Front. Electr. Electron. Eng. China 2010, 5(1): 15–22
DOI 10.1007/s11460-009-0069-z

REVIEW ARTICLE

Chi WANG, Hongbo WANG, Yu LIN, Shanzhi CHEN

Evolution of below-best-effort service

Abstract Providing different service priority levels for different kinds of traffic is an important research area in the Internet. As compared with the existing best-effort (BE) service, recent researchers consider devising the below-best-effort algorithm to realize a low priority service. The aim of below-best-effort service is to use the available bandwidth as much as possible in the premise of not interfering standard best-effort flows. With the best-effort transmission control protocol (TCP) transporting foreground traffic, below-best-effort service can transport background traffic which users do not urgently need to use. In this paper, we review the evolution of below-best-effort service and classify and compare these below-best-effort schemes. We also present some open problems in the research of below-best-effort service and point out the future work.

Keywords Internet, background traffic, low priority, below-best-effort service

1 Introduction

There are various kinds of applications in the Internet with different needs in bandwidth, delay, jitter, and so on. For example, some applications require broad bandwidth and small jitter like video streaming, some applications ask for short response time such as hypertext transfer protocol (HTTP), and there are also applications which have no strict requirement of bandwidth or delay, such as file transfer protocol (FTP) and peer-to-peer (P2P) file sharing. If all kinds of Internet traffic are transferred by the same best-effort (BE) service, it would degrade the users' satisfaction and cannot make good use of network resources, so it is necessary to provide differentiated service prioritization among different traffic classes [1]. In the context of Internet protocol (IP) networks, service differentiation usually implicitly means offering enhanced services, which consider the existing best-effort class as the low priority service, and developing algorithms that provide better-than-best-effort service. These better-than-best-effort services are mainly achieved in the form of reserved resources for specific flows and modification needs of the network. In recent years, the opposite approach which offers a degraded performance than the best-effort service has gained acceptance among network researchers.

Below-best-effort service is proposed as a service for carrying noncritical traffic. The usual definition of the below-best-effort service is as follows: in the event of congestion, all below-best-effort traffic must be discarded before any best-effort packet is dropped [2]. The goal of below-best-effort service is to exploit unused network capacity while protecting best-effort flows from such noncritical traffic. If best-effort transmission control protocol (TCP) flows use up the network bandwidth, below-best-effort flows should avoid sending any data till there is spare capacity again in the network. As a service that is inferior to best-effort which does not support quality of service (QoS), below-best-effort is not a trusted service either. Flows delivered by below-best-effort service have no guarantee of the flow rate, packet loss, delay, or jitter; the performance of below-best-effort flow is determined by the static network resources and the dynamic foreground traffic condition.

Below-best-effort service is useful in many application scenarios, such as Web prefetching, non time-critical, bulk-data transfer applications including P2P file sharing, remote data backup, operating system update, and so on. Below-best-effort service helps users that have demands for high volume transfers but no strict time constraints transferring their data to make use of all unutilized network resources in the premise of not interfering with higher

Received June 4, 2009; accepted September 3, 2009

Chi WANG (✉), Hongbo WANG, Yu LIN
State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
E-mail: chiziwang@gmail.com

Shanzhi CHEN
State Key Laboratory of Wireless Mobile Communication, China Academy of Telecommunication Technology, Beijing 100083, China

priority traffic. Furthermore, below-best-effort service can protect important flows from congestion and shift network usage to off-peak times.

Below-best-effort service was first raised in 1999 by Internet Engineering Task Force (IETF), and many algorithms are proposed since then. Nowadays, with the background traffic becoming popular in the Internet [3,4], below-best-effort service is becoming more and more useful, but as far as we know, there is no work presenting these below-best-effort schemes thoroughly yet. In this paper, we categorize below-best-effort service, introduce the evolution in the research of below-best-effort service, and discuss the future research direction.

The remainder of this paper is organized as follows. In Sect. 2, we classify the below-best-effort service and clarify its design objective. In Sect. 3, we introduce the below-best-effort service in diffserv networks. In Sect. 4, we discuss the end-to-end below-best-effort transmission control algorithms. We introduce the application layer below-best-effort service in Sect. 5, and a multipath background network architecture Harp in Sect. 6. Finally, we conclude and discuss the future research direction in Sect. 7.

2 Classification and design objective of below-best-effort service

According to the layer it works in, below-best-effort service can be divided into three categories: network layer, transport layer, and application layer below-best-effort service.

