End-to-End Security Methods for UDT Data Transmissions

Danilo Valeros Bernardo and Doan B. Hoang

iNext, Computing and Communications
Faculty of Engineering and Information Technology
The University of Technology Sydney, Sydney
Australia
bernardan@gmail.com, dhoang@it.uts.edu.au

Abstract. UDT (UDP-based data transfer protocol) is one of the most promising network protocols developed for high data speed data transfer. It does not, however, have any inherent security mechanisms, and thus relies on other transport protocols to provide them. Towards its implementation in high speed networks, security and privacy are critical factors and important challenges that need to be addressed. There were substantial research efforts we carried out so far to address these challenges. We introduced security mechanisms through the application layer using UDT's API and presented DTLS, GSS-API, and CGA, in transport and IP layers. In this paper, we make the following contributions: we out line security requirements for UDT implementation and propose practical encryption methods for securing UDT within the network layer.

Keywords: UDT, TCP, GSS-API, DTLS, Next Generation Protocol.

1 Introduction

Developments in 2007 introduced UDT, the next generation of high performance data transfer protocol for cloud computing [3], [4]. One compelling example of the implementations of UDT is the Sloan Digital Sky Survey (SDSS) project [3-6], which is mapping in detail one quarter of the entire sky, determining the positions and brightness of more than 300 million celestial objects. It measures the distances to more than a million galaxies and quasars. The data from the SDSS project so far has increased to 2 terabytes and continues to grow. Currently, the 2 terabytes data is being delivered to the Asia-Pacific region, including Australia, Japan, South Korea, and China. Astronomers also want to execute online analysis on multiple datasets stored in geographically distributed locations [3-6].

This implementation offers a promising direction for future high speed data transfer in various industries.

The absence of a well-thought security mechanism for UDT when it was developed, however, drives this paper to introduce ways to secure UDT in various environment and implementation scenarios.

In the following sections, we briefly introduce UDT. We then present the objectives of this paper; introduce and discuss the methods in securing UDT and present the results of experiments, and present the conclusion of this paper.

1.1 Background

UDT introduces a new three-layer protocol architecture that is composed of a connection flow multiplexer, enhanced congestion control and resource management. The new design allows protocol to be shared by parallel connections and to be used by future connections. It improves congestion control and reduces connection set up time. UDT provides better usability by supporting a variety of network environments and application scenarios [6]. It addresses TCP's limitations by reducing the overhead required to send and receive streams of data.

UDT is a connection-oriented duplex protocol, which supports data streaming and partial reliable messaging [3-6], [12]. It also uses rate-based congestion control (rate control) and window-based flow control to regulate outgoing traffic. This was designed such that rate control updates the packet sending period every constant interval, whereas flow control updates the flow window size each time an acknowledgement packet is received. It was expanded to satisfy more requirements for both network research and applications development. This expansion is called Composable UDT and was designed to complement the kernel space network stacks.

The pressure, however, to reduce the cost and complexity of running streaming applications over the Internet and through wireless and mobile devices continues to mount.

Moreover, users demand better security and privacy for their communication links.

Our contention for the need of security mechanisms of the new UDT is derived from 5 important observations [3-6].

- Absence of inherent security mechanism, such checksum for UDT.
- The header information is not sufficient for this protocol.
- Dependencies on user preferences and implementation on the layer on where it is implemented.
- Dependencies on existing security mechanisms of other layers on the stack.
- Dependencies on TCP/UDP which are dependent on nodes and their addresses for high speed data transfer protocol leading to a number of attacks such as neighborhood, Sybil and DoS (Denial of Service) attacks.

Earlier works in the development of security framework for UDT support the need of minimizing its sending rates [3], [6] in retransmissions and introducing its own checksum in its design. The introduction of other security mechanisms, however, to secure UDT is presented to address its vulnerabilities against adversaries exploiting the application, transport, and IP layers.

We [3-6] presented an overview on securing UDT implementations in various layers. However securing UDT in application and other layers need to be explored in future UDT deployments in various applications.

There are application and transport layer based authentication and end-to-end [7] security options for UDT.

1.2 Motivations

In this paper, there are 2 important objectives we intend to address:

Firstly, reviews on existing methods of security feasible for UDT transmission, such as:

- Security at the application layer via UDT extensions may require client and servers, and significant changes on applications to accommodate security features.
- Encryption be performed at the layer 3 (Network Layer), abstracted from the UDT application, e.g., via gateway-to-gateway, virtual private networks (VPNs), when security solution on the application layer becomes too complex to develop.

