Avoiding Useless Packet Transmission for Multimedia over IP Networks: The Case of Multiple Congested Links

Jim Wu and Mahbub Hassan
School of Computer Science & Engineering
The University of New South Wales
Sydney 2052, Australia
Email: {jimw,mahbub}@cse.unsw.edu.au

UNSW-CSE-TR-0215

November 2002

THE UNIVERSITY OF
NEW SOUTH WALES

SYDNEY · AUSTRALIA

Abstract

In this paper, we investigate UPT avoidance problem with multiple congested links. We propose three different UPTA enforcement schemes — basic UPTA (B-UPTA), Partial UPTA (P-UPTA) and Centralised UPTA (C-UPTA). The challenge of UPT avoidance with multiple congested links is to determine global fairshare and enforce UPTA based on the global fairshare. We proposed the Bottleneck Fairshare Discovery (BFD) protocol to address this issue. BFD is a feedback mechanism proposed to assist UPTA in networks with multiple congested links. We describe the implementation of BFD with UPTA, taking WFQ as an example. Our simulation study shows that B-UPTA fails to detect UPT in some situations. P-UPTA can eventually detect UPT, but bandwidth may have been wasted on upstream links before UPT is detected. C-UPTA can avoid UPT in all situations, as it always drops useless packets at network edge. Simulation results suggest that, with C-UPTA, the achieved TCP throughput improvement is very close to the maximum theoretical value. In the paper, we also analyse the performance of C-UPTA quantitatively, in terms of TCP throughput, file download time, impact on video intelligibility, and impact on fairness. Our simulation results reveal that, for all six scenarios, the TCP throughput has been significantly improved (with improvement factor up to 50%). As a result, file download times (for various file size) have been greatly reduced (more than 30%). On the other hand, incorporation of C-UPTA into WFQ has no significant impact on intelligibility of the MPEG-2 video (with a difference less than 3%). For all six scenarios, C-UPTA maintains fairness which is comparable to WFQ. This proves that UPTA does not have any adverse impact on fairness performance of fair algorithms.

Keywords: TCP, multimedia over IP, MPEG-2, Internet, Fair Packet Queueing Algorithm.

1 Introduction

The problem of *Useless Packet Transmission* (UPT) arises as a result of (i) rising popularity of audio/video applications over the Internet, and (ii) increasing deployment of fair packet queueing/scheduling algorithms (e.g. WFQ [13], FRED [10], CSFQ [14], etc.). Current Internet supports only "*best-effort*" network service, no *Quality of Service* (QoS) is guaranteed. Packets may be dropped by routers at times of congestion. Among the two transport services available on the Internet, TCP is adaptive while UDP is non-adaptive. Most multimedia applications over the Internet are UDP-based. These applications tend to grab all network bandwidth and force TCP applications to shut up. This problem is known as *fairness* problem. Over the past few years, there has been extensive research conducted to solve the fairness problem. Weighted Fair Queueing (WFQ) [13] and Core Stateless Fair Queue (CSFQ) [14] are two well-known fair algorithms. Other fair algorithms include BLUE [4, 5], CHOKe [12], and RFQ [2]. The fairness problem in the Internet is now well recognised. Many packet queueing and discarding algorithms have been proposed in the last few years to effectively address the issue of fairness. Some network equipment manufacturers have already implemented these algorithms in their latest products. For example, WFQ has been implemented in Cisco 2600/3600/3700 Series [7] and Nortel Networks Passport 5430 [11].

The issue of UPT however, is less understood. UPT is based on the fact that for packetised audio and video, packet loss rate must be maintained under a given threshold for any meaningful communication [3, 6, 9]. When packet loss rate exceeds this threshold, received audio and video become useless. We formally defined the UPT problem, and proposed an avoidance algorithm called *Useless Packet Transmission Avoidance* (UPTA) in [16]. We have also discussed UPT avoidance in networks with *single congested link* in the paper. However, in large-scale networks (e.g., the Internet), there are usually more than one congested link on an end-to-end path. This paper presents extensions of UPTA, and associated protocols, to address performance issues germane to *multiple congested links*.

The rest of the paper is organised as follows. In Section 2, we discuss problems arise with UPTA in networks with multiple congested links. In Section 3, we present a generalised analysis on end-to-end packet loss rate of a flow traversing multiple congested links. We propose an extension to UPTA (called *partial UPTA*) in Section 4, and another alternative extension (called *centralised UPTA*) in Section 5. We describe our simulation configuration in Section 6, and present our simulation results in Section 7. Finally, we present our conclusions and discuss open issues in Section 8.

2 Scope of the Problem

In large networks, traffic flows usually experience congestion at more than one network node. That is, there are usually more than one congested link on an end-to-end path. In such a circumstance, packet loss of a flow is distributed at different congested nodes. If UPTA is enforced independently at each individual router [1], UPT may go undetected. This is because the cumulative packet loss rate may be greater than the loss threshold while packet loss rate at each individual router is less than the loss threshold. We use an example to explain the problem as following. Figure 1 is an example in which an unintelligible video flow goes undetected, as a result of distributed UPTA (let's call it basic UPTA) enforcement.

In the network shown in Figure 1, the link speed of each network link is as shown in the figure. All access links have a speed of 10 Mbps. In the example network, there are three flows, of which Flow 2 ($S_2 \rightarrow D_2$, shaded) is a UDP-based CBR MPEG-2 video application (encoded at 1.5 Mbps). Other flows are greedy TCP flows. Assume the MPEG-2 video application has a loss threshold of 12%. In the

[1] We refer to such implementation of UPTA as "basic UPTA" (B-UPTA) hereafter.

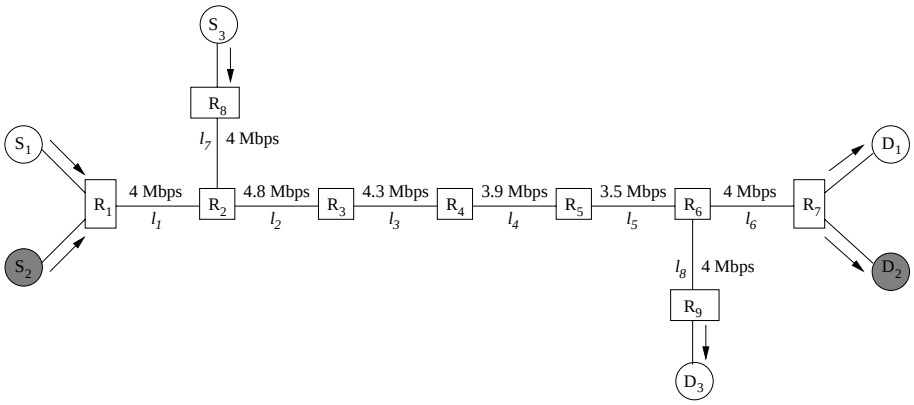


Figure 1: UPT in networks with multiple congested links.

Table 1: Packet Loss Rate of Video Flow at Various Nodes

Node	*Arrival Rate* (Mbps)	*Dept. Rate* (Mbps)	*Loss Rate* %)
R_1	1.76	1.76	0
R_2	1.76	1.60	9.1
R_3	1.60	1.43	10
R_4	1.43	1.30	9.1
R_5	1.30	1.17	10
R_6	1.17	1.17	0
R_7	1.17	1.17	0
D_2	1.17	N/A	33.5

example network, assume UPTA is implemented in all routers, in conjunction with WFQ (let's call it WFQ+). As the MPEG-2 video is encoded at a bit rate of 1.5 Mbps, packet arrival rate of the video application (at router R_1) is about 1.76 Mbps, after taking TS header (4 bytes), UDP header (8 bytes), and IP header (20 bytes) into account. Based on *max-min* fair allocation criterion [8], we calculate packet arrival rate, packet departure rate, and packet loss rate for the video flow, as shown in Table 1.

From Table 1 we can see that the packet loss rate (of the video flow) seen by routers R_1–R_7 is less than the loss threshold of 12%. Therefore, the video flow is considered as an I flow at each individual router (R_1–R_7). However, as we can see in the table, the end-to-end packet loss rate (cumulative loss rate) of the video application (as perceived by the video receiver D_2) is more than 30%, which exceeds the the loss threshold of 12%. Therefore, the video flow is actually transmitting useless packets, although it is not detected by the network. A more generalised analysis of end-to-end packet loss rate is provided in the following section.

3 Analysis of End-to-End Packet Loss Rate

In this section, we conduct a generalised analysis on end-to-end packet loss rate of a flow traversing multiple congested links. Our analysis results show that the end-to-end packet loss rate increases exponentially with the increase of number of congested links. Therefore, a multimedia flow may suffer an

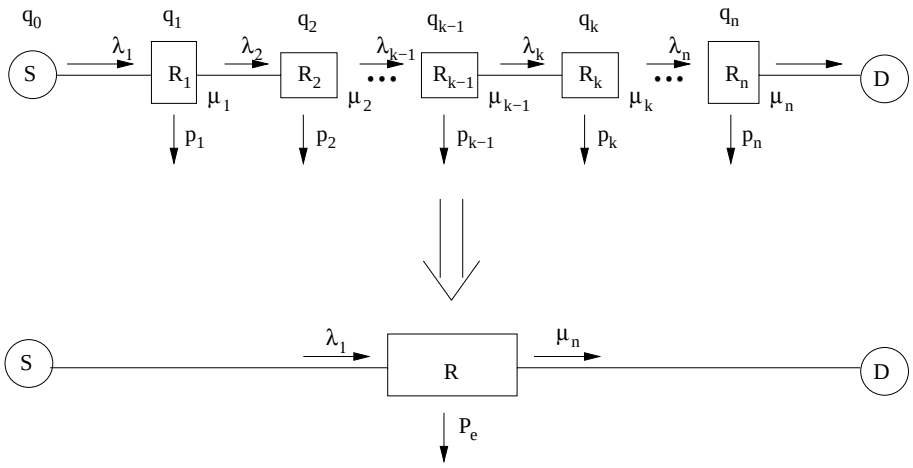


Figure 2: Analysis model of packet loss with multiple congested links.

unacceptably large end-to-end packet loss rate, even though packet loss rate at each individual router is very small.