Network layer below-best-effort service was first raised in the diffserv working group, most notably in RFC3662 [5]. One of its first applications was to re-mark packets within multicast groups[1-3] [6]. The network layer below-best-effort service needs the support or modification of routers.

Transport layer below-best-effort protocols include TCP Nice [7], TCP low priority (TCP-LP) [8], TCP westwood low priority (TCPW-LP) [9], competitive and considerate congestion control protocol (4CP) [10], and so on. These transport layer below-best-effort protocols provide end-to-end lower than best-effort transport services from the end systems. They are realized as a sender-side modification of the TCP congestion control protocol and require no functionality change from the network routers. These protocols try to provide lower priority transport services compared with standard TCP. There are also other transport layer protocols other than TCP, such as stream control transmission protocol (SCTP) [11], explicit control protocol (XCP) [12], TCP-friendly rate control (TFRC) [13], and so on. As far as we know, existing transport layer below-best-effort protocols are all proposed based on the standard best-effort TCP.

Microsoft raised an application layer below-best-effort service [14] in 2004. Compared with the network and transport layer service, it is easier to be implemented. It uses a receive window control to limit the transfer rate of the application, and the optimal rate is determined by detecting a change point.

In general, all these below-best-effort services try to achieve the following aims:

1) Make no influence to best-effort flows. In order to achieve this goal, below-best-effort flows need to sense network congestion earlier and have a quick and aggressive back-off in the presence of congestion from TCP flows;

2) Quickly utilize the available excess bandwidth in the absence of sufficient TCP traffic;

3) Assure fairness among below-best-effort flows.

The design objective of below-best-effort service can be depicted in Fig. 1. In Fig. 1(a), there are best-effort queue and below-best-effort queue separately. In each queue, service is fair among competing flows. As the real networks do not use such queuing mechanisms, the objective of below-best-effort service is to obtain an approximation of the ideal model based on the real network. The actual network system is depicted in Fig. 1(b), where all flows are multiplexed into a single first-come-first-serve queue. Best-effort flows should obtain strictly higher priority service over below-best-effort flows, and below-best-effort flows should be fair to each other.

We should notice that if below-best-effort service is achieved by end-to-end congestion control, then the least control granularity of a flow is the round-trip time, while in

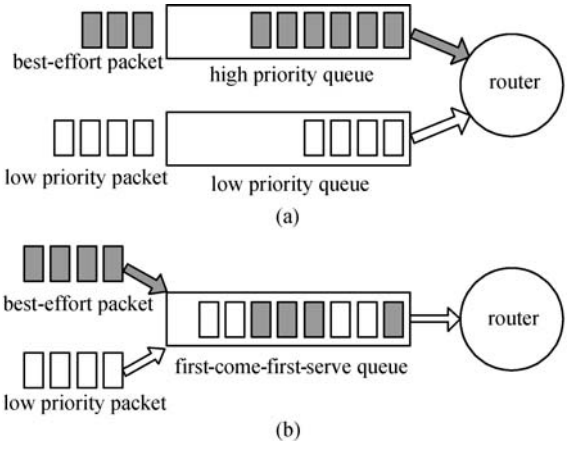

Fig. 1 Ideal and actual system. (a) Ideal model; (b) actual model

1) Internet2 QBone Initiative. QBone Scavenger Service (QBSS). http://qbone.internet2.edu/qbss/
2) TF-NGN LBE Working Group. Analysis of less-than-best-effort services. http://www.cnaf.infn.it/~ferrari/tfngn/lbe/
3) Ferrari T, Chown T, Simar N, Sabatino R, Venaas S, Leinen S. Experiments with less than best effort (LBE) quality of service. http://archive.dante.net/upload/pdf/GEA-02-067v3.pdf

the ideal model, the time granularity is the packet transmitting interval time, so it can only be nearly equal to the ideal model.

Compared with standard TCP, a strictly below-best-effort flow should have characteristics as Figs. 2(c) and 2(d) show.

