# Light-weight End-to-End QoS as DoS Prevention

Marcel Waldvogel Tobias Köck

***Abstract*—Despite decades of QoS research and many years of DoS defence work, neither group of proponents have been able to get their results included into mainstream Internet service. It seems that demand for either solution exists, but individually, they seem to be just below the cost/benefit threshold. This paper proposes a first step into a common solution, where combined and extended interests will hopefully allow us to surpass this threshold. While there are still some open issues, we hope to not only propose a basic working mechanism but also provide fresh ideas to start thinking off the beaten path. Our main contribution is to create a lightweight, end-to-end binding between path and service, which is then used as a basis to associate further attributes and mechanisms to this binding. As a result, both DoS defence and QoS can be achieved with stateless routers and only with prior consent of receiving the end systems, short, achieving several of the IntServ advantages in a DiffServ-style system, i.e., avoiding per-connection state.**

## I. Introduction

Despite decades of QoS research and many years of DoS defence work, neither group of proponents have been able to get their results included into mainstream Internet service. Besides technical issues, such as statefulness or management overhead, there seem to be different reasons for either category. For QoS, limiting factors seem to include fears from the service providers, such as the uncertainty of a new service, and the current ability to manually set the priorities for the few services or customers that need it. For DoS, the reluctance seems to be related to some ISPs considering DoS their customers' problems and to the general segregation of ISPs into two classes, namely, ISPs who serve home customers (which frequently are the DoS sources) and those who serve content providers (which became the victims). The former rarely have a business interest to act on behalf of the latter to clean up their customers' insecure clients turned into zombies.

In this paper, we propose a new mechanism which also shifts the interests: Involve different stakeholders into the DoS problem, which might actually be interested in solving it; provide more immediate return-on-investment; provide two services at the cost of one; allow other services to build on top of our mechanism; focus on end-to-end mechanisms with minimal network involvement; the absence of an *a priori* requirement modifications to a large installed base of network equipment characterise our new approach.

Our lightweight scheme–router-assisted, receiver-driven QoS (RarQoS)–allows the parties with vested interest to take action and as a result obtain better quality under heavy load. This not only presents a line of defence against rare events such as DoS, but at the same time can be used to improve QoS, a stronger driving force. The change necessary for the network providers is minimal, their role is essentially limited to only act as a third-party verifier of the sources claims. All the important decisions remain with the end systems and their users or administrators. It further allows incremental deployment, where already a small deployment will show benefits, something which is generally lacking in other approaches. We believe that this better reflects the interests, market forces, and end-to-end design of the Internet.

The paper is organised as follows. Section II provides background. Section III introduces the techniques behind RarQoS, our router-assisted, receiver-driven QoS mechanism. Section IV provides information about our ongoing implementation. Finally, Section V concludes the paper.

Distributed Systems Laboratory, Department of Computer and Information Science, University of Konstanz, 78457 Konstanz, Germany, <firstname>.<lastname>@uni-konstanz.de. Work was started at IBM Research, Zurich Research Laboratory [1].

## II. Background

### A. Denial of Service

In early 2000, the Internet world was shocked: Several resource-rich commercial sites were unreachable for several hours, probably due to the actions of a single individual who previously had gained control over many thousand computers world-wide [2]. This shock resulted in a series of proposals how to prevent future disasters. In the past six years, it was tried to reach consensus on how to improve the situation, but to no avail [3]. We believe that the reasons do partly lie in the form of the proposals, as they address the wrong audience. To set the stage, we first identify five components of a DDoS attack:

*Zombies.* The perpetrator starts collecting nodes to use for his attack, the Zombies, typically through a worm, virus or trojan with a remote control interface.

*Start signal.* One component to influence or identify the perpetrator would be to listen into his control network and/or to inject commands. Our perpetrator thus sets up a series of intermediate agents, to which the Zombies connect and which can be used to conceal the source of the commands. Alternatively, the malware that was injected into the Zombies may include time-dependent instructions.

*Attack.* At some later stage, the hosts' new master issues the attack command, causing the Zombies to send predefined byte and packet sequences to the victim, typically at maximum speed. These sequences are typically designed such that they cause maximum effect without being easily identifiable for filtering purposes.

*Fake source.* To conceal the IP addresses of the Zombies, many attacks try to spoof their source address.

*Abort.* The victim will try to abort the attack by identifying sources or packet properties and selectively shutting them down.