Secondly, where sessions are point-to-point we propose the use of host-to-host encryption, or gateway-to-gateway encryption at the border of each host's network. Utilizing encryption security, where the absence of a viable security mechanism of a particular new protocol such as UDT is a feasible option.

We looked at existing encryption support for UDP and TCP, and determined no encryption methods were proposed and available for UDT. However, our assertion is that encryption can support UDT through the network layer when it is running on top of UDP.

Existing security mechanisms for UDP developed at some layers are certainly not advanced and flexible to operate with UDT, e.g., UDP: UDT+DTLS and UDT+GSS-API, UDT+MD5, SHA-1 or 256 options as proposed earlier in our works [6]. The progression of these mechanisms require significant changes on the UDT's design structure, such as accommodating large APIs and functions for the development security mechanisms; and introducing an option for hash functions to integrate in its header to secure network gateway connections. We present in this paper that existing encryption methods particularly on the network layer used for other protocols can simply be implemented on UDT and UDP while the development of proprietary mechanisms is underway.

2 Encryption Methods

We describe 2 encryption methods not specific to UDT in which it can operate. These methods can be used to secure communications between networks, for any application using different protocols.

First, host-to-host method. In this method the encryption is set up between the two hosts wishing to exchange secure data. This has the advantage that encryption is applied end-to-end along with the full path of the connection, but it also implies the user may have to do something to initiate the encryption.

Second, gateway-to-gateway method, which encryption is applied between boundary routers at the edge of a network. This means that sessions are encrypted end-to-end, and thus are not vulnerable to snooping within the endpoint networks.

In the experiments presented in section 3, we report on tests performed on one method (gateway-to-gateway) rather than Host-to-Host method when UDT is implemented on top of UDP.

Host-to-Host is simplistic and can be attained by implementing Windows built in IPsec. This method is attainable through implementing existing OS and applications that provides end-to-end security. Subsequently, our focus on gateway-to-gateway method is relevant due to the behavior of UDT in data transmission where it performs significantly long distance high bandwidth networks (via the Internet) which most of them are behind gateways.

2.1 Internet Protocol Security (IPsec)

IPsec offers the promise of protecting against many of denial-of-service attacks. It also offers other potential benefits. Conventional software-based IPsec implementations isolate applications from developing proprietary cryptographic keys, improving security by making inadvertent or malicious key exposure more difficult. In addition, specialized hardware may allow encryption keys protected from disclosure within trusted cryptographic units. Also, custom hardware units may well allow for higher performance [6].

Implementing UDT running with IPsec provides adequate protection for data transmission (fig. 1). A datagram-oriented client application using UDT will use the connection-oriented part of its API (because it is using a given datagram socket to talk to a specific server) while the server it is talking to use the connection oriented API because it is using a single socket to receive requests from and send replies to a large number of clients.

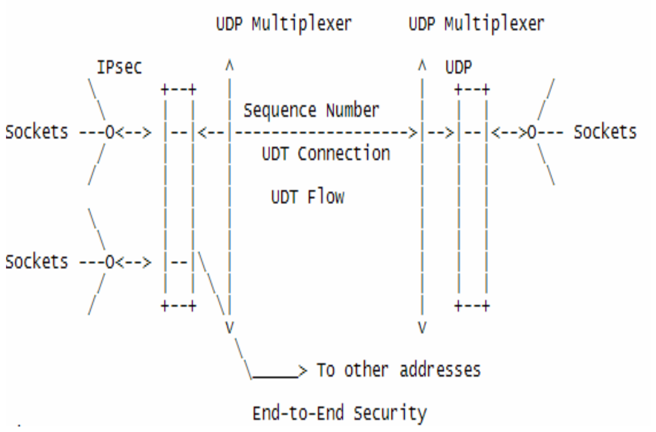

Fig. 1. UDT flow using end-to-end security. [8-11]. IPsec can be used without modifying UDT and its applications running it.

2.2 IPsec Tools (Gateway-to-Gateway Method)

There are a number of commercial IPsec VPN solutions on the market, of which Netscreen is a typical example. The key management scheme is IKE, Sha, Key Group Diffie-Hellman Group 2, Encryption algorithm is Triple DES.

