UARA in Edge Routers: An Effective Approach to User Fairness and Traffic Shaping

Accepted Version (postprint)

This is the accepted version of the following article: Masood Khosroshahy (2011), *"UARA in Edge Routers: An Effective Approach to User Fairness and Traffic Shaping"*, International Journal of Communication Systems (Wiley), 25: 169–184, which has been published in final form at DOI: 10.1002/dac.1262.

INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS
Int. J. Commun. Syst. 2010; **00**:1–10 *Prepared using* dacauth.cls *[Version: 2002/09/18 v1.01]*

UARA in Edge Routers: An Effective Approach to User Fairness and Traffic Shaping

Masood Khosroshahy*†

ECE Dept., Concordia University, Montreal, Canada

SUMMARY

The ever-increasing share of the peer-to-peer (P2P) traffic flowing in the Internet has unleashed new challenges to quality of service provisioning. Striving to accommodate the rise of P2P traffic or to curb its growth has led to many schemes being proposed: P2P caches, P2P filters, ALTO mechanisms and re-ECN. In this paper, we propose a scheme named "UARA: **U**ser/**A**pplication-aware **R**ED-based **A**QM" which has a better perspective on the problem: UARA is proposed to be implemented at the edge routers providing real-time near-end-user traffic shaping and congestion avoidance. UARA closes the loopholes exploited by the P2P traffic by bringing under control the P2P users who open and use numerous simultaneous connections. In congestion times, UARA monitors the flows of each user and caps the bandwidth used by "power users" which leads to the fair usage of network resources. While doing so, UARA also prioritizes the real-time traffic of each user, further enhancing the average user quality of experience. UARA hence centralizes three important functionalities at the edge routers: 1) congestion avoidance; 2) providing user fairness; 3) prioritizing real-time traffic. Simulation results indicate that average user quality of experience is significantly improved in congestion times with UARA at the edge routers.

KEY WORDS: bandwidth usage fairness; flow rate fairness; peer-to-peer traffic; traffic optimization; traffic shaping; active queue management

1. INTRODUCTION

According to many studies [1–4], P2P traffic has consolidated its position as the dominant component in the traffic mix flowing in the Internet today and has caused serious congestion problems. This reality has started a new wave of research efforts regarding the management of P2P traffic within networks that are suitable for client-server traffic type in terms of both network design and employed congestion control mechanisms. Some network operators have tried to contain the share of P2P traffic by various techniques such as traffic throttling. Increasingly, whether due to lawsuits brought by proponents of P2P technologies or otherwise, the telecommunication industry comes to the realization that P2P technology may have to be accommodated rather than being fought. Building on this realization, in this paper, we analyze the issues involved and propose a method to intelligently accommodate the

*Correspondence to: Office EV.010.235, 1515 St-Catherine Ouest, Montreal, Canada
†E-mail: m.kh@ieee.org

P2P traffic with minimal costs for network operators while keeping the P2P developers and end-users satisfied.

Striving to achieve "flow rate fairness" has been present in almost all queueing and congestion control mechanisms. This presence has taken the form of requiring that all applications use TCP or otherwise be TCP-friendly on the client side as well as implementing fair queueing schemes [5–7] or other flow-rate-based AQM schemes, on the router side. However, the goal of providing flow rate fairness has a reduced relevance in today's Internet. The idea of trying to provide *fairness* should be considered in the context of relationships between people and/or businesses (hereafter referred to as *users*). In the early days of Internet, providing flow rate fairness largely meant achieving fairness between users as well. The nature of applications run in the Internet, in terms of number and duration of flows, was far more harmonious than what is seen in today's Internet. This meant that most queueing and congestion control mechanisms took care of the issue of providing fairness between *users*, just by providing fairness between *flow rates*. This approach has worked relatively well before the domination of the P2P applications. P2P applications are known to: 1) create numerous flows; and 2) use them in parallel for long periods of time. As a result, P2P applications have fully exploited the two loopholes present in the idea that *flow rate fairness* naturally leads to *fairness between users* and have gained a grossly-unfair advantage over other more traditional Internet applications like HTTP[†].

In this paper, we are proposing an Active Queue Management (AQM) scheme named "**U**ser/**A**pplication-aware **R**ED-based **A**QM" (or *UARA* for short) which is to be implemented at the edge routers. UARA is *user-aware*, since it has a user identification engine which tries to identify the users who send/receive the most traffic (regardless of number of TCP or UDP connections that they have). UARA is *application-aware*, since it has a traffic shaping engine which tries to find out the type of each traffic. UARA is *RED-based*, since it has a RED [8] core and uses RED's packet drop history (using the method employed in RED-PD [9] to find high-bandwidth flows) to find users/hosts who send or receive the most traffic. UARA is designed to respect the user choice as to what application should receive the best performance in times of congestion. Only if not communicated by the user, UARA tries to detect the type of application and favors applications with real-time needs. Furthermore, UARA gives precedence to the packets of all users who have used the network the least. Details of the scheme will be developed in the later sections of this paper.

The contribution of this paper is two-fold: a) introduction and evaluation of a new AQM mechanism (UARA) which significantly enhances user fairness at congestion times; b) introduction of a method for quantifying *User Quality of Experience* through normalization. This method has been used in evaluating UARA along with few other queueing schemes (this normalization method can be used in other similar evaluation studies).

The rest of this paper is organized as follows: Section 2 discusses current mechanisms to deal with the P2P traffic. In Section 3, we present the motivation behind our proposal along with all its architectural and implementation details. Section 4 is dedicated to presenting simulation results and performance analysis of our proposed mechanism. Section 5 provides a summary as well as some discussions on aspects that are often debated in the research community. We conclude in Section 6 by outlining our future work.

[†]This issue has been raised in [1, Sec.6] as well.

2. Current Mechanisms to Control the P2P Traffic

2.1. Caching Mechanism

Cache equipments incorporate large amounts of storage along with the software which prevent unnecessary flow of traffic out of the network. The content is locally cached and served on demand from a local entity. Cache equipments intercept download requests from inside the network, thereby reducing network operator's transit bandwidth requirements. Likewise, they intercept download requests from outside the network, thereby reducing bandwidth use and congestion on the local access network. Some field tests have been conducted demonstrating the effectiveness of the P2P caching approach. However, important legal concerns are attached to caching solutions as the cache provider may seem to be facilitating illegal exchange of copyrighted content in P2P networks. These legal concerns reduce the desirability of caching solutions [1, Sec.5.1.5]. Furthermore, the benefit of caching solutions is generally limited to popular contents which represent only a portion of the overall P2P traffic. Although P2P caches can still be considered part of the solution to address the problem posed by the P2P traffic, their limited effectiveness and legal concerns have resulted in rare actual deployments in current networks. In comparison, UARA, our proposal, addresses the issues of user fairness and real-time traffic prioritization; P2P caches do not address directly any of these issues.

2.2. Traffic Shaper

Classification of applications in the Internet remains a challenging task. In the case of BitTorrent, for example, clients can operate on any port number including port numbers of traditional Internet applications like HTTP's 80. This behavior renders port-based traffic identification methods useless and has given rise to payload-based identification methods [10]. However, BitTorrent client applications are now employing payload encryption methods to avoid detection. The encryption is effective due to the fact that payload string matching misses all encrypted P2P packets. BitTorrent clients can use "Message Stream Encryption" [11] which is an encapsulation protocol, designed to provide a random-looking header and payload.