The manifold approaches at DDoS prevention try to hinder the first four components or improving the countermea-

sures in the Abort phase. We can classify these approaches into seven categories:

*Anti-spoofing.* Each router should be able to verify whether this source IP address could legitimately come in on that interface [4]. This also limits several legitimate uses of asymmetric routing, from link sharing to satellite-downlink-modem-uplink scenarios.

*Packet information.* Routers should include information in each passing packet [5, 6, 7]. Besides breaking fragmentation (or excluding fragments from traceability), it has been shown that attackers can more efficiently insert fake traceback paths than the system can insert real paths, potentially causing a DoS on the system used to analyse traceback information [8].

*Router storage.* Routers should store a fingerprint of each packet for a short period of time, allowing the return route to be identified when the receiver presents an unwanted packet to the system [9].

*Automatic identification.* Potential DoS activities should be directly identified at routers, potentially enabling them to directly take measures [10, 11].

*Manual blocks.* Some ISPs reportedly identify their network's ingress routers which provide a particularly large part of DDoS traffic to the given victim and block all traffic from these ingress routers to the victim in an attempt to minimise DDoS traffic without shutting off too many legitimate sources.

*Peer-to-peer systems.* Instead of controlling the problem, control the reaction: When a resource is in high demand, create more replicas [12, 13]. A close relative of this proposal is the Google way of throwing enough resources at the problem to handle any load. These hosts can also be used to do prefiltering of requests, such as described in SOS [14].

*Pricing.* Let the market forces decide by increasing the price for packets to overloaded destinations [15]. Obviously, such a system would charge the owners of the Zombies, not the attacker. Even though it might be argued that this will teach the careless Zombie owners a lesson, an ISP would have a hard time obtaining the money from such customers or surviving the bad press the attempt would generate.

Besides technical issues, some of which are outlined above, there are also market issues: Those who should invest money in upgrading their equipment and risk more customer support calls or dropping customer satisfaction, among other things, are frequently not those that have an interest in setting up such a system. Typically, the victim's business partner is a hosting provider. This provider can at most do per-attack filtering and has no influence on the source of the traffic. Furthermore, the large consumer ISPs frequently offer only limited commercial hosting services and rarely have customers who form an attractive target or who would complain loudly if such an attack occurred; many would not even notice.

Even ISPs which are active in both high-profile hosting and have a large consumer base may not feel the pressure or may be unable to force the responsible departments to team up. The lack of pressure is frequently caused by the minute minority of the attack traffic coming from the own network and the availability of tools and processes that can be used to trace traffic *within a single network.*

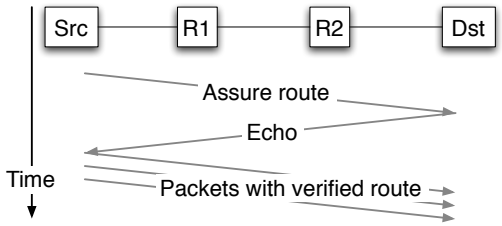



Fig. 1
RarQoS messages: Setup and data flow

## III. Router-assisted, receiver-driven QoS

### A. Overview

The design of RarQoS diverges from the established DoS prevention path. It was influenced by QoS ideas instead, noting that preventing DoS is just a special case of handling QoS. But as there is no need to provide QoS guarantees, but just a simple form of differentiating between multiple classes of best effort service, it does not suffer from the complexity and state explosion common to many QoS approaches, such as IntServ [16]. It also does not require establishing a mapping between different QoS parameters and contract negotiations, as necessitated by DiffServ [17].

How are these snippets constructed? Traditionally, routes have been recorded using the IP "record route" option, which records the full IP address of all intervening routers. The probabilistic packet marking schemes [5, 6, 7] distributed this information over multiple packets, requiring additional information for reassembly of the fragments by the receiver. RarQoS avoids both the data expansion and the potential state explosion at the receiver by making the packet marking a mechanism which is in the interest of the involved parties to gain better service. This avoids the need to squeeze data surreptitiously into some hopefully unused areas of the IP header as is currently done in anti-DoS mechanisms.

### B. Basic route recording

A first step would be to record tuples (*hop count*, *verifier code*) into the designated section of the packet. The verifier code consists of a value derived from flow information (e.g. the address/port/protocol five-tuple) and a secret known only to the issuing router. The derivation function must not be invertible by any other party than the router. Potential functions include keyed hashes or encrypting the flow information with the secret. Using 8 bits of verifier gives each router a 255-in-256 chance to identify forgeries, weakening the attacker's effort by a factor of 256.

