Link Weighting: An Important Basis for Charging in the Internet

Hans Joachim Einsiedler and Paul Hurley

Institute for Computer Communications and Applications (ICA)
Swiss Federal Institute of Technology - Lausanne (EPFL)
CH-1015 Lausanne

{hans.einsiedler,paul.hurley}@epfl.ch

April 14, 1998

Abstract

Current charging mechanisms in the Internet are restricted mainly to volume and time of the day. We propose a mechanism which allows to use additional information for charging in the edges and core of the network. Links or network clouds will have weights which are dynamic, e.g. based on congestion level. This mechanism covers the current Internet protocol, IPv4, and the future one, IPv6, and in both cases, unicast and multicast. We also provide discussion on how charging using these weights along links or network clouds from edge to edge could be implemented.

Keywords

Border Router; Charging; Congestion; Internet Protocol; Multicast; Differentiated Services; TTL Field; Unicast.

Contents

1 Introduction

The desire for simple predictable service guarantees to the already existing best effort in the Internet has resulted in differentiated service proposals which add new service classes. There has not, to date, been a specification of a mechanism for charging within each class nor a clear solution proposed.

In this paper we present mechanisms that can be used for charging within the context of differentiated service and indeed within the already existing best effort service if desired.

Volume of traffic is only one parameter in deciding on charge. An important additional parameter is *distance* which is a closer reflection of the actual cost to the network i.e. factors in the number of links, zones and Internet service providers (ISPs) a packet travels through. The availability and use of these resources can not be reflected solely in volume charging and the type of service.

To obtain distance measurements we need to be able to transfer link information from edge to edge, and maybe end to end. We facilitate this in the unicast case by changes to the mechanism of the Time to Live field (TTL); what we refer to as *Weighted TTL*. This minor change will not influence the original mechanism of the TTL field. Every packet without charging the header structure contains the distance information. No additional field or packet (which must be sent through the network) is necessary. This allows links to carry a *weight* of more than one as is standard now. In a multicast scenario we use the same weighting, but do so without modifying the TTL field's operation.

We assume that, at least at strategically important points, service class routers, which we call *trustable border routers* are installed. These routers split the network into trustable areas and are responsible for charge management, resource allocation and handling of the differentiated service flows - see [1].

1.1 Differentiated Services

Differentiated services are the provision of additional services classes to the already existing best effort service. They seek to provide priority and guaranteed services through simple bit differentiation where admission is done solely at the edges, reducing complexity within the network. The service allocation policy then becomes semi-static and domain-wide in contrast to integrated services, such as Resource Reservation Protocol (RSVP) [2], whose policies can be fully dynamic and hop-by-hop. The arrangement between host/clients and service provider or between service providers is envisaged as long-term and requiring minimum signalling.

The different proposals for differentiated services include *Assured Service* [3], *Premium Service* [1], *Differential Link Sharing* [4], *Scalable Reservation Protocol* [5], *Weighted Proportional Fair Sharing TCP* [6] and *Simple Integrated Media Access* [7].

The determination of a packet's traffic class is done using bits from the Type of Service byte (TOS) in the IPv4 header. Specifications for future differentiated services use of the TOS byte can be found in [8], [9], [10] and [11]. The *traffic class byte* in IPv6 [12] is the equivalent of the TOS byte in IPv4. This is currently being redefined to have a similar function to the IPv4 TOS byte. There also exists a 20 bit Flow Label in IPv6 for those packets which request special handling by a IPv6 router.

1.2 Outline

In the first two sections, we propose Weighted TTL for the IPv4 unicast case (Section 2) and the use of weights for IPv4 multicast (Section 3). Section 4 shows the implementation in IPv6. In Section 5, we describe the charging mechanism i.e. how the charging (weighting) information can be used between charging points. Finally we illustrate some directions this work opens up before some brief conclusions in Section 6.

2 Unicast Proposal

The TTL field is one byte in the protocol header in the current Internet protocol (IPv4). In the next generation Internet protocol (IPv6) the byte is called *hop limit*. Both the TTL and hop limit byte perform the same task, so we refer to *TTL* also when we are describing IPv6.

This proposal only concerns the TTL field. For the header structure and more details see [13] for IPv4 and [12] for IPv6.

It exists to discard packets which loop, and would otherwise not be dropped except for congestion reasons. The TTL field stores the duration (number of hops) to keep a packet before discarding. It thus also contains information about how many different routers a packet has traversed. It is initialised by the sender to some value (typically 32 or 64) and is decremented by one by each router that the packet goes through [13]. If the TTL value reaches zero it is discarded. For example a packet which travels through five routers (hops) with a TTL start value of 64, will reach the final destination with a TTL value of 59.

