Adaptive Scheduling of Data Transfer in P2P Applications over Asymmetric Networks

Yuan Pan and Jack Y. B. Lee
Dept of Information Engineering
The Chinese University of Hong Kong
{py007, yblee}@ie.cuhk.edu.hk

Abstract—The success of peer-to-peer (P2P) applications hinges on users' willingness to contribute their network bandwidth to serve other peers. However if the upload data rate to other peers is too high it could severely degrade the download throughput in an asymmetric network such as ADSL, even if the downlink has abundant bandwidth available. Experiments revealed that the download throughput degradation is in fact not caused by congestion in the uplink, but caused by increased queuing delay in the uplink path during high upload data rates. This paper tackles this problem by developing an adaptive algorithm to monitor the uplink queuing delay and adjust the upload data rate limit dynamically so that the download throughput will not be adversely affected. Experiments conducted using an open-source P2P software showed that the proposed algorithms can increase the downlink utilization over a wide range of network configurations (and over 200% increase in some cases) by automatically adjusting the upload data rate limit. The algorithms do not require any user intervention and can be readily incorporated into existing P2P systems.

Key Terms—asymmetric network, P2P, scheduling, TCP.

I. INTRODUCTION

The success of peer-to-peer (P2P) applications hinges on users' willingness to contribute their unused storage and network bandwidth to serve other peers. While broadband residential networks have become commonplace in many countries around the world, many of them offer broad bandwidth only in the downlink, i.e., from ISP to user host. By contrast the uplink bandwidth is typically significantly lower, ranging from tens to hundreds of Kbps as compared to downlink's multi-Mbps bandwidth. This is particularly common in residential networks built on ADSL and xDSL technologies [1].

While this type of bandwidth-asymmetric network does not preclude the use of P2P applications, it nonetheless creates a new problem for such applications when the data are transported over TCP. Specifically, in P2P applications such as BitTorrent many users have observed that if the upload data rate is too high, e.g., close to the uplink bandwidth limit, then the achievable download data rate will be severely degraded.

To illustrate this problem we conducted an experiment as depicted in Fig. 1 and plot the achievable download data rate versus upload data rate for a P2P software called Azureus [2], which is an implementation of BitTorrent. This software allows the user to configure the maximum upload data rate and through this feature we can clearly see its impact on the download data rate, which was drastically reduced from around 900 Kbps down to less than 200 Kbps when the upload data rate limit is increased beyond 120 Kbps (close to the 125 Kbps uplink bandwidth limit).

To remedy this problem many P2P applications (e.g., Azureus, utorrent, BitTorrent, etc.) allow the users to manually configure the upload data rate limit so that a reasonable upload data rate can be used (or else P2P will not work) while keeping the download data rate to fully-utilize the broadband network downlink. Clearly this ad hoc feature is not very user-friendly nor can it adapt to the underlying network properties automatically.

This work investigates this performance problem in running P2P applications over asymmetric networks by first analyzing the problem to identify the mechanics leading to the download data rate degradation, and then develops two versions of adaptive algorithm called Adaptive-DRC to control the upload data rate limit without the need for any human intervention. The adaptive algorithms have been implemented into the open source Azureus P2P software and were shown to be effective over a wide range of network environments.

The rest of the article is organized as followed: Section 2 reviews some previous related works; Section 3 analyzes the mechanics behind the performance problem and Section 4 develops the adaptive rate control algorithms. Section 5 evaluates the performance of the proposed adaptation algorithm and we summarize this work in Section 6.

II. LITERATURE REVIEW

Bandwidth-asymmetric broadband networks have existed for decades. Therefore numerous researchers had developed solutions to tackle performance problems in such networks. In this section, we will review more recent works which tackled the problem in the context of having competing traffic in the uplink. We will divide the solutions into three categories

namely, network layer approaches, transport layer approaches, and application layer approaches.

A. *Network Layer Approaches*

An effective network layer approach is to implement priority queuing in the network device connected to the uplink. The principle is to schedule packets from the receiver based on their packet type and gives higher priority to Ack packets [3-5]. Thus even under heavy uplink data traffic Ack packets will still not be affected.