It may seem that 256 options are easy to probe, which it in fact is. RarQoS does include mechanisms (described below), which do prevent effective probing of a single router's verifier code and restrict potential attackers to only probe the entire path.

Assuming both values in the tuple to consist of 8 bits, this would require 16 bits of information per RarQoS-enabled router along the path. Today, paths of 20...30 hops are quite common, this encoding would result in 320...480 bits to be included in each packet, plus some distinguishing header, clearly an undesirable cost-performance ratio.

The information can be halved by noting that the *hop count* fields do not provide 8 bits of information each. Options include delta-encoding the hop-count differences (more effort at the router) or having an index header pointing to the next field to use, to be updated at each RarQoS-step (requiring packet update). As a result, we get down to $8 + \varepsilon$ bits per step, but making the packet forwarding process more expensive.

The process can be further strengthened by having routers set a flag when they recognise a mismatch in the verifier code, as an alert to routers further down the road, that this packet has been tampered with and that it should be forwarded only when there is no congestion (Fig. 2, "misbehavior detected bit"). Then, the efforts of all the routers are multiplicative, no longer just additive. For $r$ RarQoS routers using $b$ bits of verifier each, we reduce the chance of an attacker picking an invalid identifier from 1 in $r \times 2^b$ to 1 in $2^{rb}$. Assuming a path of only 8 RarQoS-enabled routers, the chance of guessing the right result is thereby limited to one in $256^8 = 2^{64}$, clearly impractical for a single source.

A clever attacker, trying to create a fake verification string, might consider to do a `traceroute` using the information and see when the "misbehaviour detected" bit is set as an indication to the downstream routers. One approach would be to prevent packets containing assurances (verifiers being built) from being returned. This would require including the assurance only in ICMP error packets, which will not cause further error messages. This would be a kludge at best, many would consider this a major abuse of the Internet protocol.

It turns out that this can be avoided by including the current hop count together with the flow information as an input to the keyed hash or encryption. For `traceroute` or other programs that would like to have packets returned at controlled locations in the network, it is essential to modify the time-to-live/hop-count field of the IP header, breaking the assurance of a packet actually coming from the designated source.[1]

This multiplicative effort allows us to limit the number of bits per RarQoS step to one, the impact of an attack will be reduced by a factor of $2^r$, with each router using just a single bit. This allows us to get rid of all counters: The bit is indexed by the time-to-live/hop-count field of the IP header, modulo the maximum number of expected hops, a number which can be determined during the setup message exchange.

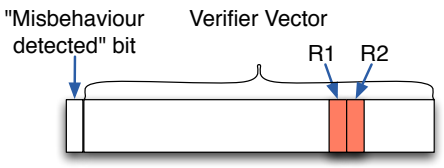



Fig. 2
Minimal RarQoS information

The entities enjoying the greatest benefit from this system are now the users/administrators of the systems that communicate. The ISPs also do not have to fear a massive surge in support calls, when they enable this feature, as it will not affect normal operation, only benefit under high load, such as those caused by DoS attacks. The hosting providers will be under the biggest pressure to enable their routers, as they will be under pressure from both the DoS load generated and their hosting customers. But also other ISPs, such as those providing connectivity to home users will have no disincentive to enable this feature. If it requires upgrade of the router hardware, this can be done in the normal router replacement process, as already a few routers spread throughout the path provide a tangible benefit to all involved parties.

### C. Path binding properties

To summarise, we have seen the following basic RarQoS path binding properties:

- Securely bind packets to a path with high probability.
- Require no router storage.
- Require minimal packet data.
- Session establishment can be piggybacked on existing transport or application layer setup or in a separate protocol.
- Receive IntServ-style path binding with minimal overhead (i.e., "DiffServ-style").
- Does not rely on symmetric routing (how to reduce the dependency on stable routing will be discussed below).