Use of transport-layer port numbers and packet payload inspection are increasingly unreliable approaches to classify applications [12]. Applications often use randomly-selected port numbers and increasingly implement some sort of payload encryption to remain in the shadows and undetected by network profilers and traffic shapers. Furthermore, the sheer number of new applications (including P2P applications) and the fact that most of them are poorly-documented present an insurmountable challenge to reverse engineering efforts in designing traffic classifiers based on payload inspection approaches. As a result, new approaches have been proposed to classify traffic based on flow connection patterns and without inspection of the payload [13–15].

The primary reason for current interest in identification of P2P traffic is improving the performance of traffic shapers deployed in the core network. While their performance has improved significantly in recent years [16], the P2P traffic filters have had to play catch-up with the latest trends in the P2P development, remaining one step behind most of the time. Furthermore, the current practice of deployment of traffic shapers in the core network has little bearing on the congestion situation in specific places on the edges. This is due to lack of real-time information in the core network about the congestion situation at the edges. UARA, on the other hand, is a solution that corrects both aspects: 1) it eliminates the incentive problem; therefore, there would be no benefit for the user to encrypt the traffic, in one way or another, in order to gain an advantage; 2) it is deployed at the edge routers where the congestion problem is mostly present and is naturally aware of the real-time congestion situation.

2.3. ALTO

ISP-assisted "Application Layer Traffic Optimization (ALTO)" mechanisms [2, 3, 17] revolve around the idea of cooperation between ISPs and P2P software developers, while preserving each party's interest to some extent in the process: lower congestion/cost for ISPs and acceptable speed/performance for P2P applications.

Being a prominent example of ALTO mechanisms, P4P [2] is a framework based on which end-host applications and network providers have an explicit communication. As a result, knowing the end-users status (location in the network topology, etc.), network operator's policies and various link-specific costs, applications can participate in the resource optimization of the operator's network.

Designing ALTO mechanisms is still considered a hot research topic, with many mechanisms still being proposed and refined [4, 18]. Apart from ISP-assisted ALTO mechanisms, there are a number of proposals that try to estimate the topology information without any help from the network operator; rather, they use distributed measurements. They aim to build distributed network coordinate systems and path evaluation services. These unassisted mechanisms do not seem to be able to go far enough in addressing the P2P traffic problem however, mostly due to impossibility of deriving information regarding link-specific costs and inter-ISP peering contracts.

Despite being part of the solution to address the P2P traffic problems, ALTO mechanisms have inherent incentive problems as their adoption will generally lead to reduced performance on the client side, as the resource consumption is optimized in the ISP's network. P2P software developers have therefore little incentive in general to incorporate these schemes in their peer selection algorithms. On the other hand, ISP-assisted ALTO schemes may only be open to commercial uses; therefore, unavailable to the bulk of the current P2P traffic. Finally, ALTO schemes do not address issues such as upholding user fairness and real-time traffic prioritization; these are addressed by UARA, as pointed out earlier.

2.4. re-ECN

Recently, Bob Briscoe has extensively discussed the shortcomings of the *flow-rate fairness* paradigm [19–21] and has designed a scheme called *re-ECN*[‡] [22] to achieve what he refers to as "cost fairness". The re-ECN scheme is a concrete proposal as to how price can be put on the congestion that users cause, achieving cost fairness as a result. Re-ECN aims to close the loopholes exploited by the P2P traffic; therefore, *fair* use of the network by all traffic types including the P2P traffic is supposed to follow naturally after deployment of re-ECN. In what follows, we briefly explain re-ECN, Briscoe's solution to the Internet's congestion problem.

As we know already, ECN allows the observation of the path congestion at the receiver side. The receiver, in turn, informs the sender, through a TCP flag, about the experienced congestion on the path. Finally, the sender *can* take action to correct the situation by reducing its rate. The obvious problem is that the network is at the mercy of the sender and cannot force the sender to reduce its rate; only an egress dropper, just before reaching the receiver, can stop the rogue flow by monitoring the ECN-CE marks and consequent sender rate response to the congestion indication, but the congestion damage is already done in the network[§]. Re-ECN corrects this situation by exposing the congestion information

[‡]Recently, the term "Congestion Exposure (ConEx)" has been adopted as a replacement for *re-ECN*.

[§]Another scenario would be that the receiver does not report at all the received congestion marks to the sender, hoping to maximize its reception rate.

at the sender's side, so an ingress policer can take action before any damage is done in the network. We refer the reader to [22] for a thorough explanation of the mechanics of re-ECN.

The main difference between *user fairness*, as we refer to it in the next section in the context of UARA's objectives, and *cost fairness* is as follows: user fairness is merely fairness between users of the network with regard to the overall traffic that they send or receive. Our implementation of user fairness scheme does not need any changes neither to IP nor to any other protocols; it only needs accurate identification of *users* of flows at the edge routers. On the other hand, cost fairness is built on the notion of putting price on the congestion that users cause in the network and its implementation entails changes to TCP/IP along with addition of other policing/dropping devices to the network.

Although we believe that the Internet-wide implementation of re-ECN might lead to a *fairer* Internet, there will be still some important concerns left unaddressed. Namely, with re-ECN, user's knowledge of congestion caused per application is assumed, i.e., the user has to know which application on their computer is consuming their congestion quota the quickest. While this knowledge might be derivable for the technical-savvy few, it is not very comprehensible for the majority of non-technical users. Another issue would be that, based on the fixed congestion allowance per user, a P2P application is still in an advantage compared to other traditional Internet applications. This is due to the fact that the P2P application has numerous relatively low-bandwidth flows which means that these flows have less chance of being ECN-CE marked compared to single flows of other application types. These concerns cast some shadow on the usefulness of re-ECN in terms of improvement in the fairness situation with regard to different traffic types. While re-ECN is a step in the right direction to improve the fairness situation, it definitely lacks the real-time traffic prioritization and congestion avoidance functionality that comes as an integrated solution with UARA.

3. Our Proposal: UARA

3.1. Motivation

In the current networks, few P2P "power users" are able to dominate and grab a grossly-unfair share of the network resources, leading to severe congestion problems. This has led to the situation in which ISPs see little business incentive to upgrade their network and increase capacity (and relieve that congestion), since most of the increase in capacity will be consumed by those few customers yet again and the majority of customers do not see any proportionate benefit to pay more and cover the upgrade costs [23, 24]. ISPs have so far tried to deal with this dilemma in two ways: 1) detect the P2P traffic and throttle it; 2) impose volume caps on users' bandwidth usage. We believe that both these solutions have fundamental problems: 1) detection of any particular application and throttling it goes against net neutrality laws in many countries, not to mention the increasing ineffectiveness of the employed P2P traffic detection mechanisms; 2) while imposing volume caps on customers could solve the congestion problem, it is far from an optimal solution, leaving many parts of the access network under-utilized. This under-utilization can be very severe if the operator wants to avoid the congestion altogether by setting a very conservative volume cap on all customers.