Assume there are n routers $R_1, R_2, \ldots, R_n$ on an end-to-end path between a source and its destination (as shown in Figure 2), we can use fluid model to derive the cumulative packet loss rate (for a given flow) at router R_k ($k = 1, 2, \ldots, n$) mathematically. For a given flow f, we use following notations:

- λ_k: Packet arrival rate at router R_k.
- μ_k: Packet departure rate at router R_k .
- p_k: Local packet loss rate (LPLR) at router R_k ($p_k = 1 - \frac{\mu_k}{\lambda_k}$).
- P_k: Cumulative packet loss rate (CPLR) at router R_k.
- P_e: End-to-end packet loss rate (CPLR at the destination).

According to fluid model, we have:

$$\begin{aligned} \mu_1 &= (1 - p_1)\lambda_1 \\ \mu_2 &= (1 - p_2)\lambda_2 \\ &\vdots \\ \mu_k &= (1 - p_k)\lambda_k \end{aligned} \tag{1}$$

Assume all links are error-free, we have:

$$\begin{aligned} \lambda_2 &= \mu_1 \\ \lambda_3 &= \mu_2 \\ &\vdots \\ \lambda_k &= \mu_{k-1} \end{aligned} \tag{2}$$

Solving Eq. (1) and Eq. (2) for μ_k, we have:

$$\begin{aligned}\mu_k &= (1-p_1)(1-p_2)\cdots(1-p_k)\lambda_1 \\ &= \lambda_1 \prod_{i=1}^{k}(1-p_i) \end{aligned} \tag{3}$$

On the other hand, as per fluid model we have:

$$\mu_k = (1-P_k)\lambda_1 \tag{4}$$

Solving Eq. (3) and Eq. (4) for P_k, we have:

$$P_k = 1 - \prod_{i=1}^{k}(1-p_i) \tag{5}$$

Assume no packet loss occurs at the destination host, we can work out the end-to-end packet loss rate P_e based on Eq. (5).

$$P_e = P_n = 1 - \prod_{i=1}^{n}(1-p_i) \tag{6}$$

From Eq. (6) we can see that, the end-to-end packet loss rate P_e increases with the increase of n (number of hops between the source and the destination). Taking the limit of Eq. (6), we have:

$$\begin{aligned}\lim_{n\to\infty} P_e &= \lim_{n\to\infty} P_n \\ &= \lim_{n\to\infty}(1 - \prod_{i=1}^{n}(1-p_i)) \\ &= 1 \qquad (\text{for } 0 < p_i < 1, i = 1, 2, \ldots, n) \end{aligned} \tag{7}$$

Figure 3 illustrates the relationship between P_e and n graphically. The figure plots P_e against n with various LPLRs which are all less than the loss threshold for our MPEG-2 video (12%). The figures shows that P_e increases rapidly with the increase of n. This implies that, in networks with multiple congested links, the end-to-end packet loss rate may exceed the loss threshold even though packet loss rate at each individual router is less than the threshold. As shown in Figure 3, P_e is greater than 12% for all five LPLRs when n is greater than 4. This implies that the combined effect is significant even if LPLRs are very small.

It is clear from above analysis, that in order to eliminate UPT completely in networks with multiple congested links, there is a need to investigate extensions to our basic UPTA algorithm (which operates based on only LPLR). In the rest of this chapter, we propose and evaluate two such extensions, Partial UPTA (P-UPTA) and Centralised UPTA (C-UPTA).

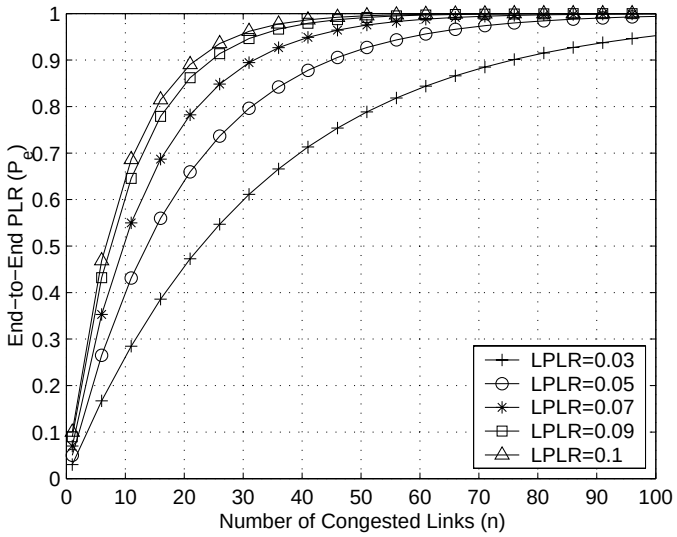


Figure 3: End-to-end packet loss rate as a function of LPLR and number of congested links.

4 Partial UPTA (P-UPTA)

The problem we discussed in Section 2 stems from the fact that all routers use the same loss threshold to detect UPT. As the same loss threshold is used, packet loss at upstream routers is not taken into account. Partial UPTA employs variable (decreasing) loss threshold, i.e. the loss threshold q is updated (decremented) at each router based on local packet loss rate (LPLR) at the router. Because P-UPTA considers decreasing loss threshold at each router, it is possible for it to detect UPT somewhere between the source and the destination. This section describes the algorithm of P-UPTA, and derives the formula for updating loss threshold at each router.

4.1 Algorithm

In P-UPTA, packet loss threshold q is carried in packet header and passed on to each router on the end-to-end path. Figure 4 illustrates the flow-chart of P-UPTA.

In Figure 4, shaded functional blocks represent basic UPTA algorithm (described in [16]); the other functional block (not shaded) represents extension to basic UPTA. We use the network shown in Figure 2 to explain the operation of P-UPTA as below. Packets generated by the source are set with loss threshold of q_0 ($q = q_0$). When a packet arrives at router R_1, R_1 extracts the loss threshold from packet header ($q_1 = q_0$). R_1 computes drop probability p_1 for the packet. The packet will be dropped immediately if $p_1 > q_1$. Otherwise, R_1 drops the packet with probability of p_1. If the packet is accepted, R_1 will compute a new loss threshold for the packet using the threshold update formula described in Section 4.2 (Eq. (17)). R_1 then updates the loss threshold field with the new threshold (q_2) which will be used by router R_2 to detect UPT. Similarly, the packet will be processed by $R_2, \ldots, R_k, \ldots, R_n$ (using the same algorithm) while it travels along. In this way, P-UPTA takes into account packet loss in upstream routers by reducing the loss threshold at each router.

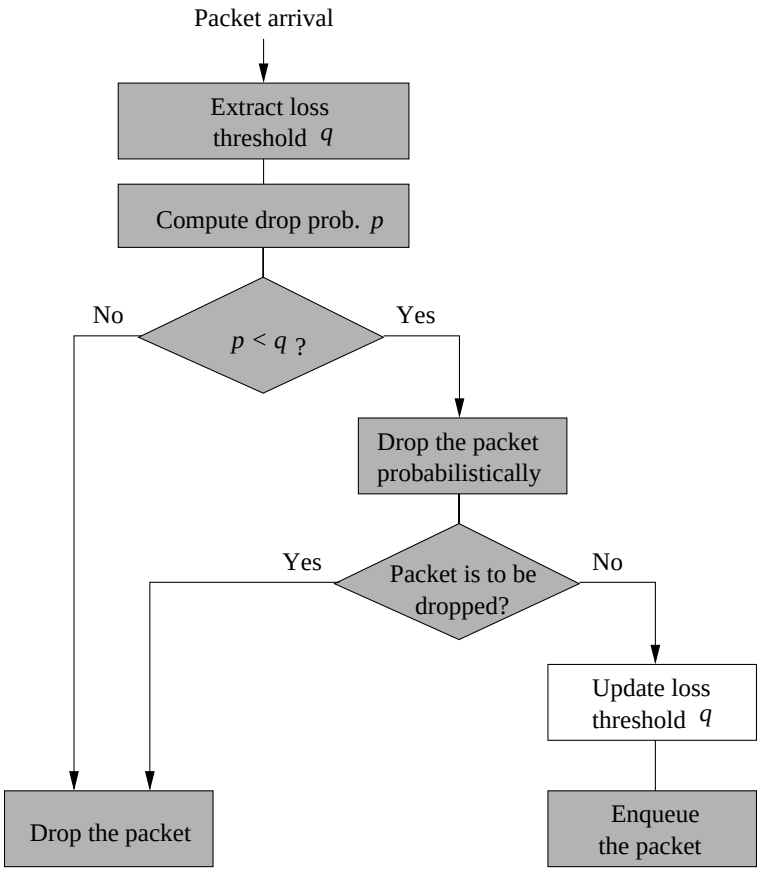


Figure 4: Flow-chart of Partial UPTA (P-UPTA) algorithm.

4.2 Update of Loss Threshold

In P-UPTA, a router rewrites loss threshold q in packet header for packets departing the router, based on the original value of q (set by the upstream router) and packet loss rate at the router. Assume q_k ($k = 1, 2, \ldots, n$) denotes loss threshold for packets arrive at router R_k (q_k is set by R_k's upstream router R_{k-1}), we can derive q_k as following (see Figure 2).