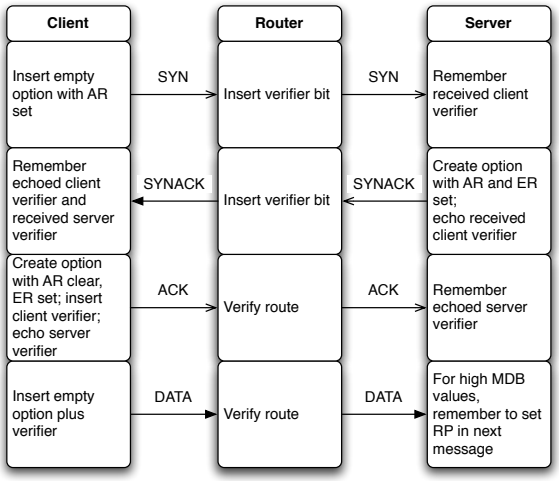
In the following section, we will extend these properties, bind parameters to the path, and discuss how these properties can be used to securely bind a path to QoS parameters or prevent senders from performing DoS under fake addresses, thus simplifying the filtering process and making it more effective.

## IV. Implementation considerations

We are currently implementing a prototype in the Scalable Simulation Framework, SSFNet,[2] to gain experience with the properties, refine open issues, and evaluate further applications. An inline session setup mechanism could

[1] It seems to be hard or even impossible to obtain echoes from selected routers along the path without using the time-to-live field in `traceroute` fashion. The use of *Path MTU discovery*-style mechanisms does not seem to be controllable enough for a successful attack, even more so in the backbone. A simple countermeasure would be to have ingress routers limit the MTU to a known maximum value valid until at least core egress, a useful mechanism in anyway. In either case the MTU reduction sites in the core will be rare. The user will have no choice on their placement, eliminating any usefulness it might have to an attacker's large-scale attempt to create false bindings, which is necessary to mount a DDoS siege.

[2] `http://www.ssfnet.org`




(Please note that RarQoS is not tied to TCP or inline session setup.)

Fig. 3
Message/component interaction

look as shown in Figures 3–5, when implemented as an IP option.[3]

Fig. 3 shows how a bidirectional RarQoS handshake could be integrated into TCP's three-way-handshake and the following data transfer. Integration into TCP's setup is not necessary, any application messages or even out-of-band signalling can be used to set up the RarQoS association. On the first message, the source asks the routers to set the bits they would like to see on further messages, such as their path can be validated. The target then echoes these bits back (potentially only after authenticating the source as trustworthy), such that the sender can now use elevated priority. A potential function for the router to use could be

$$b = h(\text{routerSecret} \parallel \text{timeToLivetypeOfService} \parallel \text{sourceAddress} \parallel \text{destinationAddress}), \quad (1)$$

where $h$ is a secure hash function returning a single bit, $\parallel$ is the concatenation operator, *routerSecret* is a per-router secret (no per-host or per-connection state), inclusion of *timeToLive* defeats hop-by-hop verifier secret gaining attempts, and the remaining variables are pieces of information from the packet which should be linked to this path.[4]

Fig. 4 shows the steps necessary in a router to process RarQoS messages, discriminating between legacy IP datagrams, setup and verification messages. The optional route adaptation process is described in Section B.

[3] We are evaluating alternatives such as IPv6-style Extension Headers, incorporation into IPv6 flow label or IPv6 addresses, as well as the reuse of existing IPv4 header fields. As the protocol relies on the co-operation of parties, unlike some surreptitious DoS packet marking schemes, the IP ID field could be "legally" used, with options to use the fragmentation header fields as well.

[4] Source and destination ports could be included into the calculation but would prevent a client from reconnecting to a server under DoS.

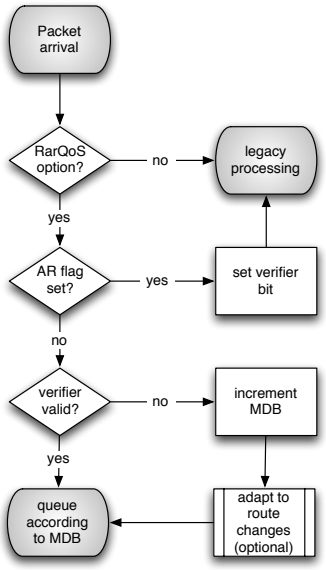
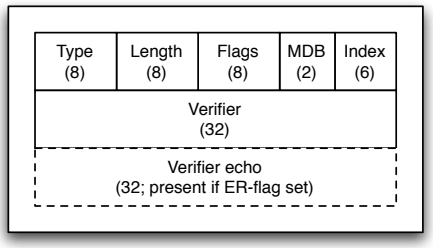



Fig. 4
Router message processing




**Legend**

Type: IP option type
Length: IP option length

*Basic flags:*
– AR: Assure Route (sender to router)
– ER: Echo Reply (sender to receiver)
– RP: Re-Probe (sender to receiver)