It is known that congestion is more likely to occur in the access network (near edge routers) compared to the core network, since capacity is more limited due to upgrade costs increasing exponentially as we move towards the edges. As UARA is deployed in edge routers, it thus solves a real and present congestion problem. This is not true with P2P traffic shapers deployed in the core network which act based on static policies and, unaware of the congestion situation in the network, interfere

without being sure that their intervention is even needed. P2P caching solutions and ALTO mechanisms are also effective methods to some extent to optimize the P2P traffic and lower its traffic volume, however, they solely concentrate on the P2P traffic; therefore, they provide neither user fairness nor real-time congestion control. Re-ECN scheme, on the other hand, seems to be a promising solution to improve the fairness situation in the current Internet, however, its implementation is within reach far in the future due to the significant Internet-wide changes that it requires. Finally, note that UARA can be considered a complementary solution to all the mentioned mechanisms in Section 2.

3.2. Basic UARA and its Extensions

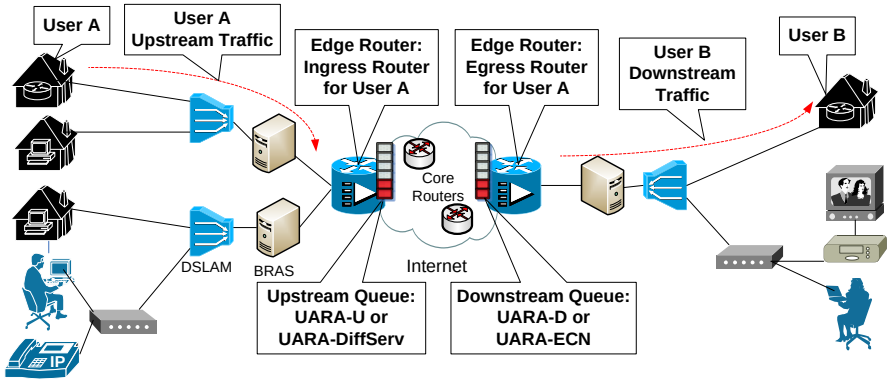


Figure 1: Simplified view of a DSL-based access network: Edge/Ingress/Egress routers and queues

As available capabilities differ from one network to another, we are proposing four editions of UARA. The following factors influence the design of UARA: 1) whether we consider the *ingress router* or the *egress router*; 2) whether or not we have ECN capability present in the network; 3) whether or not we have a DiffServ domain in the core network. Considering these factors, we can have different schemes implemented in the routers, in each direction (upstream or downstream) (cf. Figure 1):

- Ingress router (upstream traffic) without DiffServ domain: UARA-U scheme.
- Ingress router (upstream traffic) with DiffServ domain: UARA-DiffServ scheme.
- Egress router (downstream traffic) without ECN functionality: UARA-D scheme.
- Egress router (downstream traffic) with ECN functionality: UARA-ECN scheme.

Figure 1 depicts an example scenario regarding the position of different elements, including the edge routers, in order to aid the explanation of concepts in this paper. The arrangement of network elements could be different in any given access network, however, the proposed approach in this paper should be able to be adapted to suit any access network design.

For the UARA scheme that is implemented in the ingress router (UARA-U or UARA-DiffServ), the router identifies the user based on the packet's source IP address; this means that the ingress router monitors/controls the upstream traffic of the sender. On the other hand, for the UARA scheme that is implemented in the egress router (UARA-D or UARA-ECN), the router identifies the user based on the packet's destination IP address; this means that the egress router monitors/controls the downstream traffic of the receiver. Note that in the downstream, UARA controls all traffics that are received from

all sources destined to the receivers served by the edge router, leading to upholding fairness between these receivers.

In this paper, we present the architecture, implementation and simulation results of UARA-U. The other three editions of UARA, i.e., UARA-DiffServ, UARA-D and UARA-ECN, are subjects of our future work, as explained in Section 6.

3.3. UARA-U: Architecture

UARA-U, or UARA-Upstream, is a basic version of UARA that can be implemented in the upstream direction. UARA-U implements three functionalities: 1) congestion avoidance (the AQM functionality); 2) providing user fairness; 3) prioritizing real-time traffic. In designing an AQM scheme fulfilling the three aforementioned functionalities, we have been inspired by the design of RED-PD AQM scheme [9]. Schematic view of UARA-U's architecture, which is similar to that of RED-PD, is depicted in Figure 2. Although UARA-U has a complicated architecture, most of its components operate in parallel to forwarding path, i.e., they do not introduce a noticeable overhead. In this regard, UARA-U closely resembles RED-PD, in terms of the overhead introduced for processing packets in the forwarding fast path.

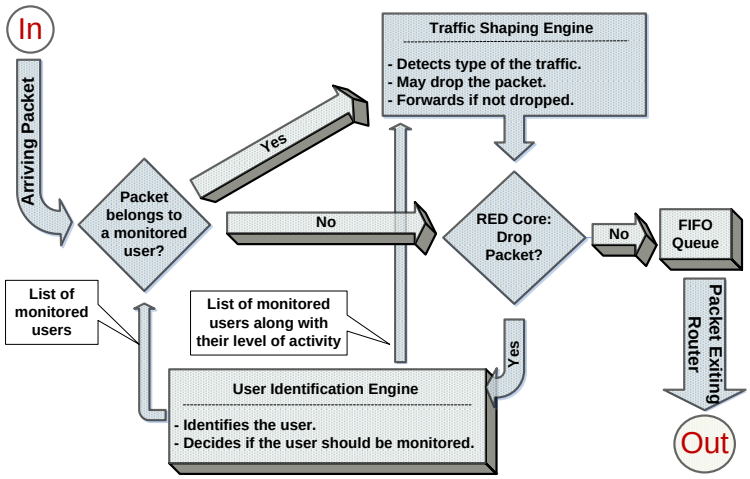


Figure 2: Schematic view of UARA-U's architecture

We briefly explain how the two engines function in UARA-U: When a packet arrives at the router, UARA-U checks whether or not the user who has sent the packet is a *monitored* user, i.e., it has a history of using too much bandwidth. If the user is not *monitored*, the packet is sent to the *RED core*. If the packet is not dropped by the RED algorithm, then the packet is placed in the FIFO queue to be sent to its next hop in its path. When the router is lightly-loaded, this is the procedure that most packets will go through and is equal to RED-PD algorithm, in terms of processing overhead.

In the procedure above, if the packet is dropped by the RED algorithm in the RED core, it will be sent to the user identification engine. In the user identification engine, RED's packet drop history is used to identify the "bandwidth hogs", i.e., users who send too much traffic. Finally, the information on monitored users and their activity levels (how much traffic they have sent) is sent to the respective

blocks of the UARA-U scheme for use in the processing of the future packets from the same user.

When a packet from a monitored user arrives, the packet is sent to the traffic shaping engine (note that when this happens, the router is necessarily congested.). The engine looks at first at the Type of Service (ToS) byte of the IP packet (more precisely, the six-bit Differentiated Services field within the ToS byte) to see if the received packet has Differentiated Services field set. If set, the engine will not try to identify the type of traffic and will respect the Differentiated Services choice[¶]. If not set, the engine tries to identify the traffic type (VoIP, P2P, etc.). After this phase, the engine knows what should be the drop probability of the current packet. This drop probability depends both on the activity level of the user and on the priority of the traffic type. If the user has been identified as a user who sends too much traffic (as communicated by the user identification engine), the drop probability will be set to a high value. If the packet is identified to be from a critical application with real-time needs, the drop probability will be lowered a bit, otherwise, the drop probability will remain high. The packet is then dropped with this drop probability. If the packet survives, it will be forwarded to the RED core. In closing, we emphasize that the storage and processing of drop history have insignificant overheads, as the user identification engine does not run in the forwarding fast path.