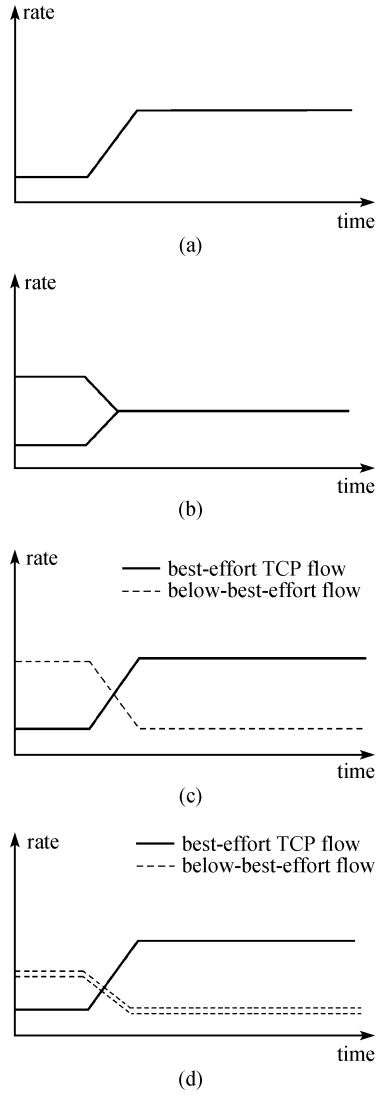

Fig. 2 Characteristics of best-effort and below-best-effort flows. (a) One TCP flow; (b) TCP versus TCP; (c) TCP versus below-best-effort flow; (d) TCP versus two below-best-effort flows

Figures 2(a) and 2(b) show that best-effort TCP flows follow the addictive increase multiplicative decrease (AIMD) congestion control algorithm and share bandwidth fairly. The situation when there are both best-effort TCP flows and below-best-effort flows is shown in Figs. 2(c) and 2(d). As Fig. 2(c) shows, below-best-effort flows would back off when standard TCP flows come into the network, which means that below-best-effort flows are transparent to higher priority flows. Figure 2(d) shows that below-best-effort flows should be fair to each other, which means that they can share the available bandwidth fairly.

3 Below-best-effort service in diffserv

Diffserv [15–18] focuses on service agreements regarding bandwidth and delay between neighboring routing domains. The differentiated service architecture is composed of a number of functional elements implemented in network nodes, including a small set of per-hop forwarding behaviors (PHB), packet classification functions, and traffic conditioning functions, which include metering, marking, shaping, and policing. Diffserv architecture is based on a simple model where the traffic entering a network is classified and conditioned at the boundaries of the network and assigned to different behavior aggregates. Each behavior aggregate is identified by a single differentiated services code point (DSCP).

Either a class selector (CS) PHB [15], an experimental/local use (EXP/LU) PHB [15], or an assured forwarding (AF) PHB [17] may be used as the PHB for the below-best-effort traffic aggregate. If a CS PHB is used, class selector 1 (DSCP = 001000) is suggested. The PHB used by the below-best-effort aggregate inside a DS domain should be configured so that its packets are forwarded onto the node output link when the link would otherwise be idle; conceptually, this is the behavior of a weighted round-robin scheduler with a weight of zero. An operator might choose to configure a very small link share for the below-best-effort aggregate and still achieve the desired goals. That is, if the output link scheduler permits, a small fixed rate might be assigned to PHB, but the behavior beyond that configured rate should be that packets are forwarded only when the link would otherwise be idle. This behavior could be obtained by using a class based queueing (CBQ) scheduler with a small share and with borrowing permitted.

In Ref. [2], Carlberg presents an approach that supports a degraded service model for IP networks using persistent class based queueing (P-CBQ). The author implemented a degradation algorithm that was based on packets counts and a corresponding drop probability as a means of selectively penalizing a specific type of flow.

RFC3662 [5] describes a lower than best-effort per-hop behavior (LBE PHB) for use within and between differentiated services domains. The primary objective of LBE PHB is to separate LBE traffic from best-effort traffic in congestion situations. LBE traffic also gets a minimal share of bandwidth so that it will not be fully preempted by best-effort traffic. The basic concept behind the proposed LBE PHB is to discard those re-marked packets more aggressively if resources become scarce. This re-marked traffic should be identifiable by a separate code point. If the queue length grows beyond a lower threshold, new arriving below-best-effort packets for this queue are going to be dropped randomly with increasing probability

until the queue length reaches an upper threshold. Beyond this upper threshold, every new incoming below-best-effort packet for this queue is going to be discarded. If the threshold values for below-best-effort packets are lower than those for best-effort packets, the queue will start discarding packets earlier.

