in Figure 8. We implemented the PDP entity program much more easily compared to AR. During the connection and communication between AR and PDP, we defined our own protocols, including a handshake protocol to finish Diffie-Hellman key exchange process. PDP's another important function is to maintain and update one Access Control List stored in shared memory between Xscale and MicroEngines, after integrity authentication stage, PDP create or update one new control rule of ACL according to TABLE II.

Authorized licensed use limited to: Tsinghua University Library. Downloaded on October 7, 2008 at 8:45 from IEEE Xplore. Restrictions apply.

C. PEP entity

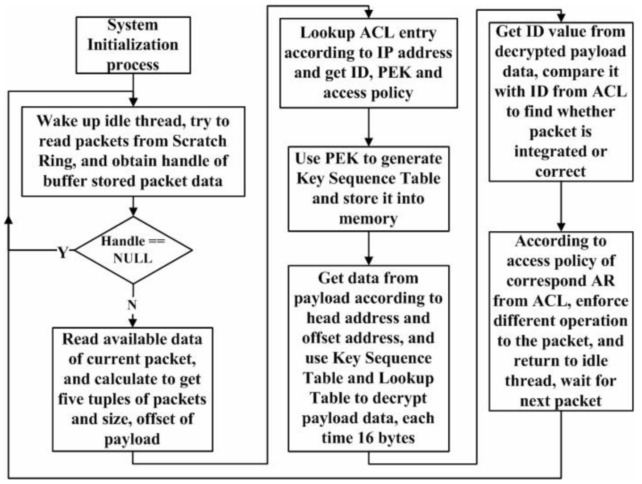
System Initialization process
Wake up idle thread, try to read packets from Scratch Ring, and obtain handle of buffer stored packet data
Handle == NULL
Y
N
Read available data of current packet, and calculate to get five tuples of packets and size, offset of payload
Lookup ACL entry according to IP address and get ID, PEK and access policy
Use PEK to generate Key Sequence Table and store it into memory
Get data from payload according to head address and offset address, and use Key Sequence Table and Lookup Table to decrypt payload data, each time 16 bytes
Get ID value from decrypted payload data, compare it with ID from ACL to find whether packet is integrated or correct
According to access policy of correspond AR from ACL, enforce different operation to the packet, and return to idle thread, wait for next packet

Figure 9. Flowchart of PEP procedure

We populated the microcode to implement the PEP entity on the MicroEngines of IXP2400, especially implemented the AES encryption algorithm using open source code from OpenSSL for reference. All the operations such as metathesis, arrangement and recurrence could be replaced by logic operations on the data from the prestored fixed datasheet, which will be much more convenient for the implementation.

As shown in Figure 9. System initialization needs the Foreign Module of the Intel Workbench to pre-store data of the Lookup Table into specified memory, which is used for AES decryption. We decrypted payload data using a Payload Encrypted Key to get the ID value, and compared it with the ID from the ACL to check packets' authenticity and integrality. If they are not matched, it means the contents of the packets have been modified or the IP and Payload Encrypted Key are forged. If above check is passed, PEP will enforce the policy from the corresponding entry of ACL, if it allows complete access privileges, all the packets will be passed; if it allows partial access privileges, only packets whose destination address is specified could be passed, others will be denied.

V. PERFORMANCE EVALUATION AND OPTIMIZATION

A. Emulation Performance Result

PDP entity runs on the ENP2611 platform of IXP2400 network processor provided by Intel with the VxWorks OS on the XScale core; PEP is coded by a micro instruction module running on MicroEngines, and the Emulation tool is Developer Workbench 4.1 provided by Intel Company. We attempt to boost the PEP's throughput and ME's utilization.

In the initial scheme of PEP design, the Lookup Table for the AES decryption algorithm is stored in Scratchpad memory, the Key Sequence Table is stored in SRAM, and both of them are external memory of the IXP2400. The frequency of XScale and MicroEngines are 900MHz. Network flow used for emulation is an Ethernet IP packet stream with 32 bytes of payload.