3.4. UARA-U: Implementation

As the functional architecture of RED-PD is similar to that of UARA-U, we have taken advantage of the existing implementation of RED-PD to implement UARA-U in the NS-2 network simulator. The implementation of RED-PD in NS-2 could be modified to arrive at an AQM scheme behaving like UARA-U, as briefly explained hereafter (see [25] to check the modified code).

Implementation of RED-PD has two main parts: 1) per-packet operations coded in *C++*; 2) periodic operations to calculate per-flow target bandwidths (and per-flow drop probabilities) coded in *OTcl*. RED-PD uses a flow monitor, which is linked to the queue, to know the statistics regarding how many packets are dropped or forwarded. A *SrcDestFid* hash classifier is linked to the flow monitor which enables it to determine the flow to which an arriving packet belongs (i.e., the granularity of a flow is "Source address, Destination address and Flow ID"). As RED-PD estimates the flow throughput based on the drop history in the underlying RED queue, it can therefore have an estimate of the throughput of the flow based on the statistics periodically provided by the flow monitor. If a flow is determined to be high-bandwidth (i.e., too many drops in the underlying RED queue), it is monitored and a *target bandwidth* is associated with the flow. This target bandwidth, along with the *estimated current throughput*, is used to calculate a per-flow drop probability. An arriving packet is dropped by the probability associated with the flow to which it belongs; if survived, it will be delivered to the underlying RED queue.

As mentioned earlier, the implementation of RED-PD relies on an external module (the hash classifier associated with the flow monitor) to determine the flow to which any given packet belongs. This design decision has served us very well, as we only needed to define our own hash classifier to classify the incoming packets based on the *granularity* of the flow that meets our needs. We have extended the notion of flow granularity so that a flow is uniquely identified based only on the source

[¶]Using this mechanism, a user can communicate to the network his preference regarding the application that should receive the best performance. Using packet's IP Precedence field (ToS) and source port, UARA-U provides real-time traffic prioritization. However, UARA-U will not set the IP Precedence field in the traversing IP packets (it will only read it if set by the user himself); therefore, it does not need, and will not work with, a DiffServ domain in the core network (as opposed to UARA-DiffServ; see Section 6).

address (and therefore ignoring source port, destination address/port and flow id). We have then defined and coded another hash classifier (named SrcUARA) which is associated with the flow monitor. The *OTcl* code which is responsible to identify and calculate target bandwidth for high-bandwidth flows has been modified to work with the new logic. The result of these additions and modifications is that we have UARA-U's *user identification engine* and a part of its *traffic shaping engine* in place.

By checking packet's IP Precedence field and source port number, just before delivering a packet to the underlying RED queue, we add *real-time traffic prioritization* to the traffic shaping engine. In this way, differentiated services and real-time traffic prioritization are provided (by enforcing a lower drop probability on the detected high priority traffic). The engine knows certain port numbers used by applications with real-time needs (e.g., HTTP, Telnet and VoIP) as well as a publicized port-range (e.g., 20100-20200). By using the publicized port-range, any new application can indicate that it needs the highest possible bandwidth allocation. The current implementation of the engine makes decision only based on known source port numbers (used by different traffic types) which is sufficient for the purpose of performance evaluation of UARA-U.

4. Simulation Scenario and Results

4.1. Simulation Scenario

The simulated scenario is depicted in Figure 3. The point of interest is the queue on the output interface of *Node 0* (edge router) in the direction of the core network. The link connecting *Node 0* (edge router) to *Node 1* (core router) is the bottleneck of the network with a chosen speed of 2.5 Mbps. This speed ensures that we will have a congestion situation as soon as all the traffic sources start to transmit. Note that this speed merely corresponds to the chosen number of users and the rate of traffic sources and does not lead to the loss of generality while interpreting the simulation results. On the other hand, the link connecting each user to the edge router has a speed of 1 Mbps in the upstream direction and 8 Mbps in the downstream direction, reflecting typical home Internet connections. Furthermore, to experiment with the effect of varied RTT values, the links connecting receivers of the traffic to Node 1 (core router) have either 25 ms (for odd-numbered receivers, like FTP receiver 1) or 100 ms (for even-numbered receivers, like FTP receiver 2) propagation delay. In what follows, we explain the types of users and the traffic flowing in the network.

To simulate the P2P traffic (capturing transport/network layer characteristics of the traffic), a rudimentary choking algorithm has been developed based on which a P2P sender (Node 19 or 20 in Figure 3) selects ten peers randomly from the twenty available P2P receivers and starts sending traffic to those selected. After a period of 10 seconds has passed (typical BitTorrent behavior), each P2P sender makes another random selection and starts sending traffic to the new peers. Each P2P sender has five simultaneous TCP flows to each selected peer. Finally, each TCP flow is set to have an application-level throughput of 10 Kbps which is the mean throughput of a BitTorrent connection, derived based on the formulas reported in [26].

The second type of traffic present in the network is FTP, a high-bandwidth TCP flow, set to have an application-level throughput of 300 Kbps. In Figure 3, Node 41 and Node 43 are FTP sender 1 and 2, respectively, and are connected to their corresponding FTP receivers, Node 42 and Node 44.

For simulating traffic produced by a video conferencing application, we have used a CBR source on top of UDP (sending 384 Kbps traffic, a typical video conferencing bandwidth). Node 14 and Node 16 (Video conferencing sender 1 and 2, respectively) send this video traffic to their corresponding

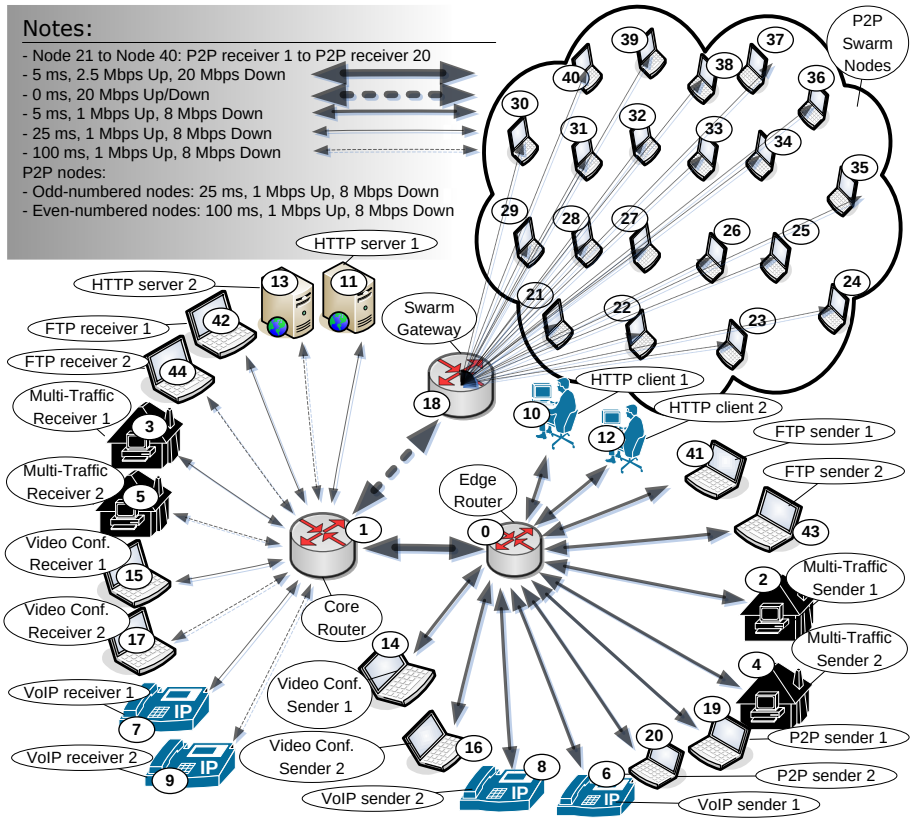


Figure 3: Simulation scenario

receivers, Node 15 and Node 17 (Video conferencing receiver 1 and 2, respectively).