*Optional flags:*
– BM/AM: Before/After Match (router to router)
– Shift/Dir: Compensate minimal route changes (router to router)

MDB: Misbehaviour Detection Bits (router to router)
Index: Number of verifier bits used (router to router)

Verifier: Route recorder (AR=1: router to receiver; AR=0: sender to router)
Verifier echo: Let receiver learn about verifier (sender to receiver)

Fig. 5
Possible message format when implemented as an IP option

### A. To option or not to option

A clean and flexible way to implement RarQoS would be as an IP option. It would be fully downward compatible for both routers and end-systems without requiring any extra packets. A full-fledged IP option is shown in Fig. 5. The most important field is the verifier field, which will be used by the source to prove the binding to the users. When the *assure route (AR)* flag is set, it will instead be used by the routers to let the receiver know the bits they would like to see for future packets following this path. The other big field, *verifier echo* can be used, together with the *Echo Reply (ER)* flag, by the interested receiver to tell the source how to get at this improved binding.

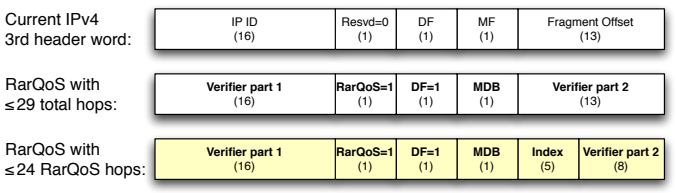



Fig. 6
Possible message format when avoiding IP options

But how do routers find out their bit in the verifier chain? One simple approach would be to take the current TTL/Hop Count value and use it (modulo 32) as an index into the verifier field. While this sounds attractive, it will limit RarQoS-protected paths to 32 hops, a design decision which, if realised, future users might soon regret. A better option is to include an index field, which will be decremented similar to the TTL field, but only by RarQoS routers. This index can then be used to select the appropriate bit to set or verify. This limits the path to contain at most 32 *RarQoS-enabled* routers, a variable which is much easier to control and provides a further benefit, as we will see.[5] Once paths start going beyond 32 RarQoS routers, an ISP might decide to disable RarQoS on select routers. This will not harm the efficiency, as most paths will still contain many RarQoS routers even after this reduction. Thanks to the exponential nature of the number of routers, already a few RarQoS routers will create a strong binding which only a tiny fraction of the total packets will be able to circumvent.

The introduction of an index field provides a further benefit, namely the opposite: allowing the introduction of virtual RarQoS routers. In the beginning, when only few routers will support RarQoS, only few hops will be verified, clearly not enough to bind paths with only a single bit per router. In this initial phase, a single router can extract several bits from its hash function to act as the local verifier "bit."

If verification of a single bit fails, a router will increment the value stored in the *Misbehaviour Detection Bits*. A single difference may not be an indication of cheating, as a router could have changed its secret key, e.g. through reboot, or a path component might have been replaced. Therefore, we propose that routers will gradually decrease the packet's priority when discrepancies in the verifier become apparent. Also, using the *Re-Probe* bit, a friendly receiver might tell the source that verifier information along the route has changed.

While the implementation as an IP option has several design advantages, as shown above, it has the practical disadvantage that many current routers will process packets with IP options very inefficiently along the "slow path" through the main CPU, bypassing the specialised forwarding hardware. RarQoS, being a co-operative protocol, unlike DDoS packet marking schemes, can use several fields from the stock IPv4 header. Especially convenient is the third word from the header, which contains IP ID, fragmentation flags, and fragmentation offset (Fig. 6). With the pervading use of Path MTU Discovery, many modern operating systems no longer require these fields. Therefore, they can be put to good use. The must-be-zero reserved bit would be modified to become a RarQoS indicator and the Don't Fragment flag would need to be set for legacy routers; the other bits could be used. Again, we have the option of using TTL indexing or putting aside a separate index field, to obtain more flexibility. Even in this space-constrained environment, the index solution shows its advantage.

[5] The *index* field with 6 bits seems excessive for enumerating 32 values. But with the ability to distinguish 33 cases, the "none used" and "all bits used" cases can be distinguished and wrap-arounds prevented.