Configuration on the Netscreen firewall for keys and a client software needs to be configured on the host machine. However, since we are looking at gateway-to gateway encryption, we use the site-to-site VPN configuration.

The product also provides secure remote access to a corporate network, client- to-site (host –to-gateway), and is able to scale up to multiple remote sites.

We also consider a downloadable IPsec software for implementation running on Linux platform.

This software is Free Secure Wide Area Network (S/WAN) or FreeS/WAN that can be used with PGP and X .509 certificates. It uses IKE for key exchange. It has an important feature called “opportunistic encryption” such that any two FreeS/WAN gateways will encrypt data when traffic is observed flowing between them.

This software, however, has ceased for further developments due to export and legal restrictions in the United States. However, its last version remains to be free and suitable for non-commercial use, such as research. We selected this tool because of its flexibility and its interesting feature of opportunistic encryption. It also has very low overhead compared to using other software.

We tested UDT on both tools. In section 3, we present the results of our experiments using Netscreen VPN and FreeS/WAN. A simple application running UDT was downloaded [3] and installed on Windows OS hosts behind the secure gateways.

3 Encryption Experiments

We describe and conduct experiments on gateway-to-gateway encryption method under two environments.

In our tests, we evaluated UDT performance using FreeS/WAN in gateway-to-gateway mode. We also tested UDT using a commercial product called Netscreen in site-to-site mode. For FreeS/WAN, the test network composed of two gateway hosts. For Netscreen, we used two firewalls at the gateways and configured them for site-to-site VPN. The client PCs used for the test were a 32GB 2.3GHz laptop running Windows XP Service Pack 2, with an application running UDT and they were behind the gateways.

3.1 FreeS/WAN (Gateway-to-Gateway)

We summarize the tests performed on FreeS/WAN. We carried out tests on FreeS/WAN in tunnel mode (gateway-to-gateway) rather than in transport mode (i.e. host-to-host).

The default algorithms used by FreeS/WAN are 3DES and MD5, with RSA authentication keys. These are defined in the */etc/ipsec.conf* file. Private and other key information is stored in */etc/ipsec.secrets*.

3.2 Netscreen IPsec VPN (Site-to-Site)

We configured the firewalls site-to-site VPN and allowed traffic from a selected gateway private network. We created route-based VPNs and bind their configuration to a virtual interface called tunnel interface, with fixed IP addresses. We assigned address for the tunnel interface, with both firewalls that make up the tunnel using route-based, in the same subnetwork. We created the IKE gateway for Phase-1 which the same for route and tunnel interface and created autokey IKE and bind it to the interface. We then configured the routing to the remote network to the outbound interface.

3.3 Measurement Schemes

To test the performance, we repeated ping and ftp tests using existing performance measurement tools. For ping and throughput measurements we used Qcheck/Statscout, and MS' pathping. We also used WS_FTP server (running on the laptop device), see results on tables 3 and 4.

Qcheck/Statcout allows network performance measurements to be taken between any two endpoints, with control from a remote console program. Qcheck results were as follows, averaging over 10 measurements see tables 1 and 2):

Table 1. Results of Qcheck, with and without encryption

Netscreen (10repeats)

Measurement	Average Plain	Average with Encryption
TCP throughput	200.9 ms	180.4 ms
UDP throughput	140.2 ms	130.3 ms
UDT throughput	9000 ms	6000 ms
TCP response time	1/1/1 ms min/avg/max	.8/1/1.2 ms min/avg/max
UDP response time	1/1/1 ms min/avg/max	1/1/1 ms min/avg/max
UDT response time	1/1/1 ms min/avg/max	.9/.95/1 ms min/avg/max

Table 2. Results of Qcheck, with and without encryption

FreeS/WAN (10repeats)

Measurement	Average Plain	Average with Encryption
TCP throughput	187 ms	162 ms
UDP throughput	120.1 ms	113 ms
UDT throughput	7000 ms	4000 ms
TCP response time	1/1/1 ms min/avg/max	1.2/2.1/3 ms min/avg/max
UDP response time	1/1/1 ms min/avg/max	1.2/2.1/3 ms min/avg/max
UDT response time	1/1/1 ms min/avg/max	1.2/2.1/3 ms min/avg/max

These results do not suggest there would be problems encrypting the sessions of UDP and UDT.

