Computer Sciences Department

From Dumb Pipes to Rivers of Money: A Network Payment System

Cristian Estan
Suman Banerjee
Aditya Akella
Yi Pan

Technical Report #1635

April 2007

UNIVERSITY OF WISCONSIN MADISON

From dumb pipes to rivers of money: a network payment system

Cristian Estan Suman Banerjee Aditya Akella Yi Pan
Computer Sciences Department, University of Wisconsin-Madison
{estan,suman,akella,yipan}@cs.wisc.edu

ABSTRACT

We propose extending the service interface of IP with the ability to carry small payments from the sender to the receiver and to ISPs on the path. This allows an endpoint to purchase improved end to end service for the packets it sends and receives. These payments are a new source of revenue for ISPs and content providers. We argue that the fine grained control and the additional revenue streams enabled by our proposal may provide a solution to the network neutrality debate without either stifling ISP growth or unfairly taxing content providers.

To support this extension of network functionality, we propose a multi-ISP accounting framework which is efficient, convenient, and requires only limited trust. The accounting header for data packets is typically no larger than 20 bytes and data plane processing is not more complex than today – no authentication or cryptography are needed. All actual payments are large and happen between neighbors who already have contracts. A scalable accounting system owned and operated by ISPs provides audit trails that expose cheating attempts. Simple mechanisms prevent malicious hackers from stealing money from hijacked computers.

1. INTRODUCTION

In its early years as a commercial network, the Internet went through a period of spectacular growth fueled by large investments and increasing numbers of users. But with the number of Internet users reaching demographic limits in developed countries (83% of active Internet users in U.S. have broadband [30]), and with ever increasing traffic due to file sharing and streaming media [12], network growth is not as smooth as it used to be [23]. The recent network neutrality debate highlighted some of the contentious questions related to the future growth of the network. Since there are profits to be made from services offered through the Internet, it is not surprising that ISPs and content providers tussle for a larger slice of the Internet pie. But the bluntness of some of the solutions proposed can cause much unintended damage. If ISPs relegate all traffic from certain content providers to a low priority status, they limit user choice and make the Internet less useful [10]. Legislation forbidding ISPs to do any traffic prioritization may hinder applications sensitive to service quality that could drive the future growth of the Internet [22, 16]. We propose a mechanism that allows endpoints to purchase improved service from all ISPs on the path of specific traffic flows. This preserves the users' right to choose, and also gives ISPs a new revenue source to support network growth. Such capability can do more than just lead to a more equitable distribution of costs, revenues, and network services, it can also increase the size of the "Internet pie" by enabling new uses of the Internet not possible today.

New service example: A user watching some streaming media can have the image freeze repeatedly due to congestion at a peering link of a remote ISP. Even if the user is willing to pay a modest fee (say a cent per minute) to improve the quality of the transfer and the ISP is willing and able to provide this improved service (say by mapping the transfer to a different diffserv class), they cannot make such an arrangement today. The user has no feasible way to pay the *remote* ISP for non-default services. But if network endpoints had the ability to purchase improved service from all ISPs on the path, the user could just push a "buy better quality" button on the media player and get the viewing experience he desires. The end user benefits from the ability to select non-default network services, and the ISPs obtain a new revenue stream.

Another new network service: For the same scenario, the user may also be willing to pay a small fee (say, a cent per minute) to get a version of the video stream without embedded advertisements, and given the low prices of advertisements, the content provider may be willing to take the offer. If network endpoints had the ability to easily make such small payments to other endpoints, this could turn into a new revenue source for content providers and the ISPs who provide the service of intermediating the payments.

How big can the new revenue streams be? In the U.S. alone if the existing Internet users (estimated number of users is 211 million [2]) spent on average 2 dollars per month for better service, it would add up to 5 billions of dollars per year. With expensive services for bandwidth-intensive applications such as video conferencing, the revenues may well be much larger.

In this paper we propose *NetPaS* (Network Payment System), a framework for network payments that consists of two core elements: 1) extending the network's service interface so endpoints can make small payments to ISPs on the path and