Researchers have implemented the below-best-effort service in the European research network (GÉANT) and National Research and Education Network (NREN). Ferrari et al.[1] demonstrate an LBE service implemented on the GÉANT backbone network, including the weights and priorities for LBE, BE, and Premium IP queues. The authors choose to be interoperable with the Internet2 QBone scavenger service (QBSS)[2] by using the same DCSP value of 001000. The results show that with appropriate priority selection, an LBE service can run without any notable disruption to the BE service and with minimal packet reordering in the presence of congestion.

4 Transport layer below-best-effort congestion control

The diffserv below-best-effort service needs the support or modification of the router, so it is hard to deploy and use. Transport layer below-best-effort transmission control protocols are designed based on standard TCP; they don't need the support or any modification of routers, so they are easier to be implemented and used.

4.1 TCP Nice

TCP Nice [7,19] borrows the congestion avoidance mechanism from TCP Vegas [20]; modifications include a more sensitive congestion detector, a faster response to congestion, and a stronger mechanism for flow back-off. The TCP Nice congestion detection mechanism depends on an estimation of minimum round-trip time (RTT) and maximum RTT, and two parameters: fraction and threshold. If a fraction of the segments sent out in the last congestion window took longer than minRTT $+$ (maxRTT $-$ minRTT) $\times$ threshold, TCP Nice assumes congestion. When congestion isn't detected by this mechanism, TCP Nice falls back on the congestion avoidance rules of TCP Vegas. Upon segment loss, it will use TCP Reno's congestion control rules. Multiplicative decrease of the congestion window during congestion avoidance allows faster response to detected congestion. In TCP Nice, the congestion window is permitted to go below one, which provides a stronger mechanism for back-off and allows multiple Nice flows to be noninterference to foreground TCP flows.

NS simulations [7] show that TCP Nice is able to achieve around 70% of spare bandwidth while interfering with the foreground flow by no more than 5%.

4.2 TCP-LP

TCP-LP [8] is realized as a sender-side modification of the TCP congestion control protocol and requires no functionality from the network routers or any other protocol changes. Its congestion avoidance policy modifies the additive-increase multiplicative-decrease policy of TCP via the addition of an inference phase and use of a modified back-off policy.

TCP-LP's main algorithm is as follows: First, it gets the maximum and minimum one-way packet delay $d_{\max}$ and $d_{\min}$ in the slow start phase. TCP-LP deems $d_{\min}$ as the estimate value of one-way propagation delay and $d_{\max} - d_{\min}$ as the estimate value of the maximum queuing delay. Denote sd_i as the smoothed one-way packet delay,

$$sd_i = (1-\gamma)sd_{i-1} + \gamma d_i,$$

where γ is the delay smoothing parameter. If

$$sd_i > d_{\min} + (d_{\max} - d_{\min})\delta,$$

then an early congestion indication occurs, where δ is the delay threshold parameter. Once an early congestion indication occurs, the congestion window is halved, and an inference time-out phase is started.

The NS-2 simulations and Internet experiments [8] show that TCP-LP is largely nonintrusive to TCP traffic and, at the same time, utilizes a large portion of the excess network bandwidth fairly. File transfer times of best-effort web traffic are significantly reduced when long-lived bulk data transfers use TCP-LP rather than TCP.

Compared with TCP-LP, TCP Nice is designed as an extension to TCP Vegas, with a more sensitive congestion detector. It uses RTT-threshold-based congestion indication scheme, where the congestion is indicated if more than 50% packets encounter RTT-delay threshold.

4.3 TCPW-LP

TCP Nice and TCP-LP are proposed to provide two-class service prioritization. These protocols attempt to detect incipient congestion and significantly reduce their congestion windows before coexisting foreground flows react to the congestion. For example, TCP-LP uses queuing delay to detect incipient congestion. It sets a threshold between minimum queuing delay and maximum queuing delay, and whenever it detects congestion by comparing current queuing delay and the threshold, it halts the increase of its congestion window or reduces the window to 1. Such

1) Ferrari T, Chown T, Simar N, Sabatino R, Venaas S, Leinen S. Experiments with less than best effort (LBE) quality of service. http://archive.dante.net/upload/pdf/GEA-02-067v3.pdf
2) Internet2 QBone Initiative. QBone Scavenger Service (QBSS). http://qbone.internet2.edu/qbss/

queuing delay can be measured by the difference between current RTT and its minimum value. Although these schemes result in below-best-effort flows that are non-intrusive to foreground flows, they are often inefficient in utilizing the eligible capacity.