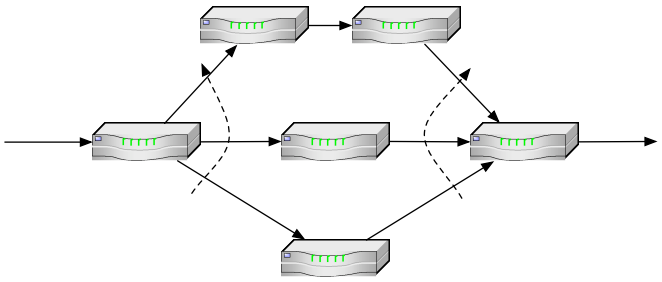


Fig. 7
Path changes with optional reduced impact

The fields, as used here, do not provide an option for establishment of the binding. This has been purposefully chosen, as legacy end-systems might become confused when receiving messages with non-zero fragment offset or must-be-zero bits set to one. In this case, the path would be bound using separate establishment datagrams sent before or with the actual connection-setup. This would be achieved using special ICMP messages or end-to-end packets with the IP option described above. This separation choice not only resolves the "slow-path" issue, it further degrades gracefully if an overzealous yet RarQoS-ignorant middlebox is in the path.[6]

### B. Dynamics options

Network paths today are rather stable, but only so much. There are always possibilities for a box to be replaced (Fig. 7), a component failing and requiring an automatic or manual replacement, or even the path getting slightly longer or shorter. Another aspects of dynamism includes wilful changes of the router secret.

While we expect RarQoS to react very quickly to path changes, especially if routers always update their verifier status in messages, not only in setup messages, the framework allows for the use of more sophisticated mechanisms to overcome small route changes at the local level. These changes will typically cause a single router to be replaced by it's hot standby or an alternative path being chosen, which might be slightly shorter or longer, in terms of hop count. These options are outlined in Fig. 7.

[6] Such firewalls have been reported to impede Explicit Congestion Notification (ECN) deployment.

Our optional mechanism includes a router, which notices a mismatch to the verifier bit and tries to guess whether the path has been shortened or prolonged by a single hop, i.e., verifying whether the previous or next bit would have matched. It records this result in the *Before/After Match* bits (shown in Fig. 5, but also possible in Fig. 6). If the next hope finds this hint confirmed, it can set the *Shift/ Direction* bits to indicate a single-bit offset from the original plan. Such guesswork slightly weakens the binding; our simulations will show whether this is worth the higher stability of the system.

A simple step to increase stability and which each router can implement independently, is to use a slightly different hash function input (Eq. (1)): Instead of taking the TTL, take the contents of the index field. This will only cause a change only, if a RarQoS router in the path changes or is inserted/deleted, but not if a legacy router is inserted into or deleted from the path.

When a router purposefully changes its secret to require old bindings to time out, it might also not flag new messages hard, but only set a flag indicating the usefulness of a reprobe to update the binding.

### C. QoS properties

While still maintaining DiffServ-style storage and communications overhead, it is now also possible to bind a path to the DiffServ parameters. This allows to reserve resources along the path and potentially also rejecting DiffServ requests by changing the code point or setting a flag in the RarQoS setup message. The path binding not only allows to borrow some IntServ properties into the DiffServ world, it may also offer new options for fraud detection and prevention in general.

This system can also be combined with (forward or reverse) charging mechanisms, requiring both parties' ongoing consent and thus making the system more transparent to the end users, combined with lower abuse potential due to the path binding.

### D. Anti-DDoS properties

How would such a system not only provide QoS path binding but also DDoS prevention? First of all, it provides a QoS differentiation for accepted users at the receiver and the passing-on of QoS binding credentials to other hosts with a sufficiently different path. These two properties will already significantly reduce the feasibility and effect of a DDoS attack. Yet, it does not prevent the sudden change of an apparently begin user of the system into a DoS attacker. As RarQoS enforces the path binding, at least if the attacker packets should consistently be treated at high priority, it allows the victim to ask upstream routers to install filters which reduce or stop the attack traffic [9]. Unlike other filter systems, it has very low false-positive *and* false-negative rates: (a) The attacker can not easily switch source address, as it would lose the QoS binding and (b) it is ineffective to blame someone else with fake sender addresses and thus causing the victim to install a filter blocking a legitimate communications peer.

RarQoS does not seem to be a mechanism that can be used to mount a DoS attack on the participating systems. The additional message processing requires slightly more CPU cycles, but it is independent of the actual message. Especially noteworthy is that setup messages do not require more processing that ordinary verification messages.