According to Eq. (6), the cumulative packet loss rate at the k^{th} router (R_k) can be expressed as:

$$P_k = 1 - \prod_{i=1}^{k}(1 - p_i) \qquad (k = 1, 2, \ldots, n) \tag{8}$$

Assume the original loss threshold set by an application is q_0. Because router R_1 is connected directly to the source host, we have:

$$q_1 = q_0$$

As per Eq. (8), the cumulative packet loss rate at R_2 is given by:

$$P_2 = 1 - (1 - p_1)(1 - p_2) \tag{9}$$

Because q_2 represents the maximum tolerable loss rate at R_2, we have:

$$P_2 = q_0, \text{if } p_2 = q_2 \tag{10}$$

Substitute Eq. (10) into Eq. (9), we have:

$$q_0 = 1 - (1 - p_1)(1 - q_2) \tag{11}$$

Solving Eq. (11) for q_2 we have:

$$q_2 = 1 - \frac{1 - q_0}{1 - p_1} = \frac{q_0 - p_1}{1 - p_1}$$

At router R_3, we have:

$$q_0 = 1 - (1 - p_1)(1 - p_2)(1 - q_3) \qquad \text{if } p_3 = q_3$$

Therefore,

$$q_3 = 1 - \frac{1 - q_0}{(1 - p_1)(1 - p_2)} \tag{12}$$

From Eq. (11) we have:

$$1 - p_1 = \frac{1 - q_0}{1 - q_2} \tag{13}$$

Substitute Eq. (13) into Eq. (12) we have:

$$q_3 = 1 - \frac{1 - q_2}{1 - p_2} = \frac{q_2 - p_2}{1 - p_2}$$

Similarly, we have:

$$q_{k-1} = 1 - \frac{1 - q_0}{\prod_{i=1}^{k-2}(1 - p_i)} \tag{14}$$

$$q_k = 1 - \frac{1 - q_0}{\prod_{i=1}^{k-1}(1 - p_i)} = 1 - \frac{1 - q_0}{(1 - p_{k-1})\prod_{i=1}^{k-2}(1 - p_i)} \tag{15}$$

From Eq. (14) we have:

$$\prod_{i=1}^{k-2}(1 - p_i) = \frac{1 - q_0}{1 - q_{k-1}} \tag{16}$$

Substitute Eq. (16) into Eq. (15) we have:

$$q_k = 1 - \frac{1 - q_{k-1}}{1 - p_{k-1}} = \frac{q_{k-1} - p_{k-1}}{1 - p_{k-1}} \tag{17}$$

We call Eq. (17) *threshold update formula*. P-UPTA uses this formula to update loss threshold at each individual router (see Figure 4). Comparing Eq. (14) and Eq. (15) we can see that $q_k < q_{k-1}$ provided $0 < p_{k-1} < 1$. This proves that loss threshold is reduced at each router as packets travel from the source to the destination. This paves the way for P-UPTA to detect UPT, if the end-to-end packet loss rate is greater than the loss threshold of a multimedia application.

5 Centralised UPTA (C-UPTA)

Although P-UPTA represents an improvement over B-UPTA, it has its limitation. As we described in Section 4, P-UPTA usually detects UPT somewhere between the source and the destination. UPT will be eliminated only from the point of detection, leaving UPT active on upstream links. In other words, P-UPTA cannot eliminate UPT completely. In this section, we propose C-UPTA, another alternative extension to B-UPTA, which is capable of eliminating UPT in the entire end-to-end path.

5.1 Upper Bound for TCP Throughput Improvement

C-UPTA can significantly improve TCP throughput by completely eliminating useless packets from multimedia flows. In this section, we derive the upper bound for TCP throughput improvement in C-UPTA. We define TCP throughput improvement factor ρ, so that:

$$\tau' = (1 + \rho)\tau$$

where τ is TCP throughput without C-UPTA; and τ' is TCP throughput with C-UPTA implemented. We can derive TCP throughput improvement factor ρ as below. Assume TCP is greedy, its throughput τ (without UPTA implemented) equals to fairshare of the bottleneck link α), i.e.

$$\tau = \alpha$$

With C-UPTA implemented, the TCP throughput equals to α and α', in I intervals and U intervals respectively. α and α' represent the fairshare of the bottleneck, in I intervals and U intervals respectively. α and α' are given in Eq. (18) and Eq. (19) [16]:

$$\alpha = \frac{C - \sum_{i=1}^{S} \lambda_i}{N - S} \tag{18}$$

$$\alpha' = \frac{C - \sum_{i=1}^{S} \lambda_i}{N - S - M} \tag{19}$$

in which C denotes the output link capacity; N denotes the total number of flows; S denotes the number of unconstrained flows; M denotes the number of flows currently in U intervals; and λ_i denotes the packet arrival rate of Flow i. Thus, the TCP throughput with C-UPTA implemented can be expressed as:

$$\tau' = (1 - \phi)\alpha + \phi\alpha' = (\alpha' - \alpha)\phi + \alpha$$

where α' is the fairshare with UPT eliminated; and ϕ is unintelligible ratio of the video flow (defined in [16]). Therefore, TCP throughput improvement factor ρ can be expressed as:

$$\begin{aligned} \rho &= \frac{\tau'}{\tau} - 1 \\ &= \frac{(\alpha' - \alpha)\phi + \alpha}{\alpha} - 1 \\ &= \frac{\alpha' - \alpha}{\alpha}\phi \\ &= (\frac{\alpha'}{\alpha} - 1)\phi \end{aligned} \tag{20}$$

Substitute Eq. (18) and Eq. (19) into Eq. (20), we have:

$$\begin{aligned} \rho &= (\frac{\frac{C - \sum_{i=1}^{S} \lambda_i}{N - S - M}}{\frac{C - \sum_{i=1}^{S} \lambda_i}{N - S}} - 1)\phi \\ &= (\frac{N - S}{N - S - M} - 1)\phi \\ &= \frac{M}{N - S - M}\phi \end{aligned} \tag{21}$$

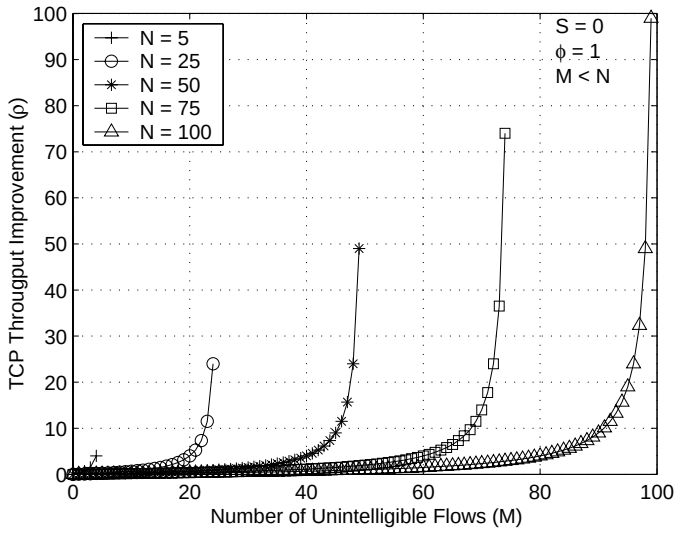


Figure 5: Theoretical TCP improvement against varying number of U flows in C-UPTA.

For the network shown in Figure 1, $N = 3$; $M = 1$; $S = 0$. Thus, TCP throughput improvement factor ρ can be calculated as:

$$\begin{aligned} \rho &= \frac{M}{N - S - M}\phi \\ &= \frac{1}{3 - 0 - 1}\phi \\ &= 0.5\phi \end{aligned} \tag{22}$$

TCP throughput improvement factor ρ under various network configuration is plotted in Figure 5. As shown in the figure, ρ increases with the number of unintelligible flows M. For a given number of flows (which share the same bottleneck link) N, ρ increases more rapidly when M is greater than a certain value.

5.2 Architecture

Figure 6 illustrates the architecture for C-UPTA. The architecture consists of edge routers, core routers and hosts. Edge routers are located at network edge. They are connected directly to users (Hosts in Figure 6). An edge router may also interconnect two networks (e.g. networks operated by two different ISPs) if traffic flows traverse multiple networks.

In C-UPTA, we make distinction between two types of edge routers — *ingress* and *egress* routers. An ingress router is the edge router where traffic flows enter a network, while an egress router is the edge router where traffic flows exit from a network. Core routers are located in network core (or network backbone). They have no direct connection to users. The feedback mechanism facilitates communication between edge routers. This allows an ingress router to determine the global fairshare for an end-to-end path and drop packets in the first place, i.e. the ingress router. In Figure 6, UPTA is needed at the ingress router only. The UPTA pseudo-code implemented in the ingress router is shown in Figure 7.

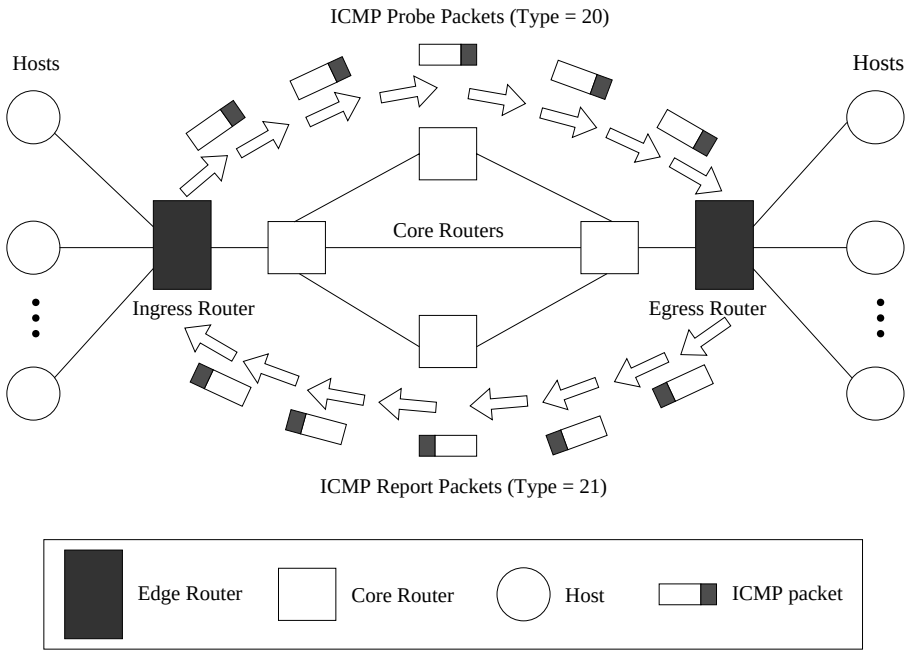


Figure 6: Centralised UPTA (C-UPTA) architecture.

C-UPTA relies on a feedback mechanism (described in Section 5.3) to determine the bottleneck link on an end-to-end basis, and enforces UPTA centrally at the ingress router. As long as the information about the bottleneck link (e.g. fairshare) is known to the ingress router, the problem can be reduced to UPT avoidance with single congested link which is discussed in [16].

5.3 Bottleneck Fairshare Discovery (BFD) Protocol

BFD is proposed to find out the fairshare of the bottleneck link (i.e. the most congested link) on an end-to-end path. The fairshare of the bottleneck link is thereby referred to as *global fairshare* [1]. As shown in Figure 6, the ingress router acquires global fairshare information (for flows that share the same ingress and egress routers) by exchanging probe/report packets between the ingress and egress routers periodically. We propose two ICMP messages for this purpose. They are *QoS Probe* and *QoS Report* ICMP messages. Figure 8 illustrates packet format for these ICMP messages.