B. *Transport Layer Approaches*

Transport layer approaches rely on modification to the transport protocol of the sender, the receiver, or both. The principle is to enable the sender transport to distinguish downlink congestion from uplink congestion, and only react to the former during congestion control [6-8]. Specifically, Since TCP uses RTT to provide congestion control, both uplink congestion and downlink congestion will affect the sending rate which is undesirable. If the transport can separate RTT into FTT (forward trip time) and BTT (backward trip time), then it will be able to react to downlink congestion only and become immune to uplink congestion. This can be accomplished using the TCP timestamp extension described in RFC1323 [9].

C. *Application Layer Approaches*

Application layer approaches refers to those solutions which can be implemented entirely within an application, without modification to the lower layers such as the transport protocol, the network protocol and network devices. Our survey revealed only one attempt in the form of an implementation called Auto-Speed in the Azureus [2] P2P client software. The Auto-Speed module in Azureus was not well-documented and based on our analysis of the source codes we found that it controls Azureus's upload data rate limit by monitoring the RTT to a Google host. An adaptive algorithm is then used to increase and decrease the limit based on variations in the continuously measured RTT.

D. *Comparisons and Contributions*

The Adaptive-DRC algorithm proposed in this paper belongs to the application layer and thus can be readily deployed without the need to modify network devices (as in network layer approaches) or operating systems (as in transport layer approaches).

Secondly, our experimental results revealed that the download throughput loss is in fact primarily due to the increased queueing delay in the uplink rather than packet losses. Thus by monitoring the RTT a P2P application can detect the onset of uplink congestion and react by cutting down the upload data rate limit to prevent congestion from occurring.

Thirdly, our results also revealed that the Auto-Speed implementation in Azureus cannot adapt to different network configurations and in many cases, underutilizes the uplink bandwidth. This is undesirable as the performance of a P2P system hinges on the amount of bandwidth its peers can contribute.

Last but not least, the proposed algorithm had been implemented into the open source Azureus P2P software which enabled us to obtain performance results from a real-world P2P system. We conducted experiments over a wide range of asymmetric network configurations, and show that the proposed algorithm can strike a good balance between uplink utilization and downlink throughput performance.

III. ANALYSIS OF THE UPLINK BOTTLENECK

To find out the reason for download throughput degradation we set up experiments as depicted in Fig. 1 using the NISTnet [10] emulator to emulate bandwidth asymmetry in the access network. All peers in the system run the Azureus P2P application. The system is set up so that the peer behind the access network – ADSL user, shares files through BT protocol with the external peer in the external network. As our focus is on the access network we do not limit the upload data rate of the external peer so that the download throughput will not be limited by the external peer. During the experiment, we gradually increase the upload limit of ADSL user every 10 seconds until it reaches the uplink capacity. There is no other competing traffic going through the link.