Here are the emulation results for Scheme I in TABLE III.

TABLE III. EMULATION RESULTS OF SCHEME I

Receive Rate (Mbps)	233.5
Transfer Rate (Mbps)	194.1
Utilization Rate of ME	60.625%
Clock Cycles	2212102

B. Performance Optimization

As shown in Table III, the throughput and utilization rate are quite low while the packet stream generating speed is 1Gbps. To optimize the performance, we track for the system bottlenecks under a highly parallel environment, while memory sharing and prolonged access latency concentrate in PEP; it is surely the focalization for our system optimization.

In the Packet Enforcement Module, there are not many computing operations caused by adopting the Lookup Table for the AES decryption algorithm. Unfortunately it brings too many read/write access operations to external memory, which costs too many clock cycles. Too many IO operations increase the time for resource competition by BUS and mutex delays for multi-threads. All of these are the primary reason for low performance of the design Scheme I.

Ciphers need memory to store their control data, i.e. subkeys and sequence tables, so reducing IO operations of accessing external memory is an effective way to optimizing performance. We proposed the Scheme II to put the Key Sequence Table and part of the AES Lookup Table into Local Memory.

As the capacity of Local Memory for each MicroEngine is very limited, we analyze access times of the Lookup Table for the AES decryption algorithm. Access times for six Lookup Table operations, including Td0~Td4 and Te4 are shown in the TABLE IV. (n means length of decrypt data).

TABLE IV. ACCESS TIMES OF LOOKUP TABLE

Lookup Table	Times	n = 16	n = 32	n = 48
Td0	40+40*(n/16)	80	120	160
Td1	40+40*(n/16)	80	120	160
Td2	40+40*(n/16)	80	120	160
Td3	40+40*(n/16)	80	120	160
Td4	16*(n/16)	16	32	48
Te4	160	160	160	160

From TABLE IV. for 16 bytes payload data, Te4 has the most access times, while access times of Td0~Td1 increase much faster with increases in payload data for decryption. So putting the table with the most access times into Local Memory will increase efficiency of IO access and enhance performance dramatically.

Here are the results for Scheme II in TABLE V.

TABLE V. EMULATION RESULTS OF SCHEME II

Receive Rate (Mbps)	255.53
Transfer Rate (Mbps)	209.35
Utilization Rate of ME	81.217%
Clock Cycles	1930189

Authorized licensed use limited to: Tsinghua University Library. Downloaded on October 7, 2008 at 8:45 from IEEE Xplore. Restrictions apply.

From TABLE V. Scheme II obviously has much better performance than Scheme I. Total time for each packet enforcement is reduced by nearly 50%, and in Scheme II, receive rate improves 9.44%, while send rate improves 7.86%. The utilization rate improves 33.97% which means the time that MicroEngines wait for external bus resources are reduced greatly and utilization rate is obviously enhanced.

There are two reasons why throughput could not be enhanced as expected, one is the competition for IO resources between multiple threads is still not mitigated while the other is limited for inner memory and system design, what we call it memory-wall effect on throughput. In the emulation of our former schemes, we adopt 32 bytes for payload data length, so it is reasonable to explain limited performance enhance by exchange ciphers control data out of shared memory into Local Memory. If there is enough room in Local Memory for Td0~Td3 or the dynamic cache algorithm for the Lookup Table, the performance of our design will be much further boosted.

Besides the method of reducing IO operations to external memory，another optimizing strategy can be made in cache cipher control data and reducing reduplicate calculates for encryption and decryption process. During AES decryption on packet payload by PEP, we cached the Key Sequence Table of IP addresses whose frequency of sending packets is very high, as we reasonably suppose to be belong to the some packet flow, then we could avoid repeated computing of the Key Sequence Table for each packet and reduce enforcement time of PEP greatly, which will surely boost throughout of PEP.

Here are the test results for Scheme III in TABLE VI. Compared with Scheme I, Rx rate improved 26.78% while Tx rate improved 25.57%.