QoS Probe messages are generated by the ingress router to probe for end-to-end QoS (e.g. global fairshare, delay, etc.). QoS Report messages are generated by the egress router to report end-to-end QoS parameters, in response to receipt of each QoS Probe message. We discuss the format of QoS Probe/Report messages as following.

5.3.1 Header Format of QoS Probe/Report ICMP Messages

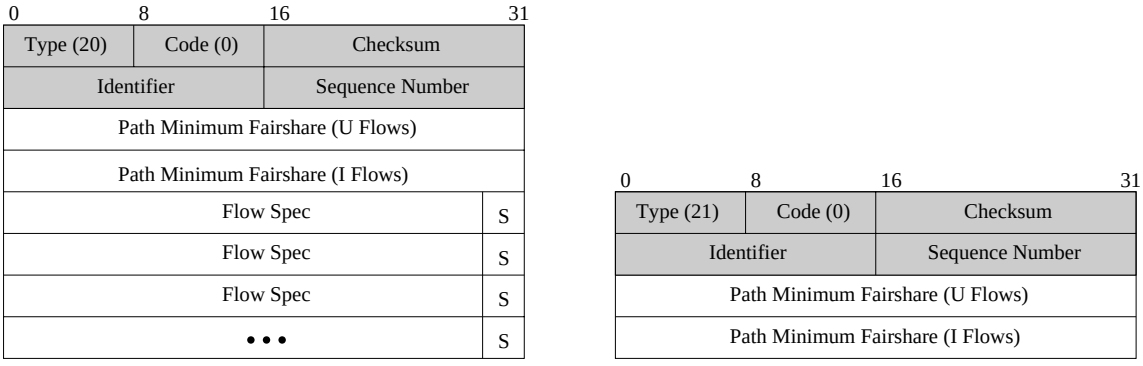
As shown in Figure 8, QoS Probe and QoS Report messages have the same ICMP header format (shaded fields in Figure 8). The QoS Probe/Report message header is made up of five fields. They are *Type* (8 bits), *Code* (8 bits), *Checksum* (16 bits), *Identifier* (16 bits), and *Sequence Number* (16 bits) fields.

The Type field specifies an ICMP message type. For QoS Probe messages (Figure 8(a)), the Type field is assigned a value of 20. For QoS Report messages (Figure 8(b)), a value of 21 is assigned. The Code

```
On packet arrival:
/* Estimate packet arrival rate */
flow_state[i].arv_rate = est_rate();
if (flow_state[i].int == I) /* I flow */
    drop_prob = max(0, 1−fs_I/flow_state[i].arv_rate);
    if (drop_prob > loss_threshold)
        flow_state[i].int = U;
        drop the packet;
    else
    /* Drop the packet probabilistically */
        if (drop_prob > unif_rand(0,1))
            drop the packet;
        else
            enqueue the packet;
else /*U flow or new flow */
    drop_prob = max(0, 1−fs_U/flow_state[i].arv_rate);
    if (drop_prob > loss_threshold)
        drop the packet;
        flow_state[i].int = U;
    else
        enqueue the packet;
        flow_state[i].int = I;

/*** Fairshare Estimation ***/
On est_timer expiration:
N = list_size(flow_state);
M = 0;
S = 0;
for (i=0; i<N; i++)
    if (flow_state[i].int == U)
        ++M;
    if (flow_state[i].arv_rate < C/N)
        ++S;
update fs_U;
update fs_I;
restart est_timer;
```

Figure 7: Pseudo-code for implementation of C-UPTA in WFQ.



(a) QoS Probe

(b) QoS Report

Figure 8: QoS Probe/Report ICMP message format.

field indicates the type of QoS information an ingress router is probing for. Because we only consider bandwidth in this research, the Code field is set to 0. Other code values may be used to indicate other QoS parameters.

The Checksum field is used to detect errors in ICMP messages. It carries the checksum of the entire probe/report ICMP message. The Identifier field identifies the output port for which a probe message is generated. It carries the output port number of an ingress router. The Sequence Number field is used to associate report messages with probe messages.

5.3.2 Payload of QoS Probe ICMP Messages

The payload of QoS Probe messages (Figure 8(a)) consists of *Path Minimum Fairshare (U flows)*, *Path Minimum Fairshare (I flows)*, and a list of *Flow Specs*. The two path minimum fairshare fields are used to record the minimum fairshare on the network path between an ingress router and its corresponding egress router (for U flows and I flows respectively). Let's assume that there are N routers on the network path between the ingress router and its egress router (including the ingress and egress routers). As each router will compute its local fairshare α_l and α'_l ($l = 1, 2, \ldots, N$). The minimum fairshare (on the network path) can be expressed as:

$$\alpha_{min} = min(\alpha_1, \alpha_2, \ldots, \alpha_N) \tag{23}$$

$$\alpha'_{min} = min(\alpha'_1, \alpha'_2, \ldots, \alpha'_N) \tag{24}$$

When the ingress router generates a QoS Probe packet, the path minimum fairshare fields are set to the local fairshare at the ingress router. As the probe packet travels through the network (from ingress router to egress router), the two path minimum fairshare fields are updated by intermediate routers as below:

$$\alpha_{min} = \alpha_l \qquad \text{if } \alpha_l < \alpha_{min} \qquad (l = l, 2, \ldots, N) \tag{25}$$

$$\alpha'_{min} = \alpha'_l \qquad \text{if } \alpha'_l < \alpha'_{min} \qquad (l = l, 2, \ldots, N) \tag{26}$$

The list of *Flow Specs* contains flow identity information for all flows sharing the same ingress/egress pair. A Flow Spec uniquely identifies a flow. In IPv6, a flow is identified by the *Flow Label* field in IP header. in IPv4, a flow can be identified by a tuple of source IP address, source port number, destination IP address, and destination port number. Each Flow Spec is followed by an 1-bit *S bit*. S bit is set by the ingress router. It indicates the intelligibility status of a flow. A value of 0 represents U flows, and a value of 1 represents I flows. The intelligibility status of a flow is necessary for fairshare estimation at intermediate routers (see Eq. (18) and Eq. (19)).

5.3.3 Payload of QoS Report ICMP Messages

The payload of QoS Report messages (Figure 8(b)) is made up of two fields: *Path Minimum Fairshare (U flows)* and *Path Minimum Fairshare (I flows)* fields. These two fields are set to the value of α_{min} and α'_{min} respectively by the egress router. They are sent back to the ingress router (in Report ICMP messages). This allows the ingress router to enforce UPTA centrally at network edge.

QoS probe packets are sent at an interval of T. The choice of a T value is a trade-off between network responsiveness and protocol overhead. A small value of T implies more probe packets will be sent by the ingress router in a given period of time. This enables the ingress router to adapt quickly to changes in global fairshare. However, the protocol overhead will be high if T is too small. On the other hand, if a large value of T is used, the network will be less responsive to network dynamics although the protocol overhead is small. In our simulations, T is set to the value of K which is a parameter for packet arrival rate estimation [16].

6 Simulation

Using OPNET Modeler 7.0.B [15], we simulate the network we discussed in Section 2 (see Figure 1). The topology simulates an intranet with three long-distance sites. In the simulated network, a greedy TCP source (S_1) competes for bandwidth with a UDP video source (S_2, shaded). Another UDP source (S_3) simulates background traffic. It generates dynamic traffic to create changes in fairshare on the network. In the network as shown in Figure 1, all workstations are connected to the network through 10 Mbps access links. Link speeds for network links are as shown in the figure. All routers have a buffer size of 100 packets. The propagation delay for network links is 1 ms; propagation delay for access links is 1 μs.

In the simulated network, the TCP source (S_1) simulates bulk file transfer. The video source (S_2) simulates transmission of an MPEG-2 video clip called `susi_015.m2v`. The video source, operating at 1.5 Mbps, employs a packet size of 188 bytes (MPEG-2 TS packet size). For other applications, a packet size of 512-byte is used. S_3 is an ON-OFF source which transmits at 10 Mbps during ON period, and stops transmitting in OFF periods. The ON-OFF source simulates the dynamics of background traffic over the network.

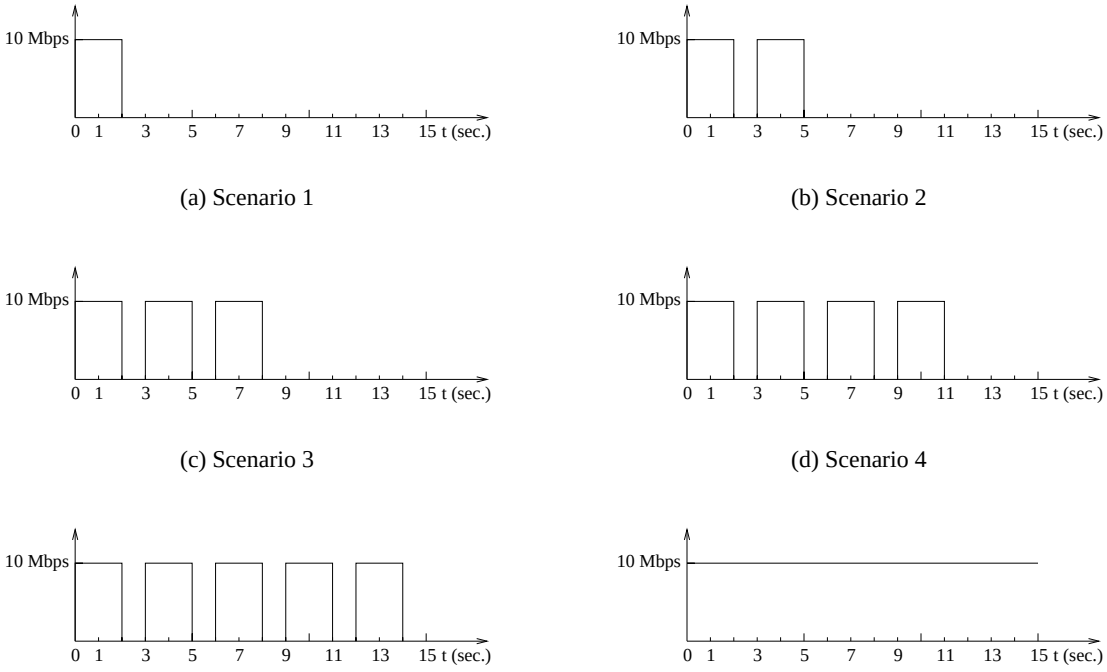
We induce and control the length of U intervals in the video connection by controlling the ON-OFF source S_3 as follows. The MPEG-2 video is encoded at a bit rate of 1.5 Mbps. Taking TS header (4 bytes), UDP header (8 bytes), and IP header (20 bytes) into account, the total bandwidth required by the video application is about 1.76 Mbps. When the ON-OFF source is off, the fairshare of the bottleneck link (l_5) is 1.75 Mbps; the video connection suffers almost no packet loss. However, when the ON-OFF source is on, the fairshare of link l_5 drops to 1.17 Mbps, the video connection suffers a packet loss rate beyond the loss threshold. Therefore, by controlling the ON-OFF source on, we can effectively induce U intervals in the video connection.