Table 3. FTP test results with (encrypted data) and without (or plain data) encryption

FTP (Netscreen)

FTP Passive Mode	Time (sec)	Plain Mbytes	Time (sec)	Encrypted Mbyte/s	Repeat
Laptop-gateway-gateway-laptop	2.3	1.2	1	0.6	1
	2	1.3	1	0.8	2
	1.8	1	1	0.6	3
	2.2	1.2	1	0.7	4
	2.4	1.2	2	0.8	5
	1.9	1.05	1	0.5	6
	2	1.3	1	0.5	7
	2.6	1.4	2	1	8
	2	1.4	1	0.8	9
	1.8	1.2	1	0.9	10

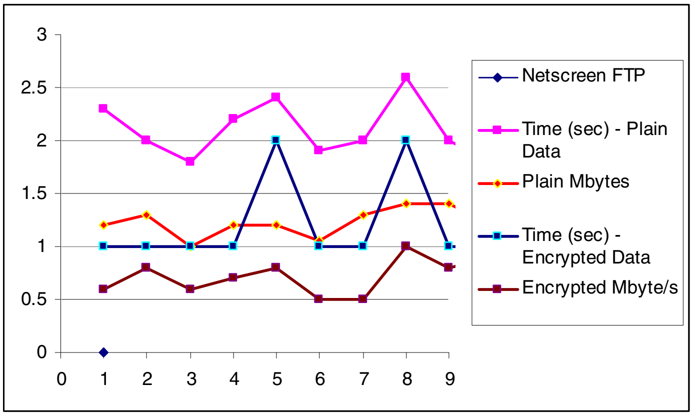

Fig. 2. Netscreen Graph results

The encryption on our test network was performed on the higher specification gateways and firewalls.

The resulting performance is acceptable with encryption. The significant result is that the encrypted FTP throughput is seen to be acceptable with 90% of the non-encrypted throughput (see tables 3 and 4, figs 2 and 3). This reflects the use of high speed bandwidth.

Finally, we performed *ping* tests (with WS Ping Pro and pathping) between the two gateways with differing size *ping* packets, with the response time in milliseconds being measured. We also ran the ping tests within the interfaces of the firewalls.

Table 4. FTP test results with (encrypted data) and without (or plain data) encryption

FTP (FreeS/WAN)

FTP Passive Mode	**Time (sec)**	**Plain Mbyte/s**	**Time (sec)**	**Encrypted Mbyte/s**	**Repeat**
Laptop-gateway-gateway-laptop	2.1	1.2	2	0.6	1
	1.6	1.3	2	0.8	2
	1.2	1	2	0.6	3
	1.8	1.2	1	0.7	4
	2	1.2	1	0.8	5
	2.2	1.05	1	0.5	6
	1.8	1.3	1	0.5	7
	2.1	1.4	2	1	8
	1	1.4	1	0.8	9
	1.8	1.2	1	0.9	10

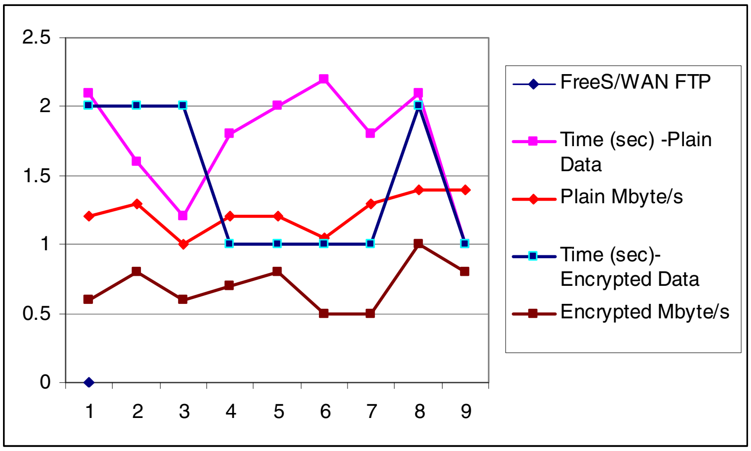

Fig. 3. FreeS/WAN Graph results

Table 5. Ping test results, with and without encryption

Netscreen (Ping test)