In our network, Node 10 and Node 12 are HTTP clients (HTTP client 1 and 2, respectively) and are connected separately to Node 11 and Node 13, the HTTP servers. Therefore, the HTTP traffic would be clients sending HTTP requests to the HTTP servers, in response of which HTTP servers send the requested objects of the webpages (each webpage consisting of several objects such as images, etc.). To simulate the HTTP traffic, we have used NS-2's PagePool/WebTraf module[‖]. Using this module, we have set up a web traffic scenario in which each client requests four webpages consecutively. There is a uniform random variable which determines the *user think time* before requesting the next webpage (uniform between 5 and 15 seconds). Each webpage consists of ten objects (images, etc.) of various sizes, with their size being determined by a Pareto distribution (which is not an issue in the upstream direction). A client sends requests for the objects of a webpage almost simultaneously (an exponential variable with an average of 10 ms determines the inter-object period).

The last simulated traffic type is Voice over IP (VoIP). We have used the Ns2voip module [27] to

[‖]The module has been modified to produce per-page traces in addition to the per-request/response traces.

simulate the VoIP traffic and gather statistics on the receiver side. This is a sophisticated and well-developed module which is able to simulate various conversation types (we have used the "one-to-one" model) and calculate the Mean Opinion Score (MOS)** in the receiver side for the received VoIP traffic. We have chosen a G.711 encoder (a high quality 64 Kbps encoder) for our VoIP sender nodes, Node 6 (VoIP sender 1) and Node 8 (VoIP sender 2) which are connected to Node 7 (VoIP receiver 1) and Node 9 (VoIP receiver 2), respectively.

4.2. Key Concepts: Benchmark Performance and User Quality of Experience

We use the term *Benchmark Performance* to refer to the performance associated with each type of traffic when all other traffics in the network are absent; therefore, we have an uncongested network (more specifically, when the queue at the edge router is lightly loaded.). We take measurements and determine the benchmark performance of each type of traffic using the DropTail queue type.

Using the measured *Benchmark Performance*, we proceed to determine what we call *User Quality of Experience (QoE)*. *User QoE* is a ratio of the performance of each type of traffic during congestion to the previously-determined *Benchmark Performance*. User QoE is therefore a number between 0 and 1; the closer to 1, the better the user's Quality of Experience (User QoE can be expressed in percentage as well). An example to calculate the user QoE is as follows: an end-to-end delay for a given traffic type is measured to be 50 ms when there is no congestion in the network (i.e., all other traffic sources are off). This measurement of 50 ms is the *Benchmark Performance* for the end-to-end delay metric. We then turn on all other traffic sources and create congestion and then measure again this end-to-end delay. This second measurement could be 100 ms, for example (the congestion in the network results in higher delay). For this particular metric, end-to-end delay, of this particular traffic type, the User QoE will be $\frac{50ms}{100ms} = 0.5$ or 50%.

This is the method that we have used for normalizing the results with regard to different traffic types and various traffic-type-based metrics. As indicated in the introduction of this paper, the notion of (Average) User Quality of Experience, and the approach taken to calculate it, is another contribution of this paper, apart from UARA, and can be utilized in other similar performance comparison studies.

4.3. Performance Metrics

We have conducted extensive simulations results of which are reported in Figures 4 and 5. In order to have a statistically sound analysis, 95% confidence intervals are shown as error bars in the sub-figures (where there seems to be no error bar, the error, i.e., the difference between simulation runs, is near zero). In each of the sub-figures, *y-axis* indicates the metric in question for the studied traffic type. The *x-axis*, however, has been used to separate the results of simulation runs for each queue type selected in the edge router. In the sub-figures where there are two curves, the curves present the results for two cases: small and large Round-trip Time (RTT) values (depicted in order to see the effect of varied RTT values).

The average delay in the queue is depicted in Figure 4a. The queue size limit has been set to 100 packets for all queue types and the bandwidth for the outbound link, i.e., the link between *Node 0* (edge router) and *Node 1* (core router), has been set to 2.5 Mbps, as mentioned earlier. The results depicted

**MOS is a subjective measurement of voice quality described in ITU recommendation P.800. The E-model, described in ITU recommendation G.107, is a calculation based on delay, packet loss, etc. which enables to have an estimated MOS. MOS value for a voice communication ranges from 1 "impossible to communicate" to 5 "very satisfied".

in the figure are for under-congestion performance with all queue types.

Regarding the P2P traffic, we have measured the overall P2P throughput sent by a given P2P sender in the link connecting Node 1 (core router) to Node 18 (swarm gateway) (cf. Figure 3). The benchmark performance of this overall throughput as well as the under-congestion performance with all queue types are presented in Figure 4b.

The metric that we are tracking regarding the Video and FTP traffics is the mean throughput. The benchmark performance as well as the under-congestion performance with all queue types for Video and FTP are depicted in Figures 4c and 4d, respectively.

For HTTP traffic, Figure 4e indicates the number of pages that have been successfully received during the simulation. This page number is our first metric for the HTTP traffic. Mean delay starting from a client sending a request for an object till that request is received successfully by the corresponding server is our second metric for the HTTP traffic (cf. Figure 4f). The third metric (figure not shown) is the mean page duration, i.e., the time from the first request for an object of a page is sent till the last response of that page is received from the server. User QoE for the web user (the HTTP client in question) is an average between the three QoEs (corresponding to the three metrics).

Finally, for Voice over IP traffic, the benchmark performance as well as the under-congestion performance with all queue types are reported in Figure 5a for sender-to-receiver VoIP frame delay and in Figure 5b for the calculated MOS at the VoIP receiver side. To calculate the user QoE for VoIP users, we have used the following formula[††]: $0.25 \times \frac{FrameDelay(Benchmark)}{FrameDelay(Congested)} + 0.75 \times \frac{MOS(Congested)-1}{MOS(Benchmark)-1}$.

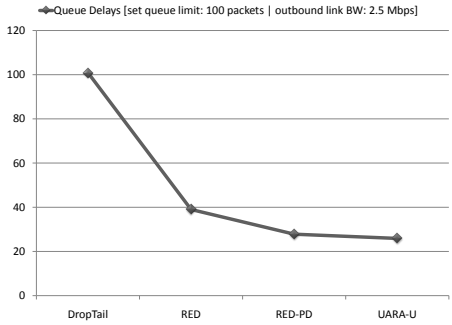
4.4. Performance Comparisons

We simulated the network while all users are active and have documented the results for each queue type. As outlined above, a QoE has been calculated for each user, comparing the performance achieved under congestion to the performance achieved when there is no congestion. Finally, an average of QoE of users has been calculated for each queue type. Note that each user has one vote in the calculation of average QoE. Therefore, for traffic types which have multiple metrics (like HTTP and VoIP), QoE has been first averaged between those metrics before including it in the overall average QoE for all users.