The length of the simulation is set to 15 seconds, which corresponds to the length of the video clip. To simulate different levels of UPT, we varied the number of ON periods of the background traffic, resulting in six different simulation scenarios. Figure 9 shows the traffic profile for the background traffic in various simulation scenarios.

As shown in Figure 9, there are 1–5 ON periods for Scenarios 1–5, respectively. Scenario 6 represents the worst case in which the video connection is rendered useless for the entire simulation duration ($\phi = 1$).

7 Results

We have incorporated C-UPTA in WFQ. Extensive simulations have been conducted to evaluate the effectiveness of C-UPTA, using following performance metrics (described in [16]):



(a) Scenario 1

(b) Scenario 2

(c) Scenario 3

(d) Scenario 4

(e) Scenario 5

(f) Scenario 6

Figure 9: Background traffic profile in various simulation scenarios.

- TCP throughput
- File download time
- Video intelligibility
- Throughput fairness

This section presents our simulation results.

First, we ran six simulations (Scenarios 1–6) with WFQ implemented in the network. Figure 10 and 11 show the Cumulative Distribution Function (CDF) of end-to-end delay and packet loss rates for the video connection. From Figure 10 we can see that the end-to-end delay of video packets increases with the increase of UPT in the network. For example, in Scenario 1, the probability of delays less than (or equal to) 0.05 sec. is over 0.8, whereas in Scenario 6, the probability is almost zero. Another observation we can make from Figure 10 is that video packet delays are less than 0.11 sec. (or 110 ms) in all scenarios. That is, the video packet delays are less than the tolerable delays (150–400 ms) for streaming video [3, 9]. This confirms that delays do not contribute to UPT in the simulated network. On the other hand, as we can see in Figure 11, packet loss rates for the video connection in U intervals (when the ON-OFF source is on) are around 33%, greater than the threshold of 12% for intelligible communication. Therefore, in our simulation study that follows, we only consider the impact of packet loss on video intelligibility.

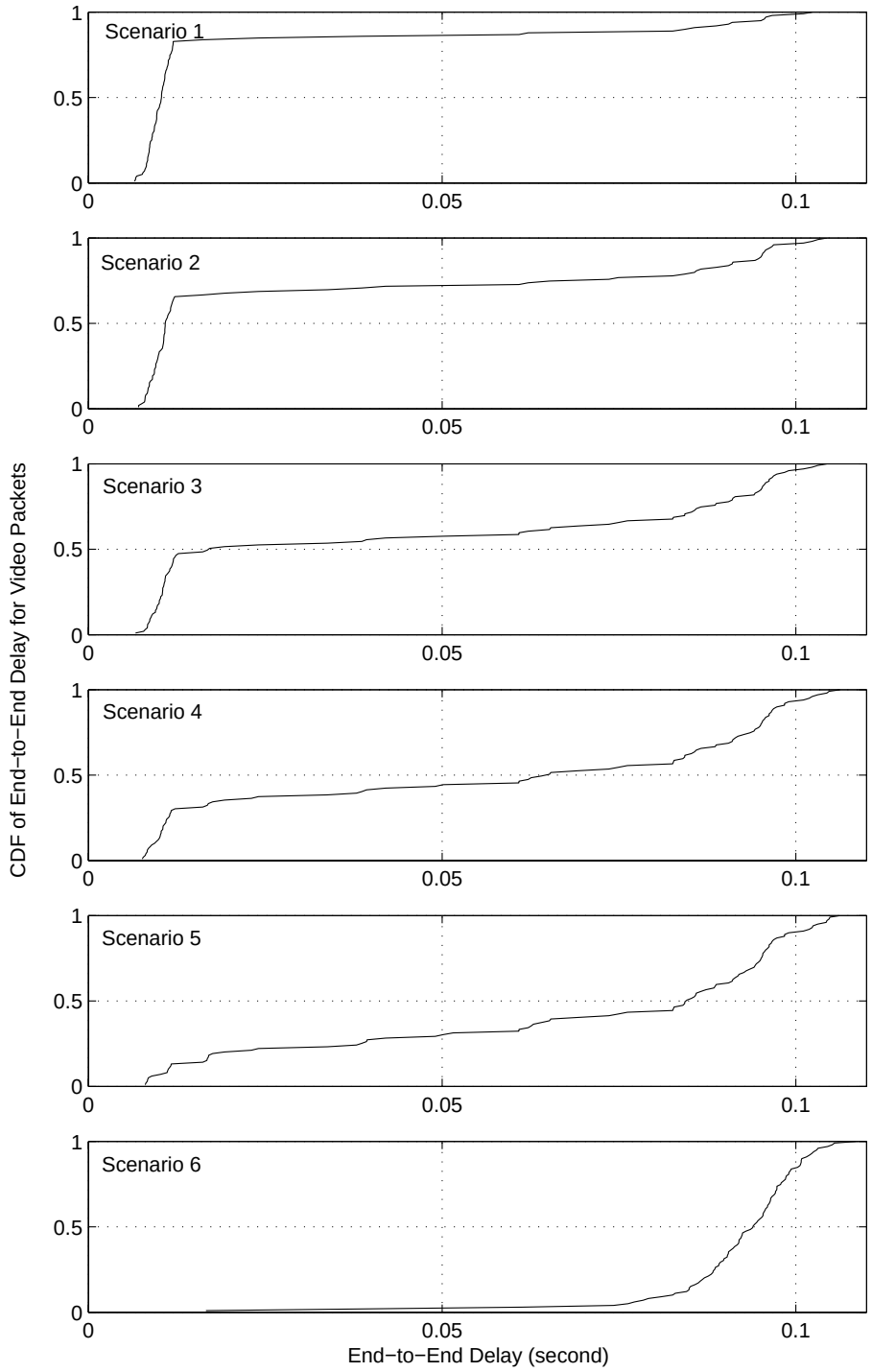


Figure 10: Cumulative Distribution Function (CDF) of video packet delay for WFQ.

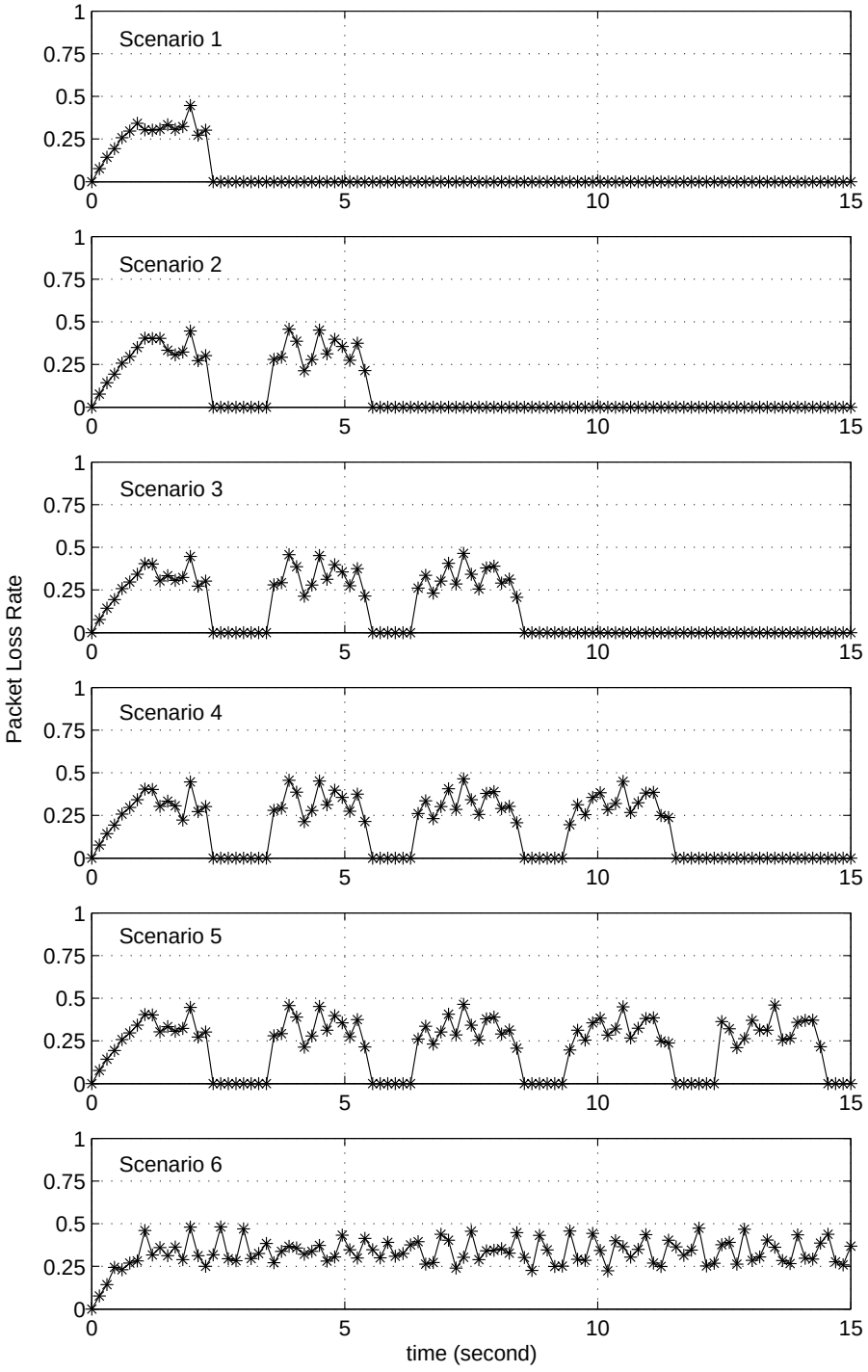


Figure 11: Packet loss rate of video flow for WFQ.