2.1 Studies

We studied the reaching of IP addresses from various different parts of the world in order to ascertain data on the number of hops. The goal was to find out the maximum and average number of hops.

First we used IP host location data gathered in [14], which we call data set I. This data is more than one year old. The information was generated by tcpdump [13]. Then we obtained more up to date information, which we call data set II.

For both data sets, we used traceroute [13] from different locations throughout Europe and one location in the US, which are listed in Table 1. We did not use the default value of traceroute (30 hops) but instead set the value to the maximum permissible of 255. During the measurement we discovered several problems, e.g. loops, unreachable networks or hosts or a number of gateways/firewalls which were not supporting the Internet Control Message Protocol (ICMP). As a result, we got valid results from the traceroute for around half the number of addresses.

Location	Hostname	IP Address
National University of Ireland, Galway	schmidt.ucg.ie	140.203.3.14
Technical University of Darmstadt, Germany	sun54.isa.informatik. th-darmstadt.de	130.83.14.8
University of Kjeller, Norway	janus.unik.no	193.156.96.46
University of California (Berkeley), USA	borel.eecs.berkeley.edu	128.32.239.102
Eidgenössische Technische Hochschule in Zürich, Switzerland	kom23.ethz.ch	129.132.66.2
Swiss Federal Institute of Technology - Lausanne (EPFL), Switzerland	lrcsun16.epfl.ch	128.178.156.78

Table 1: Locations in Europe and the USA

We tested data set I's 477 different addresses from all locations. Then we used data set II of 377 addresses and performed the same test from Berkeley and Lausanne. The results are shown in Figure 1.

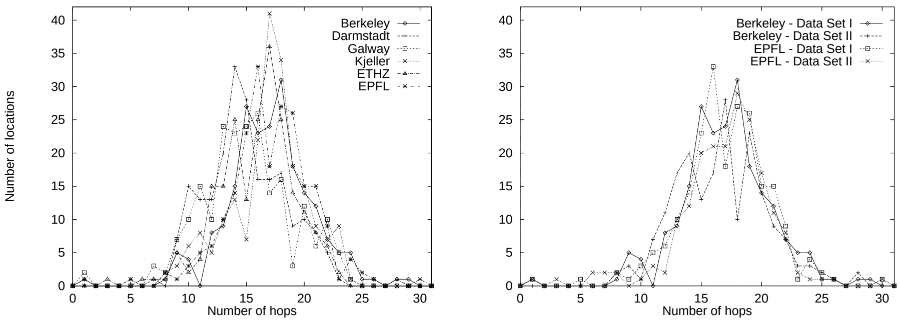

Figure 1: Graphs of Data Set I (left) and II (right)

	Location	Accessible Hosts	Maximum Hops	Hop Average
Data Set I	Galway	213	24	15.047
	Darmstadt	213	23	14.935
	Kjeller	211	24	16.825
	Berkeley	213	29	17.028
	Zürich	211	28	15.995
	Lausanne	217	30	17.023
Data Set II	Berkeley	193	28	16.679
	Lausanne	194	30	17.005

Table 2: Results of the data set I and II

Our results for both data sets I and II produced a hop average of between 15 and 17 - see Table 2. The maximum hop number for a normal packet was always less than or equal to 30.

2.2 Weighted TTL

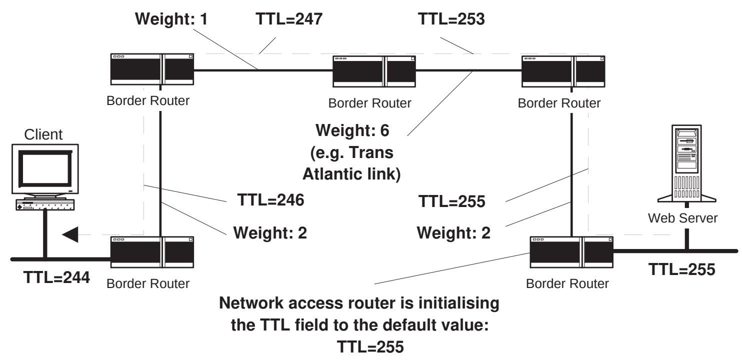

Figure 2: Weighted TTL in action: Network links with weights

We now describe *Weighted TTL*; our proposal to allow the transfer of weight information. This information is supplied by decrementing the TTL of each arriving packet by the designated weight of the previous link.

The resultant overall weight value can then be used in computing the price for link use, or even for networks between trustable border routers. The weight values are specified by the provider or owner of the link (in agreement with neighbours), and we allow the decrementation decision to be dynamic, e.g. congestion levels or the time of the day.