TCPW-LP [9] is based on TCPW [21], an extension of TCP Reno. In TCPW-LP, the window reductions are triggered not only by packet losses but also by incipient congestion so as to realize below-best-effort transfer. A TCPW-LP flow estimates the amount of its packets queued in the path routers as a backlog and reduces its congestion window if the backlog exceeds a threshold. This window reduction mechanism is called early window reduction. Upon the congestion events, it decreases the congestion window according to the eligible rate estimation (ERE) to avoid excessive window reduction and thus improve efficiency. TCPW-LP introduces a congestion metric named foreground traffic ratio. When TCPW-LP recognizes that the congestion is due to foreground flows, it sets the threshold very small to give the bandwidth to foreground flows; otherwise, it sets the threshold larger to efficiently utilize the residual capacity.

In TCPW-LP, an early window reduction (EWR) mechanism is also added to realize below-best-effort transfer. In response to an incipient congestion, EWR limits the backlog allowance of a flow by reducing the congestion window even before packet losses actually occur. This idea is similar to the window control mechanism in TCP Vegas that limits the number of backlogged packets in the bottleneck queue. As a reaction to incipient congestion, TCPW-LP congestion window is reduced when the virtual queue length exceeds a threshold. Unlike TCP Vegas which tries to fix the congestion window within an ideal range, TCPW-LP always changes congestion window through additive increase and EWR. This behavior is important to infer the existence of foreground flows and dynamically adjust the EWR threshold. In order to achieve nonintrusiveness, the threshold should be sensitive to congestion caused by coexisting foreground flows, while in order to achieve high efficiency, the threshold should not be sensitive to congestion caused by below-best-effort flows. Therefore, in TCPW-LP, the virtual queue threshold is dynamically adjusted based on both congestion level and its cause. The queue threshold is set smaller when the congestion level is higher and the congestion is caused by coexisting foreground flows.

Simulations and Internet measurements [9] show that TCPW-LP can use most of the excess bandwidth while make little influence to foreground traffic.

4.4 4CP

The transport layer below-best-effort protocols mentioned above are strictly low priority and transparent to best-effort TCP, so they are easy to be starved when there is a TCP flow in the network. Users may lack incentive using these low priority protocols because of its tendency for starvation in the presence of any traffic on a link. 4CP [10] enables provisioning of per-flow average bandwidth guarantees to normal traffic, but not at the expense of potentially starving the below-best-effort traffic.

4CP is a first practical packet-level window-based congestion controller that implements farsighted strategy [22], which was optimal for sources that value large long-run average throughput and are indifferent to sporadic throughput degradations. Following farsighted strategy, the controller distinguishes good and bad congestion phases; in the former, it aims at balancing loss probability to a target value, while in the latter, it virtually sends no data. The target loss probability is adapted very slowly so that 4CP source verifies the TCP loss-throughput formula over a large timescale. The send rate is reduced to zero in a controlled manner only during bad congestion phases.

4CP can be used in two modes: fixed target window and automatic mode. Configuration of the bandwidth guarantee is either statically configured or automatically adjusted by 4CP. The automatic mode aims to be TCP-friendly over appropriately large timescale. First, it offers a tuning knob to adjust per-flow average bandwidth guarantee for any competing normal priority connection. This is done at the sender side only in an entirely decentralized fashion and thus alleviates the need for a centralized traffic management controller. 4CP automatic mode can self-tune the parameter to adjust the average bandwidth guarantee for normal traffic so that it verifies TCP loss-throughput relation over a timescale that covers fluctuations in the network state.

4CP has been implemented in the kernel of a beta version of Microsoft Vista operating system. The validation results [23] reveal significant reductions of response times for competing short-run transfers. Extensive simulations in NS-2 and the Internet obtained by kernel implementation of 4CP show that 4CP may be beneficial to large web transfers in reducing their mean response times relative to using TCP.

We compare the end-to-end transport layer below-best-effort congestion control protocols in Table 1.