Table 2: Loss Threshold for Video Flow at Various Routers

Router	*Loss Rate (%)*	*Loss Threshold (%)*
R_1	0	12
R_2	9.1	12
R_3	10	3.2
R_4	N/A	N/A
R_5	N/A	N/A
R_6	N/A	N/A
R_7	N/A	N/A

7.1 Basic UPTA

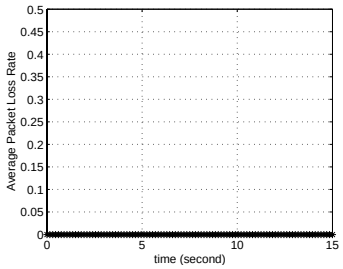
We then ran a simulation (under Scenario 6) for the network with B-UPTA implemented. We measured packet loss rates for the video flow at each router. The results are shown in Figure 12. The figure shows that the average packet loss rate as monitored at each individual router (Figures 12(a)–12(g)) is less than the threshold of 12%. Therefore, B-UPTA fails to detect UPT in this case. However, the cumulative packet loss rate (as seen by Video Dst) is about 33.5% (Figure 12(h)). Hence, the video is unintelligible in the video receiver's perspective. The simulation results are consistent with the analysis results given in Table 1.

7.2 Partial UPTA

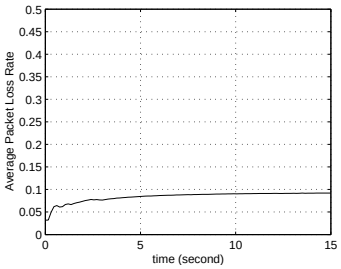
According to our analysis in Section 4.2, P-UPTA can detect UPT if the end-to-end packet loss rate (P_e) is greater than the loss threshold specified by the application. In our simulated network, the end-to-end packet loss rate P_e for the video application is 33.5% (see Table 1). Therefore, the video flow should be identified as U flow when P-UPTA is enforced. Based on Eq. (17), we calculate the loss threshold used at each router in Figure 1. The results are shown in Table 2.

From Table 2 we can see that UPT goes undetected at routers R_1 and R_2, because packet loss rate (for the video flow) is only 9.1% (less than the loss threshold of 12%). At router R_3, the loss threshold changes to 3.2%. UPT is detected at R_3 because the estimated loss rate is 10% at R_3, greater than the threshold of 3.2%. Figure 13 shows packet loss rates for the video flow measured at routers R_1–R_3. Packet loss rates at routers R_4–R_7 are not shown in the figure, because video packets are dropped completely at router R_3 (see Table 2). The figure shows that UPT from the video flow is not eliminated until router R_3 (where video packets are dropped with a probability of 1, see Figure 13(c)).

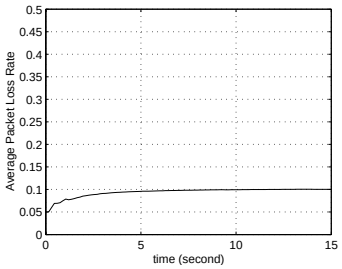
Table 3 shows TCP throughput improvement for all six scenarios with P-UPTA. In the table, simulation results are compared with theoretical values for TCP improvement factor (calculated based on Eq. (22)). The theoretical value represents the maximum TCP improvement factor we can achieve for each scenario. From Table 3 we can see that, with P-UPTA, TCP throughput is improved in all six scenarios. The improvement factor ρ is between 0.1–0.37, increasing with the increase of unintelligible ratio ϕ. On the other hand, as we can see in the table, the improvement factor based on simulation is much less than the theoretical value. For example, for Scenario 6, the theoretical value of ρ is 0.5 while the value from simulation is only 0.37. This is because P-UPTA cannot completely eliminate UPT in a network (That's why it is so called!). In the simulated network as shown in Figure 1, UPT is detected at router R_3. In other words, there is no UPT on links l_3–l_6. However, UPT is present on links l_1 and l_2 (see Table 2). Although useless video packets are eventually discarded by the network, they have wasted



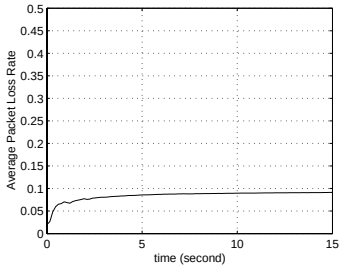
(a) Router1



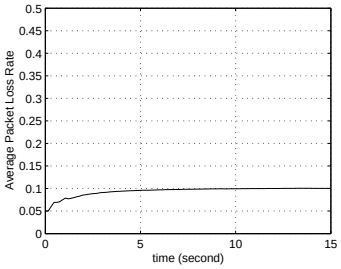
(b) Router2



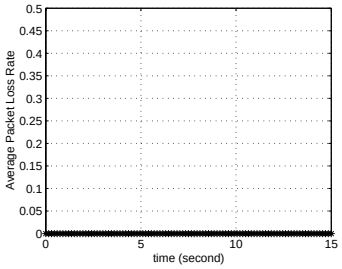
(c) Router3



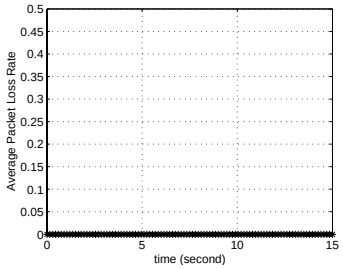
(d) Router4



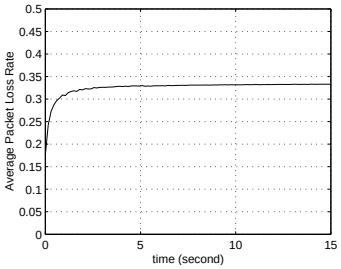
(e) Router5



(f) Router6

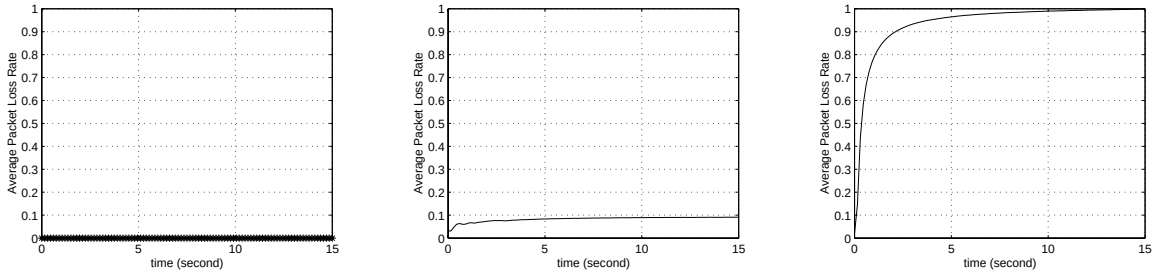


(g) Router7



(h) Cumulative loss rate (as seen by Video Dst)

Figure 12: Average loss rate of video packets at various routers under B-UPTA (Scenario 6).



(a) R1 (b) R2 (c) R3

Figure 13: Packet loss rate for video flow with P-UPTA (Scenario 6).

Table 3: TCP Throughput Improvement with P-UPTA

Scenario	*Unintelligible Ratio (ϕ)*	*Theoretical Improvement (ρ)*	*Simulation Result*
1	0.133	0.067	0.048
2	0.267	0.134	0.097
3	0.4	0.2	0.146
4	0.533	0.267	0.194
5	0.667	0.334	0.243
6	1.0	0.5	0.365

bandwidth on links l_1 and l_2. The TCP throughput is limited to 1.6 Mbps in Scenario 6, as l_2 becomes the bottleneck link in the network.

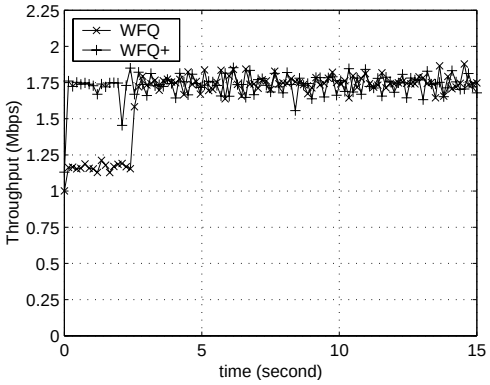
7.3 Centralised UPTA

Finally, we ran a set of simulations with C-UPTA implemented. We compare the performance of C-UPTA against WFQ.

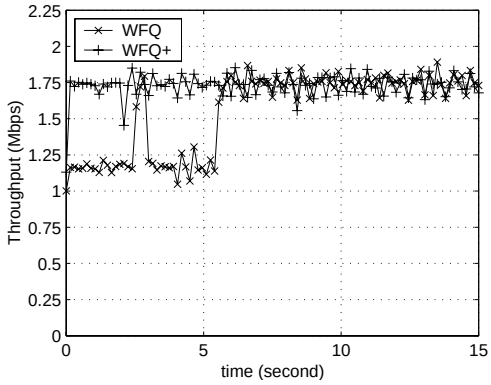
7.3.1 TCP Throughput

To analyse TCP throughput dynamics in different intervals, we have used OPNET's "bucket" option to plot throughput in consecutive buckets of 100 packets received at the destination. This short-term throughput analysis allows us to observe any variation of TCP throughput over short intervals. Figure 14 compares short-term TCP throughput with and without C-UPTA, for all six scenarios. It is evident, that with our proposed C-UPTA in place, the TCP source (S_1) receives higher throughput during U intervals. C-UPTA effectively recovers bandwidth that would have been otherwise wasted in UPT, and distributes the recovered bandwidth to other competing connections. We can also see that C-UPTA has no effect on TCP throughput during I intervals, substantiating the fact that C-UPTA can successfully switch to "normal" processing mode when there is no UPT in the system.