In [1], it is argued that for scaling reasons the complex tasks of the new service classes should be put as near to the edges as possible i.e. the decision as to how an arriving packet should be marked and deemed within a known host/client profile is made by the border router. We preserve this in that the donkey work of Weighted TTL (pricing calculations, setting default value, etc.) is placed at the edge at a trustable border router.

This first border router sets the TTL field of each IP packet for the given service class to the default value of 255, just after admitting the packet as being in profile. This condition is necessary because we assume no trustable hosts in the access network. This security also ensures that no client/user can alter the TTL field to their advantage (or disadvantage!).

The intermediate routers between hosts and border router will have no additional influence on the costs. All subsequent routers decrement the TTL field by a value of one or more, as decreed by the link's weight.

As in original TTL, Weighted TTL requires that a packet must be discarded upon TTL reaching zero.

As shown in Figure 2, the TTL field of the arriving packet in the last border router becomes a function of the number of networks passed and the specified weights. In this example, the weights are chosen according to a link's importance e.g. the Trans Atlantic link, being a very expensive and congested link is given a relatively high weight.

We propose a weight range from one to eight, as in Table 3. The standard fundamental weight between two routers is one as occurs now in the Internet. A value of two or bigger is for an interconnection of two ISPs or within an ISP network. The value reflects high maintenance cost, congestion and/or time of the day. The actual choice of weights is deliberately left open for further study. In the studies (Section 2.1) we made, we found that the maximum hop count was no bigger than 30. This means that with an implementation of this weight system in the current Internet, a packet can go over 30 hops without discarding given the assumption that each link will be weighted with the maximum of 8 in our proposal.

Weight	Comment - Status
1	Normal link (distance between two routers)
2	link between network clouds
3	important/high maintenance costs
4	.
5	intermediate stages
6	.
7	.
8	highly congested, with big maintenance costs

Table 3: Weight Range

2.3 Preservation of Loop Protection

Two types of loop are possible [15]:

- Persistent loops due to problems in some static routing tables.
- Temporary loops due to lack of synchronisation of dynamic routing table updates.

In both cases decrementations of the TTL field eventually leads to the discarding of the packet. The time before discarding depends on the initial TTL value before entering the loop.

An important design goal of Weighted TTL is to preserve the original functionality of the TTL field. We assume that loops will continue to occur for the same reasons as they currently do in the Internet. An open question is how loops will occur in networks with an implementation of differentiated service classes, e.g. loops back to the same access network.

Given that Weighted TTL still decrements the TTL field by at least one, the only change is in the amount of hops a packet takes before discarding.

With permanent looping, we could not be making the situation any worse. In the case of temporary loops any effect would be negligible.

Of course, we will no longer be able to ascertain the number of hops from TTL inspection of an IP packet but instead we have the information in every packet about the situation of the network in terms of congestion, maintenance, etc.

2.4 Weighted TTL Over Non Compliant Links

Of course, intermediate non-border routers will be placed in ISP networks. These intermediate routers need not necessarily be Weighted TTL aware. By building some intelligence into border routers we can build bridges over closed ISP networks which do not comply with Weighted TTL.

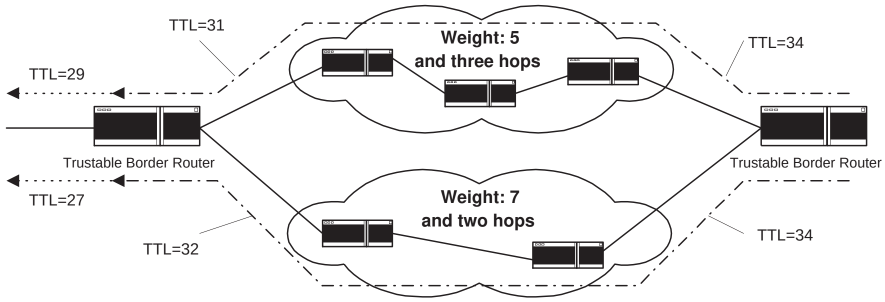

Figure 3: Example of supporting a Service Class over Non-Service Class Links

The border routers must know, or at least have a good estimate of, what weighting to apply to this area and the number of traditional hops in this area, for the purposes of computing the correct price as in Figure 3.

The weight over an ISP network with **N routers** which do not support the Weighted TTL is the following:

$$N \cdot Weight_{max} \geq Weight \geq N$$

Weighted TTL currently assumes that $Weight_{max}$ will be 8 as in Table 3. The weight is then dependent on the congestion of the links between the Non-Service class routers. The ISP can apply specific "virtual weights"(w) to the links in their networks and compute the whole value as the sum of these links:

$$Weight = \sum_{i=1}^{N} w_i$$

The trustable border router will subtract the difference $Weight - N$ because the intermediate non-border router already decremented the TTL field by the number of hops.