TABLE VI. EMULATION RESULT OF SCHEME III

Receive Rate (Mbps)	296.02
Transfer Rate (Mbps)	243.73
Utilization Rate of ME	82.188%
Clock Cycles	1664891

The three schemes' performance comparisons are shown in Figure 10.

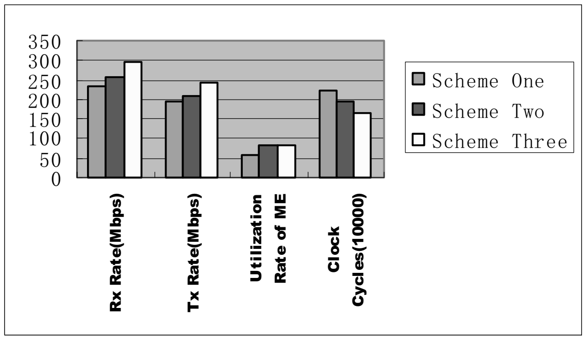

Figure 10. Emulation Performance of Three Schemes

In a word, Scheme III has the best performance with the main idea of exchanging space complexity for optimized time complexity, through improvement of the utilization rate of Local Memory and caching some data for packets to improve throughout and efficiency. We believe there still is room for performance improvement by analyzing packets distribution and optimizing memory allocation.

C. *System Performance Evaluation*

The prototype system platform mainly includes design of the AR, PDP and PEP entities. The IXP2400 Network Processor is the hardware platform for the implementation of both PDP and PEP entities by fully using XScale, MicroEngines and shared external memory, actting as an authenticate server as well as a gateway with policy enforcement for access control. We still leave some functions of TNC Specification for future development, such as the components in the Remediation Layer and Trust Layer.

The emulation environment is High-speed Ethernet LAN, which is usually simulated as an intranet environment with not many hosts; the internal hosts are able to access the external network only when authenticated by switch or gateway. The hurdles between AR and PDP's connection are not performance throughput and time delay, but trustworthiness and reliability of communication.

As network processor's shared memory hierarchy increases difficulties in providing a balance between performance and flexibility while designing and programming, we attempt to design a more efficient memory allocation scheme to get higher throughput and efficiency for PEP, which is the data path and gateway. The main bottleneck lies on time cost by external memory access and delay of bus competition between multiple threads. To alleviate this memory-wall effect, the optimization methods on cipher algorithms and authentication process and the proper memory allocation scheme were adopted. Finally, we boost the throughput of PEP more then 25% of the prime scheme and obtained nearly 300Mbps which is reasonable but not high as expected, partially due to the lack of a special crypto unit in the IXP2400 which restricts performance.

VI. CONCLUSION

Trusted Network Connect is not only a new conceptual architecture for network security, but a research hotspot for network security and an industrial specification supported and promoted by many networking industry companies. In this paper, based on Intel IXP2400 network processor platform, we built the novel TNC-compatible NAC system framework to implement network access control mechanisms which ensure network administrators enforce security policies on endpoint connections and communication with the corporate networks depending on the integrity and security status of the endpoints.

The T-NAC system framework is compatible to the TNC Specification proposed by TCG and flexible to extend network access control mechanism, and it stresses platform-authentication and secure communication further more through highly secure ciphers algorithm and security communication mechanism. To alleviate the memory hierarchy bottleneck and enhance system performance, we made optimization on cryptographic performance by adopting more effective memory allocation and utilization.

Authorized licensed use limited to: Tsinghua University Library. Downloaded on October 7, 2008 at 8:45 from IEEE Xplore. Restrictions apply.

In the future, we plan to use IXP2850 instead of IXP2400 to implement a more integrated architecture and platform and expect better performance, as Intel IXP2850 network processor consisting of independent crypto unit for cryptographic module. And another research plan is to use TPM module in endpoint to improve the trustworthy behavior of platform and trust verification. And the application of the prototype system in a wireless-LAN environment will be also attempted.