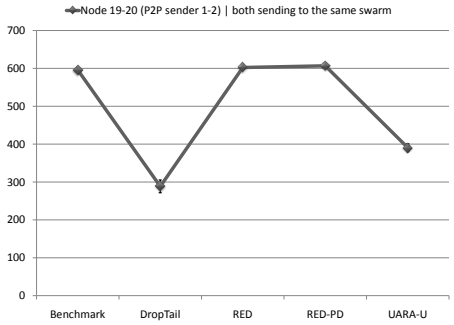
We do not delve into the details of QoE calculations for brevity, however, we explain one example (QoE for HTTP traffic with UARA-U queue type) which should be sufficient to demonstrate the approach taken. As seen in Figure 4e, for HTTP Client 1 (Node 10), the mean number of downloaded webpages is 4 in congestion situation with UARA-U queue (at the right of the figure), while this number is also 4 in an uncongested situation (Benchmark Performance, at the left of the figure). Therefore, the QoE would be $4/4 = 1.0$ or 100% for HTTP Client 1 (Node 10) with UARA-U queue type for the metric of number of downloaded webpages. QoE for other metrics has been calculated in a similar fashion and an average has been calculated between the different metrics of each traffic type where necessary (cf. Figure 5c). Finally, all the calculated QoEs of the users connected to the edge router have been averaged to report a single QoE per queue type (cf. Figure 5d).

We conclude this section by emphasizing the most important parameter which is the final calculated *Average User Quality of Experience (QoE)* for each queue type depicted in Figure 5d. UARA-U provides an average QoE of 54.6%, followed by DropTail, RED and RED-PD with average QoEs of 31.8%, 31.1% and 30.6%, respectively. The *Average User Quality of Experience (QoE)* is an all-inclusive indicator which shows the better performance of UARA-U, compared to other queue types.

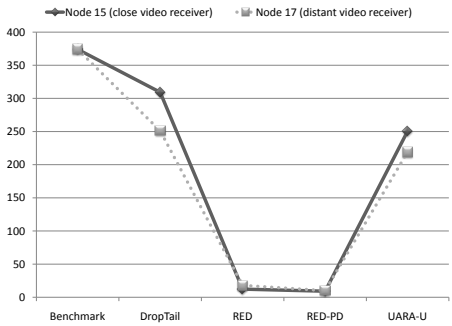
[††]The "-1" in the MOS part of the formula is due to the fact that MOS range starts from 1.



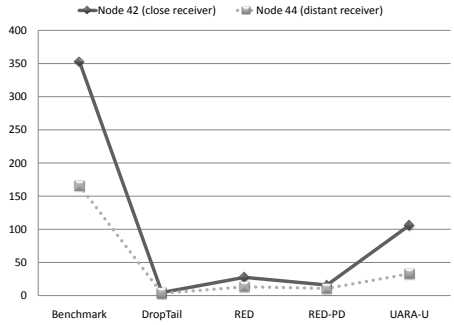
(a) Queue Delay (msec)



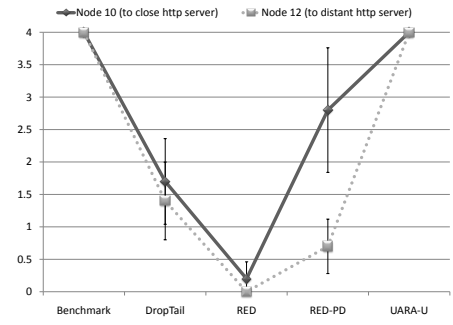
(b) P2P Throughputs (Kbps)



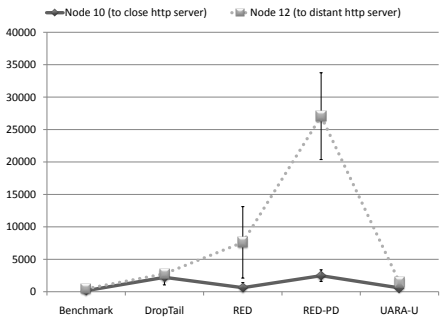
(c) Video Throughputs (Kbps)



(d) FTP Throughputs (Kbps)

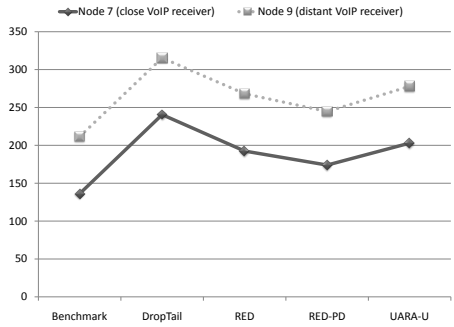


(e) HTTP: Number of Pages Downloaded

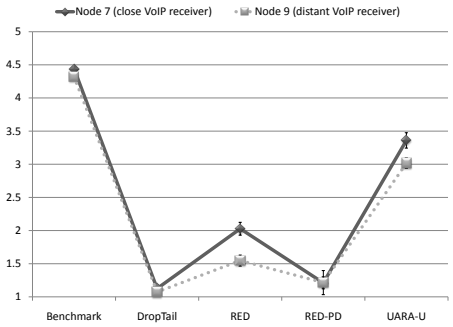


(f) HTTP: Client to Server Request Delay (msec)

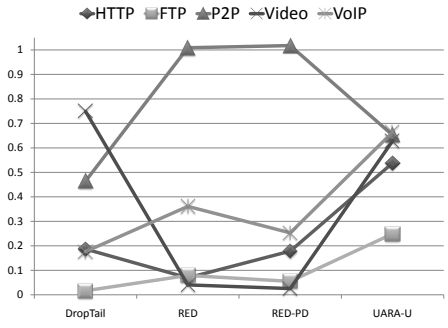
Figure 4: Simulation Results (10 runs of 100 seconds each per queue type)



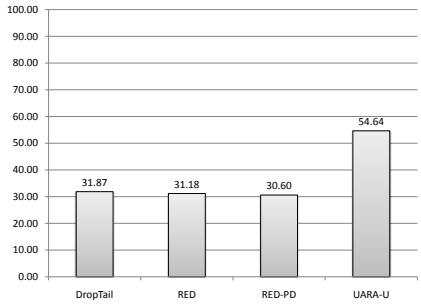
(a) VoIP Frame Delay (msec)



(b) VoIP MOS



(c) User Quality of Experience (QoE)



(d) Average User Quality of Experience (in percent)

Figure 5: Simulation Results (continued)

5. Summary and Discussions

UARA centralizes the following three functionalities which are currently implemented in a fragmented fashion (i.e., implemented in different parts of the network) and therefore lack effectiveness as a result of not being able to apply a coherent overall policy:

- Congestion avoidance: currently implemented in RED-enabled routers (most routers still use DropTail, however, which behaves poorly with regard to congestion avoidance.).
- Providing user fairness: currently offered by putting strict volume caps on all users. There is no mechanism, however, to control and provide fairness in real-time.
- Prioritizing real-time traffic: currently implemented using: 1) traffic shapers deployed in the core network (which throttle the P2P traffic as a means to provide a preferential treatment towards the real-time traffic); 2) using deployments of QoS architectures such as DiffServ, etc.