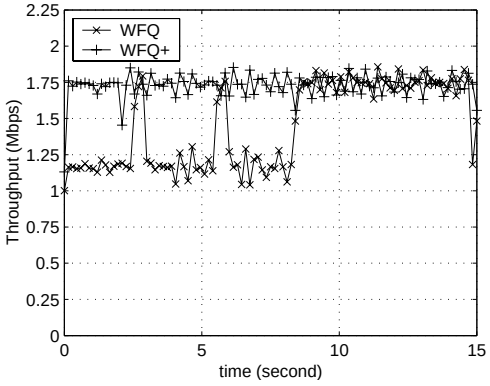
Table 4 shows the average TCP throughput over the entire simulation duration (15 seconds). In this



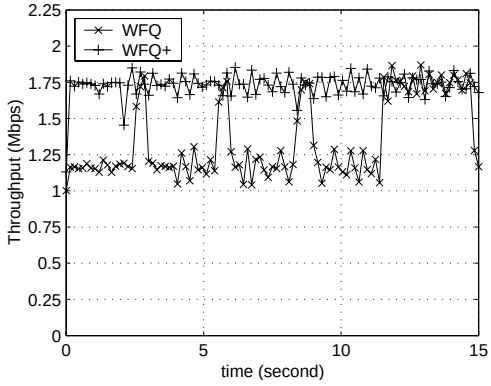
(a) Scenario 1



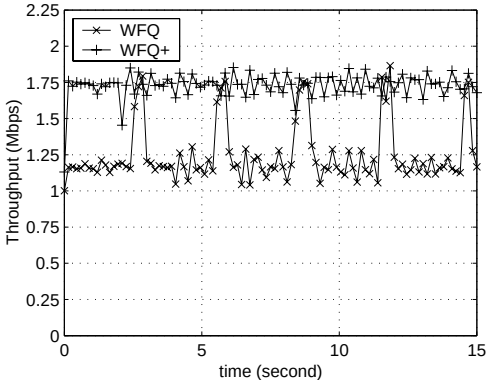
(b) Scenario 2



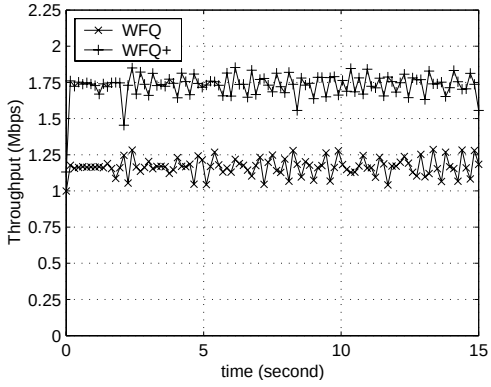
(c) Scenario 3



(d) Scenario 4



(e) Scenario 5



(f) Scenario 6

Figure 14: Comparison of short-term TCP throughput, with and without C-UPTA.

Table 4: Comparison of TCP Throughput (in Mbps) with and without C-UPTA

Scenario	*WFQ*	*WFQ+*
1	1.62	1.72
2	1.52	1.72
3	1.44	1.72
4	1.37	1.72
5	1.31	1.72
6	1.16	1.72

Table 5: TCP Throughput Improvement with C-UPTA

Scenario	*Unintelligible Ratio (ϕ)*	*Theoretical Improvement (ρ)*	*Simulation Result*
1	0.133	0.067	0.062
2	0.267	0.134	0.132
3	0.4	0.2	0.194
4	0.533	0.267	0.256
5	0.667	0.334	0.313
6	1.0	0.5	0.483

table, WFQ+ represents incorporation of the C-UPTA in WFQ. A quick glance through the table reveals that incorporation of C-UPTA increases TCP throughput in all scenarios. The magnitude of throughput increase is a direct function of the amount of bandwidth waste incurred in each scenario.

Using Eq. (22), we calculate throughput improvement factor ρ for the TCP flow under all six scenarios in Table 5. We also compare ρ derived from Eq. (22) with that obtained from simulation in the table. As shown in Table 5, for all six scenarios, the TCP throughput is improved when C-UPTA is implemented. For example, in Scenario 6, the TCP throughput is improved by a factor of nearly 0.5 (or 50%). On the other hand, the simulation results are very close to theoretical values, implying most UPT has been successfully eliminated.

Finally, Table 6 compares application level throughput (excluding TCP/IP header and any retransmissions) achieved by the three TCP sources in all six scenarios, with and without C-UPTA. In consistence with TCP throughput, we can see that C-UPTA also increases application throughput in all six scenarios.

7.3.2 File Download Time

To observe the impact of C-UPTA on file download time, we have simulated file transfers (between TCP source and destination) of three different file sizes, each representing a different type of object on the Web. A 15 KB file represents HTML Web pages; a 150 KB file represents compressed image objects; and finally, a 1.5 MB file represents compressed video objects or clips [3]. Table 7 shows improvements in file download times achieved by the TCP source in Scenario 1.

We can see that C-UPTA significantly reduces file download time for all file sizes. For Scenario 1, percentage savings in download times are more significant for small file size. For example, with C-UPTA incorporated in WFQ, file download time is reduced by around 32% for 15 KB file and 150 KB file, but

Table 6: Comparison of Application Throughput (in Mbps) with and without C-UPTA

Scenario	*WFQ*	*WFQ+*
1	1.26	1.34
2	1.18	1.33
3	1.12	1.33
4	1.07	1.33
5	1.02	1.33
6	0.91	1.33

Table 7: Comparison of Download Time (in seconds) with and without C-UPTA (Scenario 1)

File Size	*Download Time (seconds)*	
	WFQ	WFQ+
15 KB	0.132	0.090
150 KB	1.342	0.902
1.5 MB	9.518	9.023

only 5.2% for 1.5 MB file. This is because, with Scenario 1, we only have a modest amount of bandwidth waste in the network.

Figure 15 graphically compares file download times for 1.5 MB file under all six scenarios. We can see that with more UPT and bandwidth waste present in the system, C-UPTA can reduce file download time more significantly, both in absolute and percentage scales. For example, under Scenario 6, with C-UPTA incorporated in WFQ, download time for 1.5 MB file is reduced from about 13 to a mere 9 seconds, a saving of 4 seconds or 31%. Savings of such magnitudes are useful not only in traditional computing and communication environments, but also in mobile and wireless environments where battery power consumption is directly affected by the connection times.

7.3.3 Impact on Video Intelligibility

UPTA algorithm aims to recover bandwidth wasted by multimedia connections (during U intervals) without inflicting any *further* damage to the overall intelligibility of the communications. Using intelligibility index (defined in [16]), Table 8 shows the overall intelligibility of the received video at the destination in all six scenarios. As we can see, C-UPTA has very little effect on the overall intelligibility of the video connection. For all six scenarios, the difference in intelligibility index is less than 3%. This result substantiates that C-UPTA is capable of improving TCP performance without inflicting noticeable damage on multimedia.

The good performance of C-UPTA with respect to intelligibility index (Table 8) is due to its success in maintaining the number of intelligible frames, and reducing only unintelligible frames. The distribution of intelligible and unintelligible frames under all six scenarios is shown in Figure 16. Figure 17 shows the quality of some of the unintelligible frames received at Video Dst during a U interval in Scenario 5 (without C-UPTA). Clearly, by not receiving these frames (when C-UPTA is implemented), the users would not lose too much.

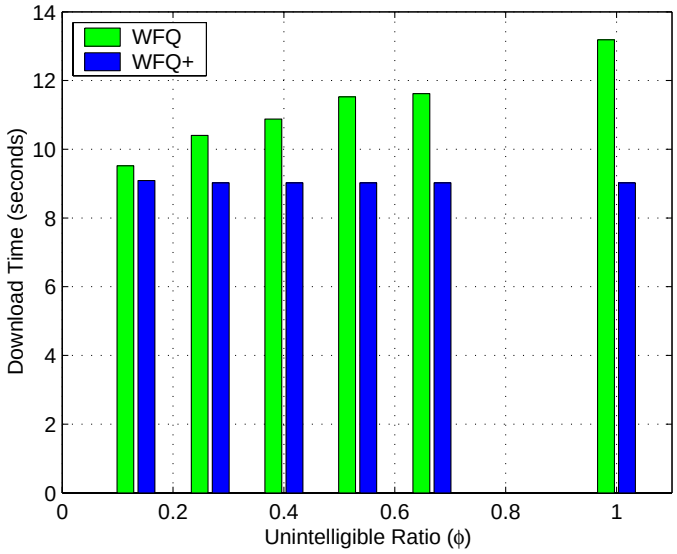
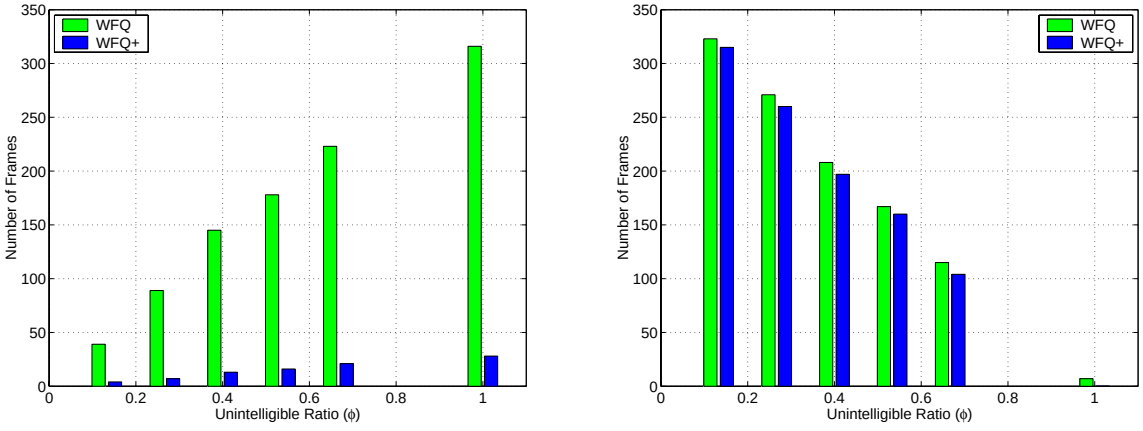


Figure 15: Comparison of download time with and without C-UPTA.

Table 8: Comparison of Video Intelligibility

Scenario	*Intelligibility Index*	
	WFQ	WFQ+
1	0.8613	0.8440
2	0.7227	0.7048
3	0.5547	0.5383
4	0.4453	0.4328
5	0.3067	0.2973
6	0.0187	0.0



(a) Unintelligible frames

(b) Intelligible frames

Figure 16: Distribution of unintelligible and intelligible frames for WFQ.