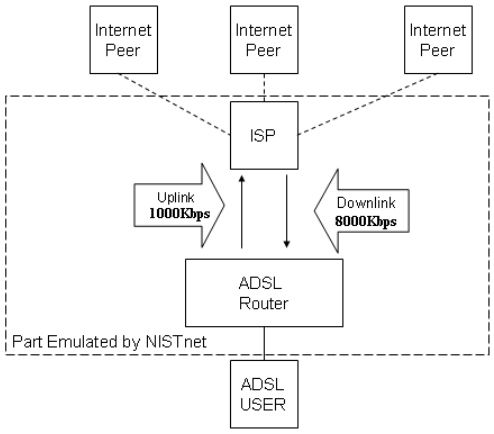

Figure 1. A P2P host running behind a bandwidth-asymmetric network

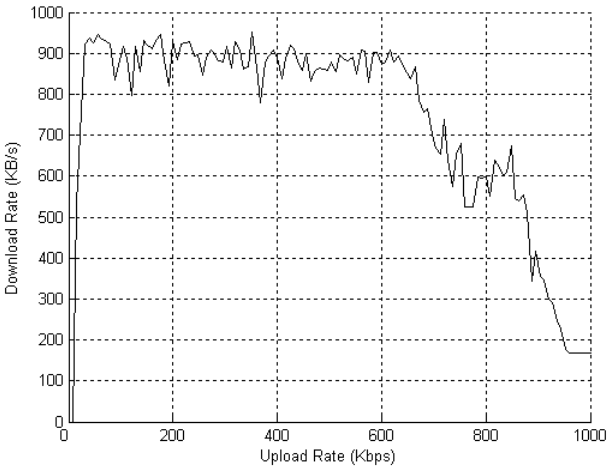

Figure 2. Effect of upload data rate on download throughput

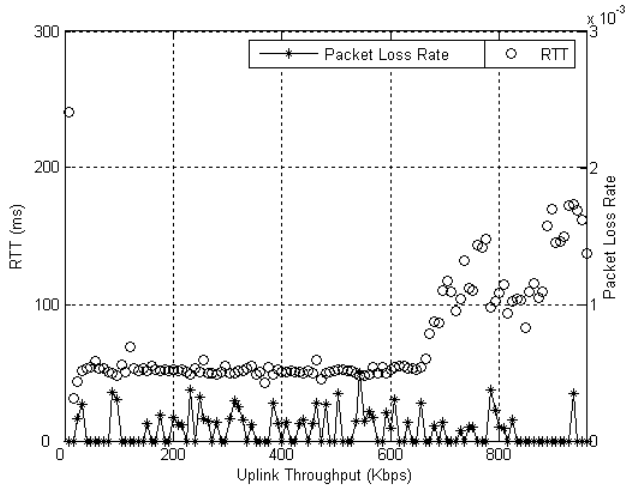

Figure 3. RTT and packet loss rate vs uplink throughput

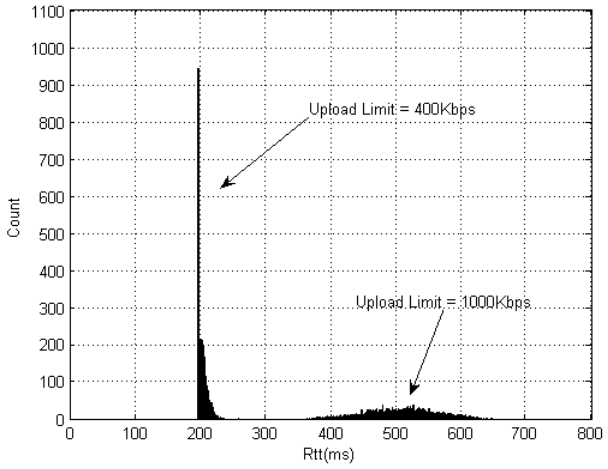

Figure 4. Comparison of RTT distributions at low and high upload data rates

TCP's throughput performance primarily depends on the path RTT and the packet loss rate [11-12]. In the experiments we measured the following at the ADSL user: download throughput, upload throughput, RTT, and packet loss. The results are summarized in Fig. 3.The results clearly show that RTT increased consistently with higher upload throughput as congestion built up. By contrast the packet loss rate remained at a low level even at high upload data rates. Contrasting this with the download throughput in Fig. 2 reveals that the download throughput degradation beginning at the upload data rate of 640 Kbps is in fact primarily due to the increased RTT. The implication of this observation is RTT can be used as an indicator of uplink congestion so that the system can react by controlling the upload data rate limit to prevent congestion from occurring.

Taking it further we conducted experiments to measure the RTT distributions when the uplink data rate limit was set to 400Kbps and 1000 Kbps respectively and plotted their RTT distribution histograms in Fig. 4. With the low upload data rate limit of 400 Kbps the RTT distribution has a small mean value of 204 ms and a narrow distribution (STD=13 ms). It is also one-sided as the minimum RTT is bounded from below by the propagation delay.

By contrast, at the high upload data rate limit of 1000 Kbps the RTT distribution has a significantly larger mean value (506 ms) and the standard deviation is far larger as well (65ms). Comparing the two distributions suggests that it is possible to detect and distinguish them so that the application can automatically adjust the upload data rate limit to prevent congestion in the uplink. We will discuss the corresponding algorithms in the next section.

IV. Adaptive Upload Data Rate Control

The principle of the proposed adaptive upload data rate control algorithms – called Adaptive-DRC, is to detect the increase in RTT caused by the uplink congestion. This can be broken down into three sub-problems: (i) RTT estimation; (ii) detection of the onset of congestion – pre-congestion detection; and (iii) adaptation of the upload data rate limit. We present below two versions of Adaptive-DRC where they differ in the way the upload data rate limit is adapted.