UARA's primary role is to provide the aforementioned three functionalities in one place, i.e., at the edge routers. Each of these functionalities affects the user experience; therefore, implementing them in a fragmented fashion, which is the case in today's Internet, leads to chronic or occasional mistreatment of users of certain types of traffic.

In what follows, we remind some important aspects, mention few limitations and address some often-expressed concerns regarding AQM schemes and fair bandwidth allocation mechanisms:

Getting the incentive right. UARA removes the incentive for any type of traffic to hide by means of changing port numbers and header/payload encryption, or any other means. The reason is that UARA does not compare the types of traffic of one user with another user; it only classifies the types of traffic of each user and prioritizes the real-time traffic of the user. Therefore, there is no incentive for the user or any user-based software (e.g. a BitTorrent client) to try to hide its traffic to gain an advantage in relation to the share of the resources that it gets. As a result, only the simple port-based traffic identification can be implemented in the upstream as a means to identify/classify the traffic in UARA's traffic shaping engine (and this is only when the user has not set the IP Precedence field of the IP packets.).

UARA is statistical-multiplexing-friendly. UARA preserves the benefit gained by statistical multiplexing and does not isolate users from each other. This can be shown in an example scenario: An ISP has a 20 Mbps access link to a neighborhood. It sells 1 Mbps subscriptions to 200 homes. This means that the ISP expects that, at any given time, only 10% of users are active. This is a typical scenario with regard to ISP subscriptions. UARA does not break this model by isolating users. In this scenario, if more than 20 users are online and actively using their connections, then there will be congestion. Therefore, UARA starts to monitor high-bandwidth users (e.g., users who send/receive higher than 100 Kbps). UARA classifies the traffic types of the high-bandwidth users and drops their packets with a probability according to the level of activity of the users as well as time-sensitiveness of the type of traffic.

UARA is at the edge routers. UARA does not address a congestion problem that might be present in the core network. As mentioned earlier, congestion problems are known to be present more at the edges than any other place in the network, as the last-mile infrastructure is far more costly to upgrade. UARA is a solution for congestion management at the last-mile and cannot be tapped into, in its current form, for addressing the congestion problems elsewhere. In the core routers, user identification is not useful, since we cannot measure the amount of all the traffic within the network concerning any particular user. However, by assigning IP precedence in the traffic shaping engine in the UARA-DiffServ scheme in the edge routers, we can provide relative fairness in the congested core routers of a DiffServ domain (see Section 6).

Effect on users with different bandwidth packages. At current form, UARA has no way of acquiring this information and potentially favoring users who have paid more to benefit from higher upload/download rates. However, some mechanism for assigning "weights" could be integrated in the future to account for this aspect. On the other hand, it is important to note that UARA concentrates on avoiding the blocking-out of a low-bandwidth user which seems to be the bigger concern. Furthermore, UARA's engines are active only during congestion times and stay aside when the router is lightly-loaded. This means that when there is no congestion, a user who has a high-end bandwidth package can benefit from what they have paid for.

Vulnerability to flow spoofing. The issue of flow spoofing, i.e., the traffic source using fake IP addresses when sending its packets, is an issue that is a common concern about any fair bandwidth

allocation scheme and UARA is no different[‡‡]. We believe that this is a security issue which should/can be dealt with using proper security equipments and UARA, being an AQM scheme, does not and cannot deal with such security threats posed by a minority of users. One approach could be that an ISP randomly checks, using other security equipments, to see if users declare their allocated IP addresses in the packets that they send. If users are found to be declaring fake IP addresses as source IP address, then they should be shut out completely due to security violations. Indeed, this issue is a security issue and cannot be taken into consideration in AQM designs.

Vulnerability to frequent IP address changes by the user. As our last point, we address the concern regarding frequent changes in IP address by a single user and whether this has any negative effect on the performance of UARA. UARA compares users in time scales in the order of few seconds to few minutes; therefore, if a user wants to fool the system and gain a greater share of the bandwidth by avoiding being monitored/bandwidth-limited, then they need to change IP address (through DHCP or else) every few seconds on average. While this is theoretically possible, it is obviously not very practical for any user to adopt such behavior due to the overheads involved as a result of continuous interruptions in communication. We therefore do not believe that there should be any concern regarding the effect of IP address changes on UARA's performance.

6. Future Work

Regarding UARA-U, we plan to carry out the following steps:

- Studying the effect of congestion on QoE of multi-traffic-type sources: Nodes 2 and 4 (multi-traffic senders) and Nodes 3 and 5 (multi-traffic receivers) (cf. Figure 3). These are users for whom all the mentioned traffic types have been set. We have not yet analyzed the effect of congestion and queue types on these multi-traffic sources and receivers and the analysis is left as future work (the real-time traffic prioritization aspect needs to be enhanced.).
- Including other AQM schemes (e.g. REM) in our performance comparisons (we predict their performance to be around DropTail, RED and RED-PD due to their similar operational logic.).

As mentioned before, there are three other editions of UARA which are currently being designed and will be evaluated later on. In what follows, we concisely explain our initial ideas regarding these three editions. UARA-D, or UARA-Downstream, is a version of UARA that can be implemented in the downstream direction. Source-port-based traffic identification (and use of IP Precedence field) works well in the upstream direction where the user can declare her preferences to the network. In downstream, a more sophisticated traffic identification should be deployed, as the user has no way of communicating her preferences to the network. This is the point of difference between UARA-U and UARA-D. For incorporating traffic type prioritization in UARA-D, we currently consider a combination of destination-port-based and behavior-based traffic type detection (as used in P2P traffic shapers). However, this issue should be explored further and is left as future work.

[‡‡]It is perhaps best put by the authors of [28] regarding their own fair-bandwidth-allocation scheme: "*Some claim that such fair allocation mechanisms open the door to denial-of-service attacks through flow-spoofing; while we do not believe such arguments are sufficient to nullify the desirability of fair bandwidth allocations, and that this paper is not the place to delve into such arguments at length, we did want to note the existence of objections to the fair bandwidth allocation paradigm.*"

UARA-DiffServ complements a deployment of DiffServ routers by providing the functionality of marking the packets (i.e., giving priority) at the edge of the DiffServ domain. The difference between UARA-U and UARA-DiffServ is as follows: In the traffic shaping engine (cf. Figure 2), after dropping the packet with the calculated drop probability, if the packet survives, the engine will probabilistically set a low IP Precedence in the IP header of the packet. This IP precedence (in the Differentiated Services field of the IP header) will be used in the core routers to decide whether or not to drop the packet when they are congested. After setting the IP precedence, the packet will be forwarded to the RED core, as in UARA-U.

In UARA-ECN, *drop the packet* is replaced by *ECN-CE mark the packet*. This is also true in the RED core, i.e., RED algorithm no longer drops packets probabilistically; rather, it marks the ECN Congestion Experienced (ECN-CE) code point in the two-bit ECN field of the packets. Note that the UARA-ECN scheme is only implemented at the egress router after packets have traversed the network and have been ECN-CE marked. The *user identification engine* is now named *user identification & ECN-check engine*; the engine now checks flows to see if they are responsive to ECN as well. If a flow is found to be ignoring the ECN marks (i.e., it is ECN-unfriendly), its user is *monitored* and the flow information is sent to another decision block. When a new packet arrives which belongs to a monitored user and to a flow that is ECN-unfriendly, the packet is dropped with a very high probability.