5 Application layer below-best-effort service

Microsoft researchers propose an application layer below-best-effort service named background transfer service (BATS) [14] in 2004. BATS requires neither network support nor changes to TCP. Instead, it uses a receive window control to limit the transfer rate of the application, and the optimal rate is determined by detecting a change point. The researchers phrase the idealized problem as an optimization problem and the solution as a dual control problem, where the aim is to control the window to a

Table 1 Comparison of transport layer below-best-effort protocols

protocol name	congestion signal	characteristic	parameters
TCP-LP	one-way delay	one-way delay avoids the influence of reverse traffic	γ: delay smoothing parameter δ: one-way delay threshold
TCP Nice	round-trip delay	use TCP Vegas style delay-based congestion control at sender side	f (fraction): congestion signal ratio t (threshold): delay threshold
TCPW-LP	round-trip delay	adapt the parameters dynamically according to the ratio of foreground and background flows	ERE: eligible rate estimation FTR: foreground traffic ratio Q_{th}: virtual queue threshold
4CP	round-trip delay	follows farsighted strategy and can work in automatic mode and static mode	tarp: target loss rate

critical value which represents a change point. The ideal operating point is relaxed to allow for measurement noise and to ensure fairness between competing below-best-effort flows. This application layer low priority transport can be used in large file backup, transferring updates to the current operating system (e.g., Microsoft's background intelligent transfer service (BITS)), and so on.

In the application layer below-best-effort service, two components are needed: to infer the available capacity and to adjust the sending rate of the background transfer accordingly. In application-level approach, the available capacity inference and rate adjustment are tightly coupled: the rate of the background transfer is controlled by adjusting the receiver-advertised window size, which enforces a limitation on the rate used by the application. In turn, the rate obtained for a given receiver window is used to infer whether that rate is above or below the available capacity, which triggers an adaptation of the receiver window. This is based on the fact that the actual TCP sending window is the minimum of the receiver window and the congestion window. Controlling the receiver window size also avoids overestimating the available capacity.

Fluid-flow models and an optimization framework show how the good put of the background flow can be used to detect the change point at which the background flow starts to interfere with the foreground flow. The authors propose two practical application layer approaches for background transfer: one is based on binary search, and the other is based on stochastic approximation. The two techniques have complementary strengths: binary search is quick to find a good operating point, but has inherent fluctuations, whereas stochastic optimization can be tuned for stability at the expense of convergence speed.

In the NS simulation, the author compares the binary and stochastic schemes with TCP-LP in a scenario where the throughput of the TCP flow is restricted by the receiver window size. In this case, the TCP flow does not fully utilize the bottleneck capacity, and some spare bandwidth is available to the background transfer. In the absence of background transfer, the TCP flow uses 57% of the bottleneck capacity. When the background flow is added, the utilization of the bandwidth is raised to 97% (binary search) or 100% (stochastic approximation) for control interval $T = 0.5$ s. The corresponding reductions of foreground throughput are only 2% or 6%, respectively. The results demonstrate that the background transfer utilizes the spare bandwidth while causing only slight perturbations to the foreground traffic. The results also indicate that binary and stochastic schemes utilize less available bandwidth than TCP-LP interaction with FTP, user datagram protocol (UDP), and Web traffic.

6 Multipath background network architecture Harp

Network layer below-best-effort service is hard to use in practical setting due to the lack of architectural consensus. On the other hand, although the transport layer below-best-effort protocols do not need any change to routers, modifying the applications or operating systems at end-hosts is a barrier to widespread adoption, and end users are reluctant to install transport layer below-best-effort protocols to slow down their background transfers. Besides, experiments [7] suggest that end-host-based protocols are imperfect due to delayed feedback and can significantly degrade utilization to maintain noninterference, e.g., in the presence of a competing Web workload, Nice can leave up to 50% of spare capacity on the path unused. And application-level below-best-effort service requires manual effort to configure parameters to reduce interference.

Harp [24] provides a network-wide abstraction of below-best-effort service; it is a network architecture using multipath routing. Harp is implemented as an overlay network consisting of two types of transit nodes: relays and gateways. An edge-service provider supports relays distributed across the Internet to enable path diversity, and organizations install gateways at their network exits. It can be deployed at end-hosts and uses peer nodes as relays.

The basic sequence for packet delivery in Harp is as follows: packets from the end-host enter the entry gateway and go to the exit gateway either directly or through one of

the relay nodes. The exit gateway reorders packets and delivers them to the receiving end-host. The acknowledgments generated by the receiver for all packets are sent back directly from the exit to the entry gateway. The multipath background congestion controller resides at the entry and exit gateways.

Harp's controller builds upon Kelly and Voice's (KV) multipath routing and TCP Nice. Similar to KV, Harp shares congestion information across multiple paths in a connection to achieve a fair allocation of network-wide resources. Similar to Nice, Harp uses delay-based signals to proactively detect congestion, and aggressively backs off in response.