A. RTT Estimation

The first hurdle in RTT estimation is that it needs to be performed at the application layer. Thus while the TCP transport already has its own estimation of RTT, it may not be possible for the application to directly access it.

In case this is not possible the application will need to implement its own RTT estimation. There are many ways to accomplish this and for simplicity in our implementation we simply invoke the operating system's ping utility to measure RTT to a designated external host.

Now the next question is to find a suitable external host to perform the measurement. This is in fact not trivial in practice for two reasons. First, the external host must not be located behind a NAT (or firewall) as most NATs ignore ping requests for security reasons. This can be challenging as in a P2P network it is common for many of the users to be located behind NATs. Second, the external host must be configured to respond to ping requests. Again due to security reasons most personal firewalls, many already built-in as part of the operating system, running in the user host will block and ignore ping requests. Similar to the first problem this can be solved by implementing RTT measurement within the application's own protocols.

As both are primarily implementation issues we simply adopted the ping command as the RTT measurement tool. The external host we used in our experiments is a fixed IP address resolved from the domain name www.google.com, which consistently responds to ping requests.

B. Pre-Congestion Detection

Pre-congestion detection is divided into two phases: initialization and monitoring. During initialization when the P2P application is first started, the application will only allow data to be downloaded from external peers so that the uplink will not be congested by upload traffic. The application then conducts RTT measurements to compute the mean RTT and its standard deviation when the uplink is in non-congested state.

These statistics will serve as the baseline for comparisons made in the monitoring phase.

In the monitoring phase the application will carry out RTT measurements periodically and compare the newly measured RTT against the baseline values to determine if the uplink is developing the onset of congestion, in which case the measured RTT is expected to increase. To compensate for the inherent variations in RTT measurements, we need to devise a threshold above the mean RTT to serve as the detection criterion.

Let d be the newly measured RTT, then according to the Chebychev's inequality [13]:

$$\Pr(|X-\mu| \geq k\sigma) \leq \frac{1}{k^2} \qquad (1)$$

where μ and σ are the mean and standard deviation of the RTT respectively, which can be estimated from the measurements during the initialization phase. By choosing appropriate value for k, we can control the probability of false positive to within $1/k^2$, i.e., ≤ 1% for k=10.

C. Upload Data Rate Limit Adaptation

The upload data rate limit is adjusted according to the result from pre-congestion detection. The principle is to increase the upload data rate limit when the uplink is not congested and decrease it when the onset of congestion develops. We developed two versions of adaptive algorithm: the first one is based on Additive Increase Multiplicative Decrease (AIMD) as in TCP's congestion control algorithm, and the second one is based on Multiplicative Increase Multiplicative Decrease (MIMD).

Specifically, after each periodic RTT measurement *d* conducted in the monitoring phase the upload data rate limit is adjusted according to:

$$U_{new} = \begin{cases} U/2, if\,(d-\mu) > k\sigma \\ U+1, if\,(d-\mu) <= k\sigma \end{cases} \qquad (2)$$

for the AIMD version and

$$U_{new} = \begin{cases} U/2, if\,(d-\mu) > k\sigma \\ U\times 2, if\,(d-\mu) <= k\sigma \end{cases} \qquad (3)$$

for the MIMD case. In both cases U_{new} is the new upload data rate limit and *U* is the current upload throughput as measured internally by Azureus, which is the running average of the upload rate for the past 3 seconds. We will compare the performance of these two adaptation algorithms in the next section.

V. Performance Evaluation

We evaluate the performance of the proposed Adaptive-DRC algorithms in the context of BitTorrent using Azureus version 3.0 as the implementation. Experiments were conducted in a controlled network as depicted in Fig. 1. We conducted experiments for 8 asymmetric network configurations listed in Table 1, which represent the commonly deployed ADSL broadband services in the industry [14].

TABLE I. Asymmetric network configurations

Network Configurations	Downlink Bandwidth	Uplink Bandwidth
1	256 Kbps	64 Kbps
2	1 Mbps	64 Kbps
3	2 Mbps	256 Kbps
4	2 Mbps	512 Kbps
5	4 Mbps	1 Mbps
6	8 Mbps	640 Kbps
7	8 Mbps	1 Mbps
8	12 Mbps	1 Mbps

All hosts ran the Windows XP operating system with default installation settings. The host machines were verified to be able to saturate the downlink and uplink so that the hosts will not be the bottleneck in the experiments.