Gateway	**Mode**	**1000 bytes**	**2000 bytes**	**10000 bytes**
Laptop-gateway-gateway laptop	Clear	1000	1988	9800
	IPsec	1000	1990	10000

Table 6. Ping test results, with and without encryption

FreeS/WAN (Ping test)

Gateway	**Mode**	**1000 bytes**	**2000 bytes**	**10000 bytes**
Laptop-gateway-gateway laptop	Clear	1000	1900	9600
	IPsec	1000	1800	10000

The raw response times are poorer (see tables 5 and 6) for our non-encrypted (clear) test at the default *ping* packet size, but we presume the intervening gateways are responsible for that effect. Otherwise, the results are significantly improved, especially for the much larger packet sizes where fragmentation is occurring.

Overall our results illustrate the improved performance that using higher specification encrypting devices with higher bandwidth links can bring. The various throughput and ping tests we performed suggest that at the bandwidth levels required fast data transfer, Netscreen would appear to offer a solution that doesn't impact significantly on latency or session quality.

4 Impact on Performance

Our experience with trials of both Netscreen and FreeS/WAN implies that encryption at the network layer does not impose significant performance problems. Perceived latency was very similar in our tests, and the empirical results imply the overhead at the network layer is in the order of a handful of milliseconds.

It should be noted that in addition to the encryption overhead on the CPU, the encrypted packets will also be larger due to the additional AH/ESP data being sent, and the packet re-assembly at the far end will take longer, i.e. there are delays in passing encrypted data beyond just the raw computational burden [7].

In terms of algorithms, one might assume that weaker algorithms are less computationally expensive. However, existing encryption algorithms such as AES, can offer better encryption in comparison to 3DES for less processor effort (an important consideration when encryption is required on miniature smart type devices). Coupled with advances in processor and bandwidth speed, the latency penalty for encryption will continue to fall as a percentage of the time and bandwidth required for high data transfer. The same, of course, may not be true for UDT implementations over a low bandwidth network such as cellular wireless (where the packet size overhead is far more significant). AES also has the advantage of being an open standard [7].

5 Conclusion

There are a number of conclusions that we can draw from the above comparisons and experiments. Since UDT is very new, there is very little adoption of this new protocol, consequently no security mechanisms available for the application layer. We initially consider host-to-host encryption as a feasible solution, depending on the operating system and method desired. However, it is likely to require some expertise in the end user of the end host's system, and will cause some problems for firewalls, because stateful inspection for UDP in which UDT runs on cannot be performed on an encrypted session [7].

We assert that gateway-to-gateway encryption would appear to offer a flexible and relatively efficient means to encrypt UDT and UDP data over the public (Internet) part of a session connection, but is vulnerable to snooping on the internal site network behind the gateway, unless appropriate security solutions are also put in place.

We observed that latency effects of encryption do not appear to be significant based on limited tests performed with FreeS/WAN and Netscreen on entry-level high speed gateway devices. Increasing commodity CPU power is making encryption ever more viable for reasonable UDT data transfer. Opportunistic encryption is desirable. FreeS/WAN includes support for this, but we have been unable to test it in details – the scaling issues may be significant and should be tested further if FreeS/WAN is to be considered for UDT wider deployment. Gateway encryption via a product like FreeS/WAN for smaller and less bandwidth and Netscreen for higher bandwidth and bigger environments may be effective methods at present. Such products and their configurations however, need wider-scale testing prior to their potential use. End-to-end encryption through gateway-to-gateway encryption, therefore offers security for UDT as it is to other protocols.

References

[1] Bellovin, S.: Defending Against Sequence Number Attacks. RFC 1948 (1996)

[2] Bellovin, S.: Guidelines for Mandating the Use of IPsec, Work in Progress, IETF (October 2003)

[3] Bernardo, D.V., Hoang, D.B.: A Conceptual Approach against Next Generation Security Threats: Securing a High Speed Network Protocol – UDT. In: Proc. IEEE the 2nd ICFN 2010, Shanya, China (2010)

[4] Bernardo, D.V., Hoang, D.B.: Security Requirements for UDT, IETF Internet-Draft – working paper (September 2009)

[5] Bernardo, D.V., Hoang, D.B.: "Network Security Considerations for a New Generation Protocol UDT. In: Proc. IEEE the 2nd ICCIST Conference 2009, Beijing, China (2009)