3 Multicast Proposal

3.1 Multicast and Differentiated Services

How multicasting can be done within the framework of differentiated services has not yet been explored.

The various current multicast protocols Distance Vector Multicast Routing Protocol (DVMRP), Multicast extension for OSPF (open shortest path first) [16], and Protocol- Independent Multicast (PIM) share two procedures in common [17]. The first datagram of a receiver is broadcast to all multicast routers. Additionally all leaf routers, which are the multicast routers next to the receiver, receive requests from nodes which will participate in a multicast session and these routers ask periodically if there is still someone on the same physical link connected.

On a shared medium we need only know if one is connected. To keep track of group membership, the multicast router sends a Internet Group Management Protocol (IGMP) query which starts random timers in all group members.

When the first timer expires the responsible group member sends a IGMP report to inform the multicast router that at least one group member is present. The other group members receive this IGMP report and reset their timers [13].

From the above facts, it can be decided whether or not a host/node is allowed to receive or send in a given service class and how much it is permitted to send. When a host sends multicast data and requires a specific service class for its multicast session, it must communicate with the first trustable multicast border router to decide if the host is in or out of profile. The exact semantics have yet to be defined and they are outside the scope of this proposal. For illustration purposes, however, consider the following. If the host is allowed to send multicast traffic in the requested service class, then the multicast router announces the session by broadcasting including the payment direction which is described later. If not, the router rejects the request and the host must either send a new request (i.e. reduce magnitude of the request or change traffic class) or give up.

In order to proceed we make the following service class multicast assumptions.

- At every branching point, there is a trustable multicast router which knows its number of branches (for definition please see section 3.2). Normally, edge routers on shared mediums do not explicitly know how many different branches they have. However, given a scenario where these routers are also differentiated service enabled, they will have profiles for each of their branches. Each branch must explicitly register to obtain their profile and a router can ascertain its number of branches from this.

- In a shared medium, the data which is forwarded to a member of the multicast group is encrypted if the data is payment sensitive. This is a fundamental requirement that is independent of and complementary to our proposal.

3.2 Unsuitability of Weighted TTL in Multicast

Weighted TTL is not suited for direct migration to multicast for the following two reasons:

- TTL Field's Use in Multicast DVMRP already uses the TTL field to indicate the scope of a multicast packet. All multicast router interfaces have configured TTL thresholds which any multicast packets must exceed in order to cross this interface. Forwarding of a packet will only happen if the value of the TTL field is greater than the TTL threshold assigned to the interface as shown in Table 4. For example a packet with a TTL field less than 16 should not be forwarded to another site in the same region. It is restricted to the same site. For further details see [17].

  Thus we can not use the TTL field for the transfer of costs in multicast.

- Fair Distribution of Cost

  Consider a multicast session where each receiver is responsible for paying. The cost of a given multicast session should be shared fairly amongst its receivers in relation to its cost to the network i.e. the distance travelled is shared.

TTL	Scope
0	Restricted to the same host
1	Restricted to the same subnetwork
15	Restricted to the same site
63	Restricted to the same region
127	Restricted to the same Worldwide
191	Restricted to the same Worldwide; limited bandwidth
255	Unrestricted in scope

Table 4: TTL scope control values

We define the *number of branches* as the number of active receiving nodes (hosts or routers) for a given multicast session connected to a multicast router. This value includes the number of local receivers i.e. the number of receivers who partake in the session directly from the router.

A fair distribution of distance cost amongst subtrees is given upon application of the recursive formula

$$cost_i = \frac{cost_{i-1} + \sum_{n=1}^{m} weight_n}{br} \tag{1}$$

where:

$cost_i$	Cost at branching point where $cost_0 = 0$
m	Number of links between the two branching points
$weight_n$	Weight of this links
br	Number of branches at branching point i

Thus, branching and cost information is needed at every multicast router in order to provide fair distribution of cost, and this information needs to be passed to the edge.

The TTL value cannot be used to store the fractions of weight as it can only be an integer. For this we propose a message which will carry this information down along a multicast tree.

Ways to share the cost of a multicast tree are analysed in [18]. Our scheme can be considered to be "Local members and next-hops are allocated identical hops" in their notation which is a one-pass mechanism. We don't require the total number of receivers downstream from each multicast router to be known; just its branching information. We share the cost fairly amongst subtrees and not total receivers. Cost division into areas facilitates collective charging and we believe it provides better control than direct division of each link's cost amongst all receivers. Our scheme, if desired, would require minute changes to work with the "All locals are considered as one next hop" (ENHS) scheme discussed in [18].