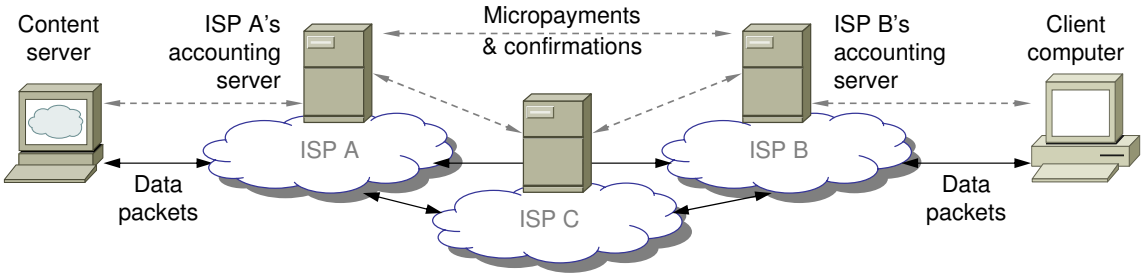

Figure 1: The accounting system is organized as an ISP-operated overlay network.

to other endpoints and 2) a scalable accounting system that ensures that payments reach their intended targets and that blocks misbehavior by hijacked endpoints. Our proposed framework introduces a few radical changes and demonstrating its viability is quite challenging. We could have run various simulations or experiments (say, on PlanetLab) to validate the architecture, but such an exercise would be misleading for two reasons. First, all of our constructs are simple enough to implement and there is much prior work on implementing processing steps of similar complexity to what we propose. Second, such an effort would only validate our assumptions about the system, rather than validating potentially unforeseen *usage* behavior of the system. Hence, such an effort might even instill misplaced confidence in the success of our proposal.

In this paper, our goal, therefore, is slightly weaker. We would like to instill *reasonable doubt* in the readers, that our proposed framework is feasible and trustworthy. We do this through detailed discussions and arguments about *NetPaS*. We also present a few performance measurements for core operations of *NetPaS* to support our arguments about its feasibility and scalability. While some questions remain, such an accounting framework may be an important step in enabling higly desirable outcomes: popular new applications and network services valued by end users, and massive new revenue streams for Internet incumbents and innovative newcomers.

2. TECHNICAL OVERVIEW

Adding payments to packets: Currently the public Internet offers a single type of service: best-effort end-to-end delivery of packets. We propose extending the service interface in two ways: by adding *nanopayments* to data packets to purchase non-default services from ISPs on the path, and by giving endpoints the ability to send *micropayments* to other network endpoints. We discuss possible uses for these extensions in Section 3.

Data packets requesting non-default services include a small additional accounting header which specifies for each ISP on the path the type of service requested and the amount the endpoint pays for the service. Since the price for servicing a single packet is very small, we call these payments to ISPs along the path nanopayments. The ISPs are not obligated to service the packet, but if an ISP delivers it to the next one, the user must pay for the ISP's service. If the end to end service does not meet the endpoint's expectations, it can change the combination of services in future packets, or stop sending altogether.

Special micropayment messages contain an identifier for the receiver, and the amount of the payment. They also contain nanopayments for the ISPs intermediating the transaction. The sender commits to paying all intermediaries and the receiver if the message reaches the correct destination. In this paper we use the term "micropayment" for (small) payments to endpoints and "nanopayment" for payments to ISPs on the path. We use *network payment* to refer to both.

Aggregating small payments: The *NetPaS* accounting system builds onto existing contracts between neighbors. Its core idea is to cumulate nanopayments and micropayments from packets into large payments between neighbors at the time scale of a billing cycle. The accounting system is organized into a secure overlay network whose structure mirrors the relationships between ISPs (see Figure 1). The volume of traffic through this overlay is low as it only carries the micropayment messages and cryptographic *confirmations* for micropayments and for sampled data packets. Border routers of every ISP providing non-default services to data packets interact with accounting servers operated by the ISP.

A simplistic version of the accounting system can rely entirely on *payment counters* placed on links between organizations that operate similarly to the ubiquitous SNMP counters. For example a micropayment message from endpoint B to endpoint A sent through ISPs Y and X (as in Figure 2), will indicate the amount of the micropayment to A, m_A, and the sizes of the nanopayments to X and Y n_X and n_Y respectively (in Figure 2 $m_A = 100,000$ nanodollars, $n_X = 2,000$ nanodollars, and $n_Y = 500$ nanodollars). After the message arrives to A, B would owe Y $n_X + n_Y + m_A$, Y would owe X $n_X + m_A$, and X would owe A m_A. More generally, when an ISP (or the sender) hands over a micropayment message to a next hop ISP (or the receiver) it commits to pay to the next hop an amount equal to the cumulative downstream payments in the message. The payment counters simply cumulate these quantities for the duration of the billing cycle and the amount exchanging hands at the end of