[6] Bernardo, D.V., Hoang, D.B.: A Security Framework and its Implementation in Fast Data Transfer Next Generation Protocol UDT. Journal of Information Assurance and Security 4(354-360) (2009) ISN 1554-1010

[7] Chown, T., Juby, B.: Overview of Methods for Encryption of H.323 Data Streams. Technical Paper, University of Southampton (March 2001)

[8] Blumenthal, M., Clark, D.: Rethinking the Design of the Internet: End-to-End Argument vs. the Brave New World. In: Proc. ACM Trans Internet Technology, vol. 1 (August 2001)

[9] Clark, D., Sollins, L., Wroclwski, J., Katabi, D., Kulik, J., Yang, X.: New Arch: Future Generation Internet Architecture, Technical Report, DoD – ITO (2003)

[10] Falby, N., Fulp, J., Clark, P., Cote, R., Irvine, C., Dinolt, G., Levin, T., Rose, M., Shifflett, D.: Information assurance capacity building: A case study. In: Proc. 2004 IEEE Workshop on Information Assurance, pp. 31–36. U.S. Military Academy (June 2004)

[11] Gorodetsky, V., Skormin, V., Popyack, L. (eds.): Information Assurance in Computer Networks: Methods, Models, and Architecture for Network Security. St. Petersburg, Springer, Heidelberg (2001)
[12] Gu, Y., Grossman, R.: UDT: UDP-based Data Transfer for High-Speed Wide Area Networks. Computer Networks 51(7) (2007)
[13] Hamill, J., Deckro, R., Kloeber, J.: Evaluating information assurance strategies. Decision Support Systems 39(3), 463–484 (2005)
[14] H.I. for Information Technology, H. U. of Technology, et al.: Infrastructure for HIP (2008)
[15] Harrison, D.: RPI NS2 Graphing and Statistics Package, `http://networks.ecse.rpi.edu/~harrisod/graph.html`
[16] Jokela, P., Moskowitz, R., Nikander, P.: Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP). RFC 5202, IETF (April 2008)
[17] Kent, S., Atkinson, R.: Security Architecture for the Internet Protocol. RFC 2401 (1998)
[18] Leon-Garcia, A., Widjaja, I.: Communication Networks. McGraw Hill, New York (2000)
[19] Mathis, M., Mahdavi, J., Floyd, S., Romanow, A.: TCP selective acknowledgment options. IETF RFC 2018 (April 1996)
[20] Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)
[21] NIST SP 800-37, Guide for the Security Certification and Accreditation of Federal Information Systems (May 2004)
[22] NS2, `http://isi.edu/nsna/ns`
[23] PSU Evaluation Methods for Internet Security Technology, EMIST (2004), `http://emist.ist.psu.edu` (visited December 2009)
[24] Rabin, M.: Digitized signatures and public-key functions as intractable as Factorization. MIT/LCS Technical Report, TR-212 (1979)
[25] Rescorla, E., Modadugu, N.: Datagram Transport Layer Security. RFC 4347, IETF (April 2006)
[26] Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signature and public-keycryptosystems. Communication of ACM 21, 120–126 (1978)
[27] Schwartz, M.: Broadband Integrated Networks. Prentice Hall, Englewood Cliffs (1996)
[28] Stewart, R. (ed.): Stream Control Transmission Protocol, RFC 4960 (2007)
[29] Stoica, I., Adkins, D., Zhuang, S., Shenker, S., Surana, S.: Internet Indirection Infrastructure. In: Proc. ACM SIGCOMM 2002 (2002)
[30] Szalay, A., Gray, J., Thakar, A., Kuntz, P., Malik, T., Raddick, J., Stoughton, C., Vandenberg, J.: The SDSS SkyServer - Public access to the Sloan digital sky server data. In: ACM SIGMOD 2002 (2002)
[31] Wang, G., Xia, Y.: An NS2 TCP Evaluation Tool, `http://labs.nec.com.cn/tcpeval.html`
[32] Globus XIO: http://unix.globus.org/toolkit/docs/3.2/xio/index.html (retrieved on November 1, 2009)
[33] Zhang, M., Karp, B., Floyd, S., Peterson, L.: RR-TCP: A reordering-robust TCP with DSACK. In: Proc. the Eleventh IEEE International Conference on Networking Protocols (ICNP 2003), Atlanta, GA (November 2003)