We captured the network traffic using Wireshark[15] and calculated the download and upload throughputs from the packet trace files. Each experiment run lasted for 5 minutes and a separate set of experiments were conducted for the following four scenarios on ADSL user host: (i) no upload limit; (ii) Auto-Speed; (iii) Adaptive-DRC: AIMD; and (iv) Adaptive-DRC: MIMD

In scenario (i) we did not limit the upload data rate and simply let TCP control the data rate via its built-in congestion-control algorithm. In scenario (ii) we enabled the Auto-Speed feature in Azureus to control the upload data rate limit. In scenario (iii) and (iv), we ran the proposed Adaptive-DRC AIMD version and MIMD version respectively.

We define link utilization to be the ratio between actual throughput achieved and the link bandwidth available. For example, a download throughput of 500 Kbps over a downlink of 1 Mbps will give a link utilization of 500K/1M=0.5.

We compare the link utilization for the downlink in Fig. 5 for the 8 network configurations listed in Table 1. First, without any upload limit the download throughput suffered significantly as expected. The extent of degradation is correlated to the downlink-to-uplink bandwidth ratio. For example, the poorer performing configures (network configurations 2, 3, 6, 7, 8) have larger downlink-to-uplink bandwidth ratios (with ratios of: 16, 8, 13, 8, 12) than the better performing ones (at a ratio of 4 for network configurations 1, 4, and 5). This can be explained by the observation that the amount of TCP ACKs generated on uplink is proportional to the download throughput. Thus smaller downlink-to-uplink bandwidth ratio will have relatively more uplink bandwidth for the upstream TCP ACK traffic.

Second, the performance of Auto-Speed is not consistent across the 8 network configurations. It performed best in the higher bandwidth configurations (4~8) but in configurations 1 and 2 it had nearly the lowest downlink utilization of all four scenarios.

Moreover, if we consider also the uplink utilization in Fig. 6 then we can see the reason for the observed results. The Auto-Speed algorithm turned out to be too aggressive in the uplink for network configurations 1 and 2 but too conservative for network configurations 4~8, resulting in uplink utilization lower than 0.3.