ACKNOWLEDGMENT

This research is supported by Intel IXA University Program (No. 0411A66), the National Natural Science Foundations of China (No. 90412012, No. 60673187) and the National Grand Fundamental Research 973 Program of China (No. 2006CB708301)

REFERENCE

[1] Shinichi Suzuki, Yasushi Shinjo, Toshio Hirotsu, “Capability-based Egress Network Access Control for Transferring Access Rights”, Proceedings of the Third International Conference on Information Technology and Applications (ICITA’05), 2005.

[2] Gabriel López, Antonio F. Gómez, Rafael Marín, Oscar Cánovas, “A Network Access Control Approach based on the AAA Architecture and Authorization Attributes”, Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS’05), 2005.

[3] Trusted Computing Group. TCG Trusted Network Connect TNC Architecture for Interoperability Specification Version 1.1. TCG published, May, 2006. pp 7~35.

[4] IEEE802, “Port-Based Network Access Control”, IEEE Std 802.1X-2001.

[5] Trusted Computing Group, IWG Reference Architecture for Interoperability (Part 1), Specification Version 1.0, June 2005.

[6] Douglas E. Comer, “Network System Design: Using Network Processors,” Pearson Education Inc., 2004.

[7] Wajdi Feghali, Brad Burres, Gilbert Wolrich, Douglas Carrigan, “Security: Adding Protection to the Network via the Network Processor,” Intel Technology Journal, Volume 06, Issue 03, Published August 15, 2002.

[8] Intel® IXP2400 and IXP2800 Network Processor Programmer’s Reference Manual, Intel Press, July 2005.

[9] Joan Daemen, Vincent Rijmen. Advanced Encrypt Standard Arithmetic (Translated by: Gu Dawu, Xu Shengbo). Beijing: Tsinghua University Press, 2003.

[10] Wenbo Mao. Modern Cryptography: Theory and Practice (Translated by Wang Jilin, Wu Qianhong etc.). Beijing: Electronics Industry Press, 2004.

[11] Douglas E.Comer. Network System Design: Using Network Processors. Pearson Education Inc., 2004.

[12] Erik J. Johnson, Aaron R. Kunze. IXP2400/2800 Programming : The Complete Microengine Coding Guide. Intel Press, 2003.

[13] Ram Bhamidipati, Ahmad Zaidi, Siva Makineni, Kah K. Low, Robert Chen, Kin-Yip Liu, Jack Dahlgren, "Challenges and Methodologies for Implementing High-Performance Network Processors," Intel Technology Journal, Volume 06, Issue 03, Published August 15, 2002.

[14] Sridhar Lakshmanamurthy, Kin-Yip Liu, Yim Pun, Larry Huston, Uday Naik, "Network Processor Performance Analysis Methodology," Intel Technology Journal, Volume 06, Issue 03, Published August 15, 2002.

[15] Uday Naik, Alex Shoykhet, Larry Huston, Donald Hooper, Raj Yavatkar, Duke Tallam, Travis Schluessler, Prashant Chandra, Adrian Georgescu, "IXA Portability Framework: Preserving Software Investment in Network Processor Applications," Intel Technology Journal, Volume 06, Issue 03, Published August 15, 2002.

[16] Bill Carlson, Intel Internet Exchange Architecture and Applications: A Practical Guide to Intel Network Processors, Intel Press, 2003.

[17] Erik J. Johnson and Aaron R. Kunze, IXP2400/2800 Programming: The Complete Microengine Coding Guide, Intel Press, 2003.

[18] Chen Zhiyu, Wen Yanjun, Chen Qi. VxWorks Program Develop Practice. Beijing: People Posts & Telecom Press, 2004.

[19] Joan Daemen, Vincent Rijmen. Advanced Encrypt Standard Arithmetic (Translated by: Gu Dawu, Xu Shengbo). Beijing: Tsinghua University Press, 2003.

[20] Feng Dengguo, Cai Jiren. Network Security and Cryptology. Guiyang: Guizhou Sceience and Technology Press, 2004.

Authorized licensed use limited to: Tsinghua University Library. Downloaded on October 7, 2008 at 8:45 from IEEE Xplore. Restrictions apply.