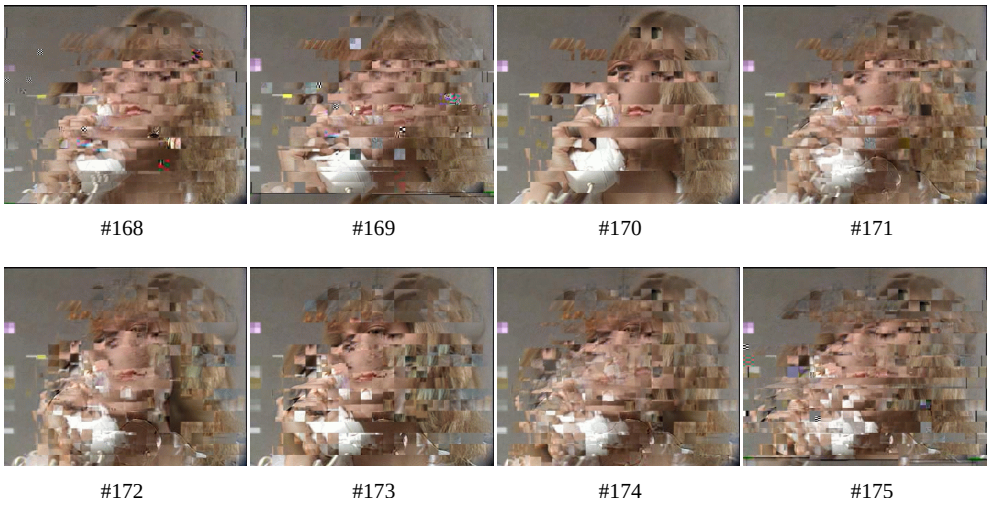


Figure 17: Snapshots of unintelligible video frames during U intervals.

Table 9: Throughput Fairness

Scenario	*Fairness Index in U Interval*		*Fairness Index in I Interval*	
	WFQ	WFQ+	WFQ	WFQ+
1	0.9999	0.9999	0.9999	0.9999
2	0.9998	0.9997	1.0000	1.0000
3	0.9999	0.9997	1.0000	1.0000
4	0.9997	0.9996	1.0000	1.0000
5	0.9999	0.9998	1.0000	1.0000
6	0.9999	0.9997	1.0000	1.0000

7.3.4 Throughput Fairness

We compute fairness index separately for U intervals and I intervals. In U intervals, the video flow is not included in fairness calculation, as it is not supposed to receive fair throughput (and waste bandwidth) under C-UPTA. However, during I intervals, all flows are considered. Table 9 shows throughput fairness achieved with and without C-UPTA under all six scenarios. It is encouraging to see that C-UPTA can maintain the same level of fairness obtained with WFQ. In other words, the implementation of C-UPTA into fair algorithms does not have any adverse effect on the fairness performance of the algorithms.

7.4 Discussion

We have evaluated the performance of three different UPTA enforcement schemes (i.e., B-UPTA, P-UPTA, and C-UPTA) in the simulated network. To summarise, we plot TCP improvement factor ρ achieved with different approaches in Figure 18. As shown in the figure, no improvement achieved with B-UPTA. This means, in networks with multiple congested links, UPT may go undetected if B-UPTA is implemented. P-UPTA represents an improvement over B-UPTA, with some improvement in TCP throughput. However, the improvement factor is far less than the theoretical value. This implies that UPT has not been eliminated completely. There is still bandwidth waste due to useless packet transmission

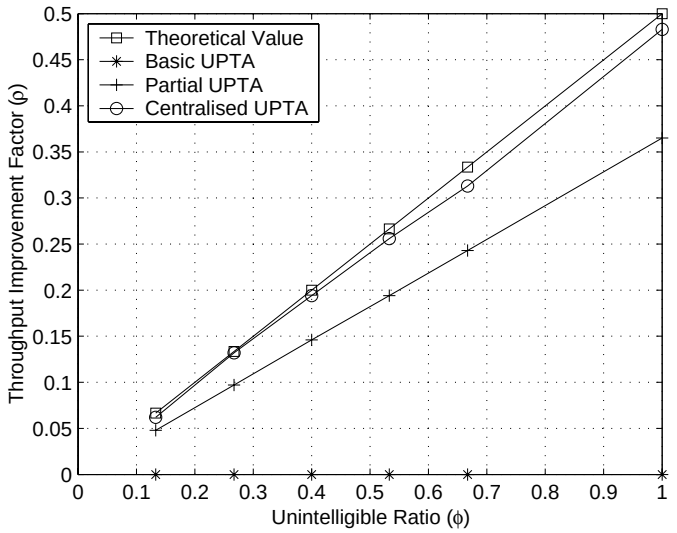


Figure 18: TCP throughput improvement factor against unintelligible ratio.

Table 10: Comparison of Various UPTA Enforcement Schemes

Feature	*Removal of UPT*	*TCP Throughput Improvement*	*Feedback Mechanism*	*Complexity*
B-UPTA	No	No	No	Low
P-UPTA	Partial	Less than Max.	No	Medium
C-UPTA	Complete	Maximum	Yes	High

with P-UPTA. With C-UPTA, the improvement factor is very close to the theoretical value. This proves that C-UPTA yields the best result in terms of UPT avoidance.

Table 10 compares features and performance of the three different UPTA enforcement schemes. Obviously, there is a trade-off between performance and complexity. The performance increases in order of B-UPTA, P-UPTA, C-UPTA. However, complexity also increases in the same order. B-UPTA has minimum protocol overhead, while C-UPTA has the maximum overhead. In C-UPTA, a feedback mechanism (BFD) is needed to determine global fairshare. With B-UPTA and P-UPTA, no extra protocol is needed. P-UPTA is a little more complicated than B-UPTA (as loss threshold q needs to be updated by each router on the end-to-end path), but less complicated than C-UPTA.

8 Conclusion

We have investigated the UPT problem with multiple congested links in this paper. In networks with multiple congested links, the end-to-end packet loss rate of a multimedia flow may exceed the loss threshold while local loss rate at each router *en route* is less than the loss threshold. This complicates UPT avoidance in networks with multiple congested links. Using WFQ as an example, we discussed three different UPTA enforcement schemes — B-UPTA, P-UPTA and C-UPTA. B-UPTA is enforced independently at each router. The algorithm is the same as that used with single congested link (described in [16]). P-UPTA extends B-UPTA by introducing variable loss threshold at each router. The loss

threshold is decreased at each router based on local packet loss rate at the router. As the loss threshold is updated at each router, it is possible for P-UPTA to detect UPT somewhere in the network, provided the end-to-end packet loss rate is greater than the threshold specified by an application. C-UPTA introduces a protocol called Bottleneck Fairshare Discovery (BFD) which enables it to eliminate UPT completely at the ingress router.

Using OPNET simulation and an MPEG-2 video clip, we have evaluated the performance of the three different UPTA enforcement schemes. For our simulated network, B-UPTA fails to detect UPT. As a result, the TCP source sees no improvement in throughput even after UPTA is implemented. Although P-UPTA can eventually detect UPT, bandwidth has been wasted on links before the router where useless packets are dropped. Consequently, only modest TCP throughput improvement is achieved with P-UPTA. C-UPTA always drops useless packets at the network edge (ingress router). This effectively avoids the problem with P-UPTA. As a result, more bandwidth is made available to the TCP flow. Our simulation results show that, with C-UPTA, the achieved TCP throughput improvement is very close to the maximum theoretical value. This indicates that useless packets have been effectively prevented from entering the network.

There remains two open issues. We have assumed that routers have the knowledge of packet loss rate thresholds (for intelligible communication) for all multimedia applications. This threshold information is used by our UPTA algorithm to detect the unintelligible intervals where UPT occurs. How applications can pass this information to the network remains an open issue, and has not been explicitly addressed in this paper. Another issue not addressed in this paper is the possibility of multiple multimedia flows a large IP network. In situations where multiple multimedia flows sharing the same bottleneck link are marked as U flows, all these flows will be cut off when UPTA is enforced. An undesirable effect of this is that all multimedia flows become useless, and the bottleneck link is under-utilised. We are currently investigating this issue.

References

[1] Dimitri Bertsekas and Robert Gallager. *Data Networks*. Prentice Hall, 2nd edition, 1992.

[2] Zhiruo Cao, Zheng Wang, and Ellen Zegura. Rainbow Fair Queueing: Fair Bandwidth Sharing without Per-Flow State. In *Proceedings of IEEE INFOCOM 2000*, volume 2, pages 922–931, Tel Aviv, Israel, March 2000.

[3] N. Chapman and J. Chapman. *Digital Multimedia*. John Wiley & Sons, 2000.

[4] Wu-chang Feng. *Improving Internet Congestion Control and Queue Management Algorithms*. PhD thesis, University of Michigan, 1999.

[5] Wu-chang Feng, Dilip D. Kandlur, Debanjan Saha, and Kang G. Shin. BLUE: A New Class of Active Queue Management Algorithms. Technical Report CSE-TR-387-99, University of Michigan, Ann Arbor, MI 48105, April 1999.

[6] F. Fluckiger. *Understanding Networked Multimedia: Applications and Technology*. Prentice Hall, 1995.

[7] Cisco Systems Inc. Advanced QoS Services for the Intelligent Internet. http://www.cisco.com/warp/public/cc/pd/iosw/index.shtml.

[8] J. M. Jaffe. Bottleneck Flow Control. *IEEE Transactions on Communications*, COM-29(7):954–962, July 1981.

[9] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison Wesley, 2001.

[10] Dong Lin and R. Morris. Dynamics of Random Early Detection. In *Proceedings of ACM SIGCOMM*, pages 127–137, October 1997.

[11] Nortel Networks. http://www.nortelnetworks.com/products/01/passport/wan/p5430/doclib.html.

[12] Rong Pan, Balaji Prabhakar, and Konstantinos Psounis. CHOKe, A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation. In *Proceedings of IEEE INFOCOM 2000*, volume 2, pages 942–951, Tel Aviv, Israel, March 2000.

[13] A. Parekh and R. Gallager. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: the Single-Node Case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[14] I. Stoica, S. Shenker, and H. Zhang. Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks. In *Proceedings of ACM SIGCOMM*, pages 118–130, Vancouver, Canada, September 1998.

[15] OPNET Technologies. OPNET Modeler 7.0.B. http://www.opnet.com/.

[16] Jim Wu and Mahbub Hassan. The Issue of Useless Packet Transmission for Multimedia over the Internet. *Computer Communications*, accepted for publication.