3.3 Weights in Multicast

The weight of the link and number of branches must be supplied from each router to the edge to enable fair cost distribution. We refer to this information in each router as its (branch, weight) pair.

This data is supplied by a message packet per service class per multicast session. The message can be the payload of a newly defined IGMP "cost" message or embedded in a UDP packet which is send out in the same service class with a different but related multicast address to the multicast data. It is then forwarded along to each branch.

The first trustable multicast border router next to the sender within the multicast tree generates the message periodically. Lost messages are not that important due to the periodicity. This message also announces the payment direction i.e. if the sender or receiver should pay. The message is built by adding this information into the first field and then the first pair (number of branches, 0). The weight field is set to zero because the multicast router does not yet know the weight of the outgoing link.

Every subsequent router is responsible for adding a (branch, weight) pair to every arriving message packet for a multicast session within the specified service class before forwarding. Each pair is built as follows.

- Each router increments the weight of the previously inserted pair by the weight of the previous link.
- If the number of branches from the router is more than one, it adds a new pair consisting of (number of branches, 0).

This ensures that routers with only one branch doesn't need to add another pair. For intermediate non-multicast routers which don't support our scheme, the tagging of weights is done at the next multicast border router.

3.4 Example

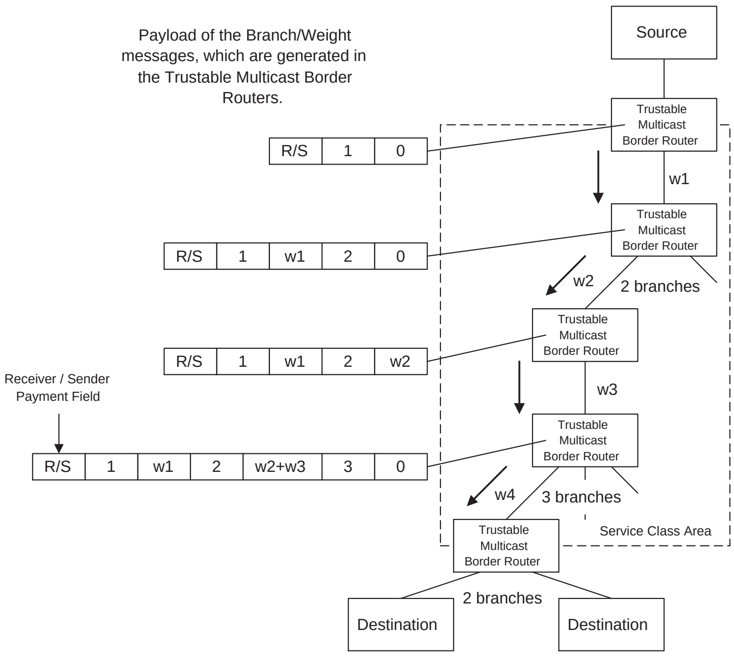

Figure 4: Branch of a multicast tree

Figure 4 shows part of a multicast tree. Each branch of the multicast is given a weight. The number of branches are known to the trustable multicast border routers. In this figure it is also shown the Branch/Weight message without IP header for clarity. It is periodically generated and goes down from the root to the leaf. This information is then passed to the charging application in each router. The last Trustable Multicast Border Router (leaf router) does not generate this message but adds the needed weight of the last link (w4) and the last Branch/Weight pair (2,0) and passes this information to the charging application.

4 Implementation in IPv6

4.1 Branch/Weight and IPv6

Using IPv6, there are two ways we could implement the Branch/Weight information packet.

- *Separate UDP message in IPv6 as in IPv4*
  The trustable multicast border router next to the sender periodically generates this message and the message is forwarded down the multicast tree.
- *Additional Header for IPv6*
  IPv6 allows additional information to be conveyed from the source to intermediate systems along the path, by the use of extension headers. Extension headers follow the basic header as shown in [12] and [19]. Below we define a multicast Branch/Weight header.

We propose an IPv6 header with the following structure (see also Figure 5).

- As defined, the first byte is the mark for the next header. The choice of header number is left open. (There are some headers already defined [19]). Then there is one byte for length. The length byte size allows us to add 84 Branch/Weight pairs which is more than acceptable.
- The third byte contains the payment information.
- Two bytes follow with the number of branches. We assume that two bytes are enough to store all possible branches in a future multicast network.
- The next byte contains the weight of the path.
- Then a new Branch/Weight pair, three bytes, is created and the process repeated.

There are two ways how this header can be used. Either the first trustable multicast router as seen by the sender can extend the new header to each multicast datagram or this can be done periodically with a multicast datagram. In this case, the subsequent multicast routers should update only this specific multicast datagram.