## V. Conclusions and future work

We described a lightweight scheme where the parties with vested interest need to take action and then obtain better quality under heavy load. The change necessary for the network providers is minimal, their role is essentially limited to only act as a third-party verifier of the sources' claims. All the important decisions remain with the end systems and their users or administrators. It further allows incremental deployment, might be combined with future charging mechanisms, where already a small deployment will show benefits, something which is generally lacking in other approaches. We believe that this better reflects the interests, market forces, and end-to-end design of the Internet. Our analysis of the properties of this scheme is very promising and will open new opportunities for merging QoS, DoS and path management.

Our next steps are to finish our SSFnet implementation, gain experience from our simulations and to experimentally implement it in a real operating system.

## References


[1] Sean Rooney, Christopher J. Giblin, Marcel Waldvogel, and Paul T. Hurley, "Identifying a distributed denial of service (DDoS) attack within a network and defending against such an attack," European Patent Application EP04405438.5, 2004.

[2] Jelena Mirkovic, Janice Martin, and Peter Reiher, "A taxonomy of ddos attacks and ddos defense mechanisms," Tech. Rep. 020018, Computer Science Department, University of California, Los Angeles, 2002.

[3] Rich Pethia, Alan Paller, and Gene Spafford, "Consensus roadmap for defeating distributed denial of service attacks," `http://www.sans.org/dosstep/roadmap.php`, 2000.

[4] Jun Li, Jelena Mirkovic, Mengqiu Wang, Peter Reiher, and Lixia Zhang, "SAVE: Source address validity enforcement protocol," in *Proceedings of IEEE Infocom*, 2002, pp. 1557–1566.

[5] Dawn X. Song and Adrian Perrig, "Advanced and authenticated marking schemes for IP traceback," in *Proceedings IEEE INFOCOM*, 2001.

[6] Drew Dean, Matt Franklin, and Adam Stubblefield, "An algebraic approach to IP traceback," in *Proceedings of the Network and Distributed System Security Symposium*, Feb. 2001.

[7] Stefan Savage, David Wetherall, Anna R. Karlin, and Tom Anderson, "Practical network support for IP traceback," in *Proceedings of ACM SIGCOMM*, 2000, pp. 295–306.

[8] Marcel Waldvogel, "GOSSIB vs. IP traceback rumors," in *18th Annual Computer Security Applications Conference (ACSAC 2002)*, Dec. 2002, pp. 5–13.

[9] John Ioannidis and Steven M. Bellovin, "Implementing pushback: Router-based defense against DDoS attacks," in *Proceedings of Network and Distributed System Security Symposium*, Reston, VA, USA, Feb. 2002, The Internet Society.

[10] João B. D. Cabrera, Lundy Lewis, Xinzhou Qin, Wenke Lee, Ravi K. Prasanth, B. Ravichandran, and Raman K. Mehra, "Proactive detection of distributed denial of service attacks using MIB traffic variables – a feasibility study," in *Proceedings of International Symposium on Integrated Network Management*, 2001.

[11] Ratul Mahajan, Steven M. Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, and Scott Shenker, "Controlling high bandwidth aggregates in the network," Tech. Rep., AT&T Center for Internet Research at ICSI, July 2001.

[12] Jianxin Yan, Stephen Early, and Ross Anderson, "The XenoService – a distributed defeat for distributed denial of service," in *Proceedings of CERT Information Survivability Workshop 2000*, Oct. 2000.
[13] Marcel Waldvogel, Paul Hurley, and Daniel Bauer, "Dynamic replica management in distributed hash tables," Research Report RZ–3502, IBM, July 2003.
[14] Angelos D. Keromytis, Vishal Misra, and Dan Rubenstein, "SOS: an architecture for mitigating DDoS attacks," *Journal on Selected Areas in Communications (JSAC)*, vol. 22, no. 1, pp. 176–188, Jan. 2004.
[15] David Mankins, Rajesh Krishnan, Ceilyn Boyd, John Zaho, and Michael Frentz, "Mitigating distributed denial of service attacks with dynamic resource pricing," in *Proceedings of Annual Computer Security Applications Conference (ACSAC 2001)*, 2001.
[16] Robert Braden, David Clark, and Scott Shenker, "Integrated services in the Internet architecture: An overview," Internet RFC 1633, June 1994.
[17] Steven Blake, David Black, Mark A. Carlson, Elwyn Davies, Zheng Wang, and Walter Weiss, "An architecture for differentiated services," Internet RFC 2475, Dec. 1998.