After UARA-U, we will finalize the design, implement and evaluate UARA-D, UARA-DiffServ and UARA-ECN. In parallel, we are also exploring the possibility of developing an analytical framework (mathematical modeling and analysis) to further examine and refine our proposal, the UARA scheme.

ACKNOWLEDGEMENTS

Special thanks to Prof. D. Qiu and Prof. M. K. Mehmet Ali, author's Ph.D. advisors at Concordia University, for their comments on an earlier version of this paper. Some of the work has been done while the author was with École Polytechnique de Montréal. Finally, the feedback of the anonymous reviewers has led to a great improvement in the presentation and overall quality of this paper.

REFERENCES

1. Peterson J, Cooper A. Report from the ietf workshop on peer-to-peer (p2p) infrastructure (may 28, 2008). IETF RFC 5594 July 2009.
2. Xie H, Yang YR, Krishnamurthy A, Liu YG, Silberschatz A. P4p: provider portal for applications. *SIGCOMM '08: Proc. of the ACM SIGCOMM '08 conf. on Data comm.*, ACM, 2008; 351–362, doi:http://doi.acm.org/10.1145/1402958.1402999.
3. Aggarwal V, Feldmann A, Scheideler C. Can isps and p2p users cooperate for improved performance? *SIGCOMM Comput. Commun. Rev.* 2007; **37**(3):29–40, doi:http://doi.acm.org/10.1145/1273445.1273449.
4. Gurbani VK, Hilt V, Rimac I, Tomsu M, Marocco E. A survey of research on the application-layer traffic optimization problem and the need for layer cooperation. *IEEE Communications Magazine* August 2009; **47**(8).
5. Demers A, Keshav S, Shenker S. Analysis and simulation of a fair queueing algorithm. *SIGCOMM '89: Symposium proceedings on Communications architectures & protocols*, ACM, 1989; 1–12, doi:http://doi.acm.org/10.1145/75246.75248.
6. Parekh A, Gallager R. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Transactions on Networking* 1993; **1**(3):344–357, doi:10.1109/90.234856.
7. Stoica I, Shenker S, Zhang H. Core-stateless fair queueing: a scalable architecture to approximate fair bandwidth allocations in high-speed networks. *IEEE/ACM Transactions on Networking* 2003; **11**(1):33–46, doi:http://dx.doi.org/10.1109/TNET.2002.808414.
8. Floyd S, Jacobson V. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking* Aug 1993; **1**(4):397–413, doi:10.1109/90.251892.
9. Mahajan R, Floyd S, Wetherall D. Controlling high-bandwidth flows at the congested router. *Proc. Ninth International Conference on Network Protocols*, 2001; 192–201.

10. Sen S, Spatscheck O, Wang D. Accurate, scalable in-network identification of p2p traffic using application signatures. *WWW '04: Proceedings of the 13th international conference on World Wide Web*, ACM, 2004; 512–521, doi:http://doi.acm.org/10.1145/988672.988742.
11. Message stream encryption specification. URL `http://www.azureuswiki.com/index.php/Message_Stream_Encryption`, consulted on 15-Jun-2008.
12. Madhukar A, Williamson C. A longitudinal study of p2p traffic classification. *Proc. 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems MASCOTS 2006*, 2006; 179–188, doi:10.1109/MASCOTS.2006.6.
13. Karagiannis T, Papagiannaki K, Faloutsos M. Blinc: multilevel traffic classification in the dark. *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer comm.*, ACM, 2005; 229–240, doi:http://doi.acm.org/10.1145/1080091.1080119.
14. Karagiannis T. Novel techniques and models for network traffic profiling: Characterizing the unknown. PhD Thesis, University of California Riverside June 2006.
15. Erman J, Mahanti A, Arlitt M, Cohen I, Williamson C. Offline/realtime traffic classification using semi-supervised learning. *Performance Evaluation* 2007; **64**(9-12):1194 – 1213, doi:DOI:10.1016/j.peva.2007.06.014. Performance 2007, 26th Int'l Symp. on Compu. Performance, Modeling, Measurements, and Evaluation.
16. Rossenhövel C. Peer-to-peer filters: Ready for internet prime time? Internet Evolution Technical Report March 2008. URL `http://www.internetevolution.com/document.asp?doc_id=148803`.
17. Aggarwal V, Akonjang O, Feldmann A. Improving user and isp experience through isp-aided p2p locality. *Proc. INFOCOM Comp. Comm. Workshops IEEE Conf. on*, 2008; 1–6, doi:10.1109/INFOCOM.2008.4544640.
18. Ietf p2p infrastructure workshop May 28, 2008. MIT campus in Cambridge, MA USA.
19. Briscoe B. Flow rate fairness: dismantling a religion. *SIGCOMM Comput. Commun. Rev.* 2007; **37**(2), doi:http://doi.acm.org/10.1145/1232919.1232926.
20. Briscoe B, Moncaster T, Burness L. Problem statement: Transport protocols don't have to do fairness. IETF Internet-Draft [draft-briscoe-tsvwg -relax-fairness-01] July 2008. (Work in Progress).
21. Jacquet A, Briscoe B, Moncaster T. Policing freedom to use the internet resource pool. *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, ACM: New York, NY, USA, 2008; 71:1–71:6, doi:http://doi.acm.org/10.1145/1544012.1544083.
22. Briscoe B, Jacquet A, Moncaster T, Smith A. Re-ecn: Adding accountability for causing congestion to tcp/ip. IETF Internet-Draft [draft-briscoe-tsvwg -re-ecn-tcp-06] July 2008. (Work in Progress).
23. Woundy R, Livingood J. Ietf p2p infrastructure workshop presentation (comcast) May 2008.
24. Briscoe B. A fairer, faster internet protocol. *IEEE Spectrum* December 2008; .
25. Khosroshahy M. Uara resources. URL `http://www.masoodkh.com/uara/`.
26. He G, Hou J, Chen WP, Hamada T. One size does not fit all: A detailed analysis and modeling of p2p traffic. *Proc. IEEE Global Telecom. Conference GLOBECOM '07*, 2007; 393–398, doi:10.1109/GLOCOM.2007.80.
27. Bacioccola A, Cicconetti C, Stea G. User-level performance evaluation of voip using ns-2. *Workshop on Network Simulation Tools (NSTools)*. Nantes, France, October 22, 2007.
28. Pan R, Breslau L, Prabhakar B, Shenker S. Approximate fairness through differential dropping. *SIGCOMM Comput. Commun. Rev.* 2003; **33**(2):23–39, doi:http://doi.acm.org/10.1145/956981.956985.

AUTHOR'S BIOGRAPHY

Masood Khosroshahy is currently a Ph.D. candidate in the Electrical and Computer Engineering Department of Concordia University, Montreal, Canada and an IEEE Graduate Student Member. He received his B.Sc. in "Electrical Engineering-Telecommunications" from Iran University of Science and Technology, Tehran, Iran (2004) and his M.Sc. in "Networked Computer Systems" from Télécom ParisTech (École nationale supérieure des télécommunications), Paris, France (2007). His current research interests include Peer-to-Peer Botnets, Peer-to-Peer Traffic Optimization and Congestion Avoidance and Control.