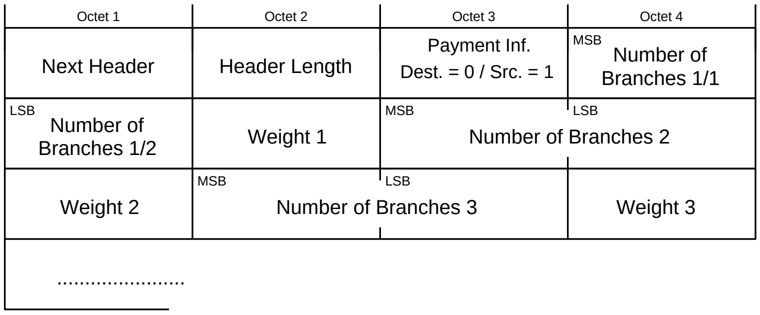

Figure 5: The Branch/Weight IPv6 header

4.2 Weighted TTL and IPv6

Since the TTL field also exists in the header of an IPv6 packet, we can also use the same mechanism as for IPv4. Another possibility is to use the header defined in 4.1. In this case the header is only six bytes long. The passing routers must modify only the weight field. The two branch bytes are set to zero for the MSB and to one for the LSB.

5 Charging Mechanism

As already mentioned, trustable border routers are placed at least at borders of ISPs. These strategically important points are places where *collective charging* with the sum of link weights per service class can be done. The sum of the link weights is stored in the TTL field of each IP packets and can be computed as $255 - TTL_{field}$. We only wish to provide an outline of how to use weights for charging. The use of volume or content (e.g. Web pages) combined with service classes is additional to this proposal. Collective charging signifies that the charging is done in reverse to the flow direction to the next ISP and independent of specific flows from or to certain clients/hosts. There is no distance charging in the access networks because we have only service classes from edge to edge.

We have two different flow directions. A data flow from the sender to the receiver and an acknowledgement flow in the opposite direction.

5.1 Data Flow and Charging Directions

We assume that there are two different cases for client charging in the end points of the network. The first one is when we have a receiver who wants to connect to a provider to "buy" data i.e. the receiver will pay for the entire transaction (both data and acknowledgement traffic) as in Figure 6. Host 1 is the receiver, host 2 the sender. In this case, host 1 must pay the ISP of network A for the received data and network B for the acknowledgement flow. The costs are accumulated and each pays for the part which they receive. Host 1 pays for the whole costs. Network provider A pays provider C the costs of network C and B. The provider of network C pays to provider B its cost.

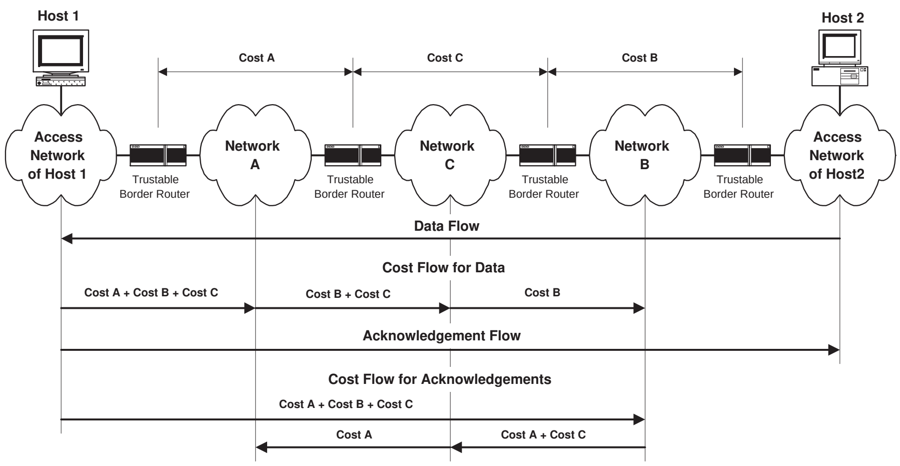

Figure 6: Receiver pays for transaction

In the second case, we have a sender which sends to a destination and is willing to pay for it. Host 2 pays the ISP of network A the amount for the sent data (Cost of ISP network A, B and C) and the ISP of network B the amount for the received acknowledgements as well as all costs. In the network we have the same costs transfer as in the case when the receiver pays.

For security reasons as well as to prevent “free riders” we need an authentication mechanism in the case where the receiver wants to “buy” data. This is not necessary when the sender pays because if they haven't the required service/charging profile, the first trustable border router will reject the request or send the data as best effort only.

Figure 7 illustrates how to handle the case when the receiver pays. Host 2 received a request for assured or guaranteed service from host 1. It then sends a newly defined ICMP message which is processed at the edges and not at all trustable border routers.