On the arrival of a packet from the sender, the entry gateway attempts to balance load across the available paths by choosing for each packet a path. Once the entry gateway selects a path to send the packet through, it uses packet encapsulation to facilitate routing packets through the relay node. Additionally, each encapsulated packet carries a multipath header that contains a packet type, a timestamp representing when the packet left the entry gateway, and a path identifier. This header allows the exit gateway to identify the path taken by the packet, and detect and associate congestion to the path based on the delay observed by the packet. If packets are in sequence, they are sent immediately to the receiver. Otherwise, they are kept in a reorder queue till either the packets before them arrive or a timer expires.

Evaluations [24] using prototype PlanetLab as well as NS simulations demonstrate that Harp can improve transfer completion times. For example, simulations show that Harp improves foreground TCP transfer completion time by a factor of five. Prototype experiments show that background completion time can be reduced by a factor of two using two extra paths per connection.

7 Conclusions and future work

In this paper, we survey and introduce the development of below-best-effort service. Below-best-effort service in diffserv needs the support of routers; thus, it is hard to be implemented in a real network. Transport layer below-best-effort congestion control protocols don't need any special support from routers; but they need sender side modification, and this modification of TCP stack may still be difficult to deploy widely. What's more, although these schemes result in below-best-effort flows nonintrusive to foreground flows, they are often inefficient in utilizing the eligible capacity. Since the queuing delay can not indicate whether the congestion is caused by foreground flows, below-best-effort flows react to incipient congestion even if such congestion is caused by the below-best-effort flows themselves. In addition, it is difficult to set a proper queuing delay threshold because buffer sizes vary from one router to the other; furthermore, the bottleneck router can change frequently. The application layer below-best-effort service doesn't make any modification of network or transport layer, but it injects extra probe packets in the network and the utilization of bandwidth is less than end-to-end below-best-effort transport protocols. Harp is a network architecture which can achieve below-best-effort service, but it adds complicated functions in the gateway, such as reordering, congestion path control, and so on.

From the evolution of below-best-effort research, we can see that developing a below-best-effort service which needs less implementation cost is still an open problem.

Another problem is how to encourage the use of below-best-effort service. From the view of the whole network, it is obvious that using below-best-effort service can use network resources more effectively and improve the total network utility. While end-users are selfish and only care about their own utilities and satisfactions, they have no incentive to use some methods even if it improves the total utility of the network but has no improvement in their own utilities. So if the below-best-effort service cannot increase their own utilities, users are reluctant to use the service. But now, most congestion occurs in the access part of the network, that is, a user's own data create his congestion and deteriorate the network condition. So if using below-best-effort service can help users allocating their bandwidth resources more efficiently, hence increasing their own utilities, they would have the incentive to use below-best-effort service to avoid the not instantly needed flow interfering the data they really need.

Pricing [25–30] is an effective way to inspire the use of below-best-effort service. If the network charges according to the congestion, that is, more serious congestion means higher charges, then users would prefer to set the flow they don't need immediately to be below-best-effort to cut down their congestion payment. Another pricing scheme is like Paris-metro method [25]: the network charges by the rate and priority of the flow. For example, we can set the price of 1 M standard TCP flow ten times more expensive than 1 M below-best-effort flow.

In the future, with the applications and end systems becoming diverse, and with users' delay, bandwidth, and loss demand to different types of applications varying largely, service differentiation will be more and more useful. Below-best-effort service would be largely used to transport not so important data or the data users are not waiting for, so that the more important data or data with higher requirement of short delay can have a much better quality.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Grant No. 60502037), the Specialized Research Fund for the Doctoral Program of Higher Education of China (No. 200800131019), Program for New Century Excellent Talents in University (No. NECT-07-0109), and the National High Technology Research and Development Program of China (No. 2007AA01Z206).