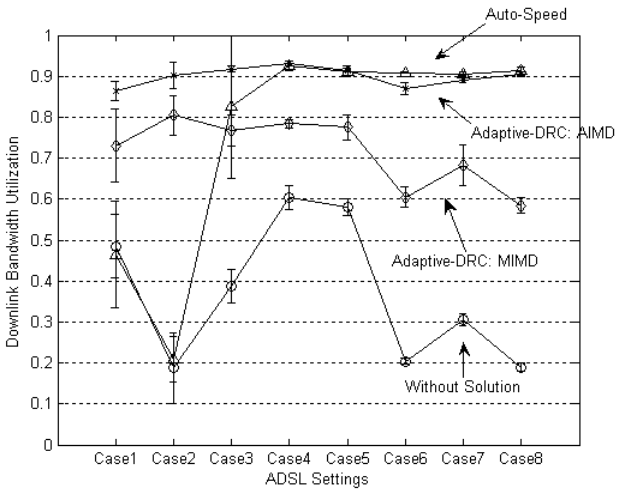

Figure 5. Comparison of downlink bandwidth utilizations

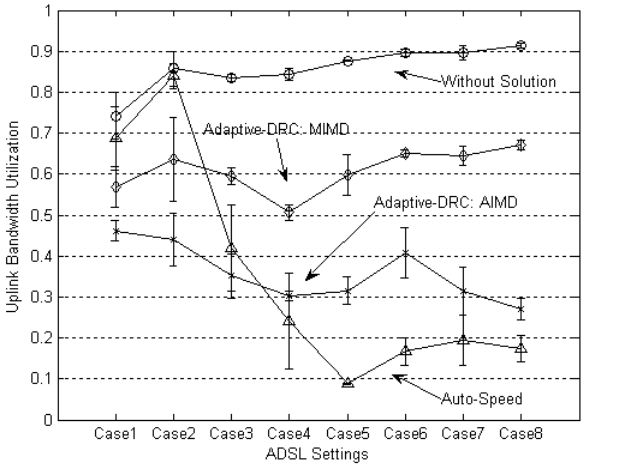

Figure 6. Comparison of uplink bandwidth utilizations

Third, performance of the proposed Adaptive-DRC is substantially more consistent by comparison. For the AIMD version, we can achieve downlink utilization above 0.8 and uplink utilization varies between 0.28 and 0.5. While for MIMD version, the downlink and uplink utilizations were maintained within 0.58 to 0.8, and 0.5 to 0.67 across the 8 network configurations. Generally speaking AIMD is more conservative in utilizing the uplink (thus leading to higher downlink throughput) while MIMD can achieve higher uplink utilization (with slightly lower downlink throughput). More importantly both Adaptive-DRC algorithms perform consistently across all 8 network configurations and thus can reliably strike a balance between uplink and downlink utilizations without the need for human intervention.

VI. CONCLUSION

This work investigated the performance issues of running P2P applications in asymmetric network environments. Without proper control of upload data rate the download throughput will be adversely affected due to significantly increased queueing delay in the uplink. This problem can be tackled at various layers and this work proposed Adaptive-DRC algorithms which can be implemented entirely within the application layer so that it can be readily incorporated into existing P2P applications without need to modify operating systems or network routers.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their comments in improving this paper. This work was funded in part by the Shun Hing Institute of Advanced Engineering of the Chinese University of Hong Kong under project number MMT 9/07.

REFERENCE

[1] "ADSL overview", [Online]. Available: http://www.wowarea.com/dyn/vge.php/g_1/k_1/o_0 [Accessed: Apr. 25, 2009]

[2] "Azureus FAQ", May, 2008. [Online]. Available: http://wiki.vuze.com/index.php/Azureus_FAQ#Azureus_FAQ_.28Frequently_Asked_Questions.29. [Accessed Apr.25, 2009]

[3] H. Balakrishnan, V. N. Padmanabhan and R. H. Katz, "The effects of asymmetry on TCP performance," in Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking, p77-89, Sept. 1997.

[4] F. Louati, C. Barakat, and W. Dabbous, "Handling Two-Way TCP Traffic in Asymmetric Networks," HSNMC 2004, LNCS 3079, pp.233-243, Sept 2004.

[5] W. Al-Khatib and K. Gunavathi, "A New Approach to Improve TCP Performance over Asymmetric Networks," Electronics and Electrical Engineering, No. 7(71), p. 13-18, 2006.

[6] Elloumi, H. Afifi, and M. Hamdi, "Improving Congestion Avoidance Algorithms for Asymmetric Networks," in Proceedings of ICC'97, Montreal, 1997, Vol. 3, pp. 1417-1421.

[7] H. Afifi, O. Elloumi and G. Rubino, "A Dynamic Delayed Acknowledgement Mechanism to Improve TCP Performance for Asymmetric Links," in Proceedings of theThird IEEE Symposium on Computers and Communications, 1998, pp. 188-192.

[8] C. Fu, L. C. Chung and S. C. Liew, "Performance Degradation of TCP Vegas in Asymmetric Networks and Its Remedies," in Proceedings of the IEEE International Conference on Communications, 2003, Vol. 7, pp. 42-44.

[9] V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance," Request for Comments 1323, May 1992.

[10] M. Carson and D. Santay, "NIST Net – A Linux-based Network Emulation Tool", Computer Communication Review, June 2003.

[11] M. Mathis, J. Semke, J. Mahdavi, T. Ott. "The macroscopic behavior of the TCP Congestion Avoidance algorithm," Computer Communications Review, vol. 27, no. 3, p67-82, July 1997.

[12] J. Padhye, V. Firoiu, D. Towsley, J. Kurose. "Modeling TCP throughput: a simple model and its empirical validation," in Preceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communications, p303-314, Sep. 1998.

[13] Mendenhall, R. Beaver, B. Beaveer, A Brief Introduction to Probability and Statistics, Wadsworth Group, 2002.

[14] "ADSL Broadband Packages". [Online]. Available:http://www.promotion.hinet.net/adsl_hot_03.htm, [Accessed, Apr.25, 2009]

[15] U. Lamping, R. Sharpe, and E. Warnicke, "Wireshark User's Guide". [Online]. Available: http://www.wireshark.org/docs /. [Accessed Apr.25, 2009].