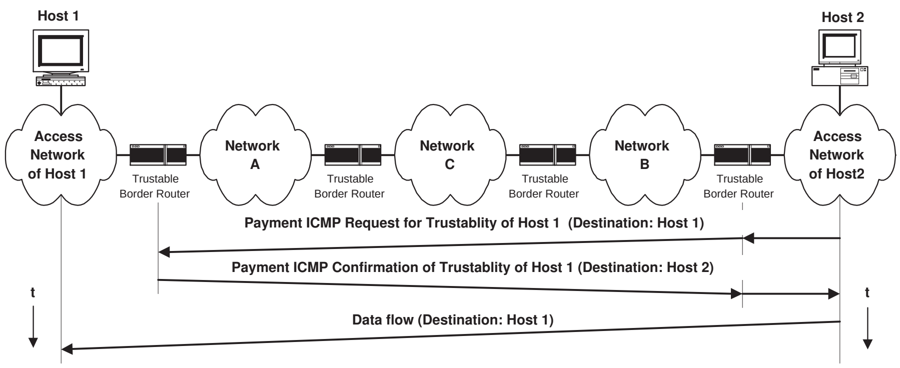

Figure 7: Announcement that receiver pays for transaction

The destination address here is the requesting host. The trustable border router next to the receiver picks up this message and processes it i.e. if the destination is permitted to receive service class data, it responds to the sender. The sender and responsible edge border router for this host receive the reply.

If the receiving host is not able or allowed to receive service class data, the trustable edge border router does not send any reply. If the reply gets lost, there is also no service class data transfer.

We propose an ICMP message for this basic signalling and authentication. This allows ease of implementation within routers and they do not have to examine all regular IP packets. We wish to leave open the actual design of this message because this message will be used by a higher layer charging application and we do not yet know the requirements this charging application will have. However, the message, in addition to carrying the information for announcement, can for example include the amount of data which is to be transferred.

5.2 Multicast Transmission Computation of Costs

Recall that a fair distribution of the costs between branching points was given by Formula 1 in Section 3.2.

Multicast Session Where Receiver Pays

The computation of the cost using the formula is done by the last trustable multicast border router to which the receiver is connected.

For example, the cost in example 3.4 is

$$cost_{Receiver} = \frac{\frac{\frac{w1}{2}+w2+w3}{3} + w4}{2}$$

Multicast Session Where Sender Pays

When the sender pays for the transmission, the final field for the number of branches is not needed because there is no distinction amongst service classes on the last hop (i.e. the last hop is "free").

The cost to the sender is the sum of all branches.

$$cost = \sum_{br=1}^{m} cost_{br}$$

where:

$cost_{br}$	Cost of the multicast branch
m	Number of all multicast branches

For the same example 3.4, the cost for the shown branch is

$$cost_{br(anch)} = \frac{\frac{w1}{2} + w2 + w3}{3} + w4$$

5.3 Fairness in Cost Allocation and Money Flows

We assume that packets which are dropped in the network (congestion or loops) are the responsibility of the provider. They are responsible for the costs which cannot be forwarded to the next domain. This produces an inducement for the provider to ensure that charged service class traffic will not get lost and that the end-to-end service will be guaranteed.

In the current Internet most traffic is not source routed. Packets from the same flow can take different paths. However, they merge at the last border router and with this so do the costs. Different paths do not influence our weight charging mechanism.

For the overall charging mechanism we have to include other parameters like volume, service class, flat rate, etc. Payment can be facilitated by already existing transfer methods such as Micropayment [20]. Special application will compute in border routers the costs and can be transferred to those responsible such as end-users/clients or ISPs.

6 Conclusions and Future Work

We have shown a mechanism which allows charging based on the path a packet travels. Charging based solely on volume and a service class is not a true reflection of a flow's cost to the network.

For unicast our proposal, *Weighted TTL*, enables this by extending the use of the Time To Live (TTL) field while still preserving its original functionality.

For multicast, we propose supplying the same path information as in the unicast case but to use a separate cost packet to ensure fair distribution of cost and because multicast protocols use the TTL information differently. We showed also that the branching factor is needed in the cost computation. Weights and branching factor as well as the payment direction are the information contained in this cost packet. It is possible that the cost packet could also be implemented by putting the aggregate fraction of cost calculation in the router thus keeping the size of the packet small and independent of the number of routers traversed. This has the disadvantage of requiring the router to perform these calculations. We are investigating the feasibility and advantages of this as part of our ongoing work.

In Internet studies, we have shown that in the currently used Internet protocol (IPv4) the mechanism can easy be implemented by making small changes in border routers. The structure of the IP packet is not modified. This is also possible in the future Internet protocol (IPv6). We can exploit the newly defined protocol new subheaders in IPv6, to choose a different way for transferring the information in unicast and multicast.