References

1. Floyd S, Allman M. Comments on the usefulness of simple best-effort traffic. RFC5290, 2008
2. Carlberg K, Gevros P, Crowcroft J. Lower than best effort: a design and implementation. ACM SIGCOMM Computer Communication Review, 2001, 31 (suppl 2): 244–265
3. Vishwanath K V, Vahdat A. Evaluating distributed systems: does background traffic matter? In: Proceedings of USENIX 2008 Annual Technical Conference on Annual Technical Conference. 2008, 227–240
4. Ha S, Le L, Rhee I, Xu L. Impact of background traffic on performance of high-speed TCP variant protocols. Computer Networks, 2007, 51(7): 1748–1762
5. Bless R, Nichols K, Wehrle K. A lower effort per-domain behavior (PDB) for differentiated services. RFC3662, 2003
6. Chown T, Ferrari T, Leinen S, Sabatino R, Simar N, Venaas S. Less than best effort: application scenarios and experimental results. In: Proceedings of Quality of Service in Multiservice IP Networks International workshop. 2003, 131–144
7. Venkataramani A, Kokku R, Dahlin M. TCP Nice: a mechanism for background transfer. In: Proceedings of the 5th Symposium on Operating systems Design and Implementation. 2002, 329–343
8. Kuzmanovic A, Knightly E W. TCP-LP: low priority service via end-point congestion control. IEEE/ACM Transactions on Networking, 2006, 14(4): 739–752
9. Shimonishi H, Sanadidi M Y, Gerla M. Service differentiation at transport layer via TCP westwood low-priority (TCPW-LP). In: Proceedings of the Ninth International Symposium on Computers and Communications. 2004, 2: 804–809
10. Liu S, Vojnovic M, Gunawardena D. Competitive and considerate congestion control for bulk data transfers. In: Proceedings of the Fifteenth IEEE International Workshop on Quality of Service. 2007, 1–9
11. Stewart R, Xie Q, Morneault K, Sharp C, Schwarzbauer H, Taylor T, Rytina I, Kalla M, Zhang L, Paxson V. Stream control transmission protocol. RFC2960, 2000
12. Falk A, Katabi D, Pryadkin Y. Specification for the explicit control protocol (XCP). draft-falk-xcp-03.txt (work in progress), July 2007
13. Handley M, Floyd S, Padhye J, Widmer J. TCP friendly rate control: protocol specification. RFC3448, 2003
14. Key P, Massoulie L, Wang B. Emulating low priority transport at the application layer: a background transfer service. ACM SIGMETRICS Performance Evaluation Review, 2004, 32(1): 118–129
15. Nichols K, Blake S, Baker F, Black D. Definition of the differentiated services field (DS Field) in the IPv4 and IPv6 Headers. RFC2474, 1998
16. Blake S, Black D, Carlson M, Davies E, Wang Z, Weiss W. An architecture for differentiated services. RFC2475, 1998
17. Grossman D. New terminology and clarifications for diffserv. RFC3260, 2002
18. Nichols K, Carpenter B. Definition of differentiated services per domain behaviors and rules for their specification. RFC3086, 2001
19. Kokku R, Yalagandula P, Venkataramani A, Dahlin M. NPS: a non-interfering deployable Web prefetching system. In: Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems. 2003, 4: 1–14
20. Brakmo L, Peterson L. TCP Vegas: end to end congestion avoidance on a global Internet. IEEE Journal on Selected Areas in Communication, 1995, 13(8): 1465–1480
21. Mascolo S, Casetti C, Gerla M, Sanadidi M Y, Wang R. TCP westwood: bandwidth estimation for enhanced transport over wireless links. In: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking. 2001, 287–297
22. Key P, Massoulie L, Vojnovic M. Farsighted users harness network time-diversity. In: Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies. 2005, 4: 2383–2394
23. Liu S, Vojnovic M, Gunawardena D. Competitive and considerate congestion control for bulk-data transfers. Microsoft Research Technical Report 2006-24, 2006
24. Kokku R, Bohra A, Ganguly S, Venkataramani A. A multipath background network architecture. In: Proceedings of the 26th IEEE International Conference on Computer Communications. 2007, 1352–1360
25. Odlyzko A. Paris metro pricing for the Internet. In: Proceedings of the 1st ACM Conference on Electronic Commerce. 1999, 140–147
26. Yuksel M, Kalyanaraman S. A strategy for implementing smart market pricing scheme on diffserv. In: Proceedings of Global Telecommunications Conference. 2002, 2: 1430–1434
27. Kelly F P, Maulloo A K, Tan D K H. Rate control for communication networks: shadow prices, proportional fairness and stability. Journal of the Operational Research Society, 1998, 9 (3): 237–252
28. La R J, Anantharam V. Utility-based rate control in the Internet for elastic traffic. IEEE/ACM Transactions on Networking, 2002, 10(2): 272–286
29. Mandjes M. Pricing strategies under heterogeneous service requirements. Computer Networks, 2003, 42(2): 231–249
30. Wang X, Schulzrinne H. Pricing network resources for adaptive applications. IEEE/ACM Transactions on Networking, 2006, 14(3): 506–519