We described the concept of collective charging in the core network where each individual flow is not charged by the ISP. ISP will charge the traffic in bilateral exchanges with their neighbours. The use of weights then allows a degree of dynamic charging in the network.

The distance information supplied by Weighted TTL has the potential for other uses besides charging. The new TTL value at a given point is a measure of how much resources or expensive the packet has been. One could imagine a scheme where packet dropping in the event of congestion would be done on cheaper packets first since an expensive packet has used more of the global network resources.

With a facility to gather distance information, we may then consider controlling factors such as user behaviour. How we could then implement such control is left for future study.

Acknowledgements

We would like to thank Gerd Aschemann, Jürgen Jähnert, Trond Dragland, Thomas Plagemann, Goetz Pfeiffer and Stephan Robert for facilitating our Internet studies. Special thanks for the fruitful discussions with Ljubica Blazevic, Jean-Yves Le Boudec, Heiko Boch and Thorsten Kurz, but not forgetting to thank Sam Manthorpe, Claudia Rominger and Swiss fondue for helping to develop this idea. Special thanks to Burkhard Stiller for his comments and review.

References

[1] K. Nichols, V. Jacobson, and L.Zhang, "A Two-bit Differentiated Services Architecture for the Internet," Nov 1997. Internet Draft: draft-nichols-diff-svr-arch-00.txt.

[2] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource Reservation Protocol (RSVP) Version 1 Functional Specification," May 27, 1997. Internet Draft: draft-ietf-rsvp-spec-15.txt.

[3] D. Clark and J. Wroclawski, "An Approach to Service Allocation in the Internet," July 1997. Internet Draft: draft-clark-diff-svc-alloc-00.txt.

[4] Z. Wang, "User-Share Differentiation (USD) - Scalable bandwith allocation for differentiated services," Nov. 1997. Internet Draft: draft-wang-diff-serv-usd-00.txt.

[5] W. Almesberger, T. Ferrari, and J.-Y. L. Boudec, "Scalable Resource Reservation for the Internet," Nov. 1997. Internet Draft: draft-almesberger-srp-00.txt.

[6] J. Crowcroft and P. Oechslin, "Differentiated End to End Services using a Weight Proportional Fair Sharing TCP," December 18, 1997. University College of London.

[7] K. Kilkki, "Simple Integrated Media Access (SIMA)," June 1997. Internet Draft: draft-kalevi-simple-media-access-01.txt.

[8] J. Heinanen, "Use fo the IPv4 TOS Octet to Support Differential Services," Nov. 1997. Internet Draft: draft-heinanen-diff-tos-octet-01.txt.

[9] P. Ferguson, "Simple Differential Services: IP TOS and Precedence, Delay Indication, and Drop Preference," Nov. 1997. Internet Draft: draft-ferguson-delay-drop-00.txt.

[10] E. Ellesson, "A Proposal for the Format and Semantics of the TOS Byte and Traffic Class Byte in IPv4 and IPv6 Headers," Nov. 1997. Internet Draft: draft-ellesson-tos-00.txt.

[11] S. Blake, "Some Issues and Applications of Packet Marking for Differentiated Services," Dec. 1997. Internet Draft: draft-blake-diffserv-marking-00.txt.

[12] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification." Internet Draft: draft-ietf-ipngwg-ipv6-spec-v2-01.txt.

[13] W. R. Stevens, *TCP/IP Illustrated Volume 1 - The Protocols.* Addison Wesley Publishing Company, March, 1996.

[14] S. Manthorpe, *"Implications of the Transport Layer for Network Dimensioning".* Thesis no 1671, Ecole Polytechnique Federale de Lausanne, 1997.

[15] V. Paxson, "End-to-End Routing Behavior in the Internet," in *IEEE/ACM Transacations on Networking Vol.5*, October 1997.

[16] M. Steenstrup, *Routing in Communications Networks.* Prentice-Hall, Inc, 1995.

[17] T. Maufer and C. Semeria, "Introduction to IP Multicast Routing," July 1997. Internet Draft: draft-ietf-mboned-multicast-03.txt.

[18] S. Herzog, S. Shenker, and D. Estrin, "Sharing the "Cost" of Multicast Trees: An Axiomatic Analysis," in *Proceedings of ACM/SIGCOMM'95*, Aug. 1995.

[19] S. A. Thomas, *IPng and TCP/IP Protocols - Implementing the Next Generation Internet.* John Wiley and Son, Inc, 1996.

[20] N.Askan, P. Janson, M. Steiner, and M. Waidner, "Electronic Payment Systems," 1997. ACTS Project AC026: SEMPER.