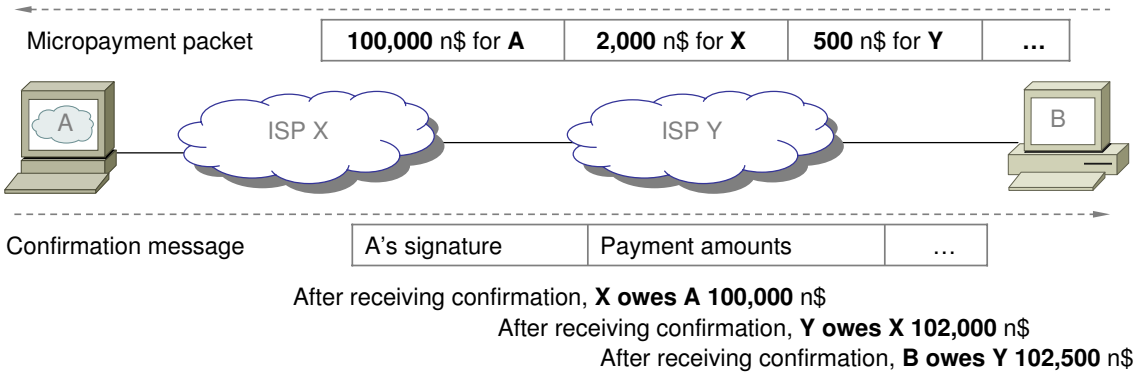

Figure 2: The confirmation for a micropayment is signed by the payee (the receiver).

the month equals the difference of the payment counters for the two directions.

Operating with limited trust: This simple solution has very low overhead, but it is not suitable for an environment with limited trust. The recipient can claim that it never received the micropayment packet even if it did. Either of the ISPs can drop the packet and effectively retain the micropayment. ISPs can also just reduce the size of the micropayment in the packet and retain the difference. To defend against such forms of cheating we introduce confirmation messages: A must send a signed confirmation message to B and the commitments to pay come in effect only when the confirmation message returns (as shown in Figure 2). During normal operation, only B would check the signature, but all ISPs log the confirmation.

For data packets requesting non-default services, each ISP along the path generates confirmations for the service of the previous ISP for a few sampled packets (as shown in Figure 3). The destination confirms only the services of the last ISP, and if the destination does not participate in the system, the last ISP can generate confirmations for its own services. Hence each ISP relies only on the verifiable actions of its neighbors to guarantee that it is not defrauded from payments for its services. Confirmations are generated only for a few sampled packets and each ISP makes independent sampling decisions. By using smart sampling [15] we ensure that the errors introduced by sampling are low and the number of confirmations is proportional to the total value of the network services purchased, not to the volume of traffic. For today's bandwidth prices, a $1,000 commodity PC can handle the confirmations for a peak traffic volume of 15 Gbps. For this PC it takes 30 minutes to confirm services worth $1,000. See Section 6 for more details.

But relying on confirmations alone is not a good solution either. Rogue endpoints and dishonest ISPs can increase their income by generating confirmations for fictitious packets. For micropayments, the solution is that the sender and all ISPs on the path retain a copy of all micropayment packets for a short time and accept a confirmation only if they find the corresponding micropayment packet. For data packets, ISPs on the path compare the volume of confirmations with the volume of payments in the data plane measured with SNMP-like payment counters. For further protection, routers keep a separate counter for each downstream ISP. This way *NetPaS* ensures that no ISP receives payments exceeding the total price of the services purchased from it by the data packets, except for small errors that may arise from sampling. To make these types of checking possible, the overlay ensures that all confirmations propagate on the same path as the data or micropayment packet causing them.

With *NetPaS*, an intermediate ISP needs not authenticate the endpoint purchasing its services, nor does it need to trust that the end user will pay his dues at the end of the billing cycle. The upstream ISP has the obligation to pay for properly confirmed services even if the end user breaks his commitment. This is similar to how an ISP today commits to pay its transit costs to larger ISPs even if some of its users fail to pay their monthly bills. But the problem is much worse if an endhost is hijacked and used to purchase expensive network services or to send many micropayments causing large bills. The measures the ISP can use to protect its clients range from simple limits on the volume of payments that are easy to enforce by routers to discriminating, yet efficient filtering solutions discussed in Section 4.1.2.

3. SERVICES *NetPaS* CAN ENABLE

Network payments are a mechanism which can only be useful if there are network services and applications that build on it. To offer many of these services, one would need to solve open problems other than the problem of payments, and we do not solve them in this paper. Our only claim is that the adoption of *NetPaS* would remove one of the obstacles in the way of these services. The goal of this section is to show what could motivate the deployment of *NetPaS*, not to give a complete solution for implementing the services discussed.

3.1 What ISPs can offer

3.1.1 Better service along the path

The most obvious service ISPs can offer is more favorable

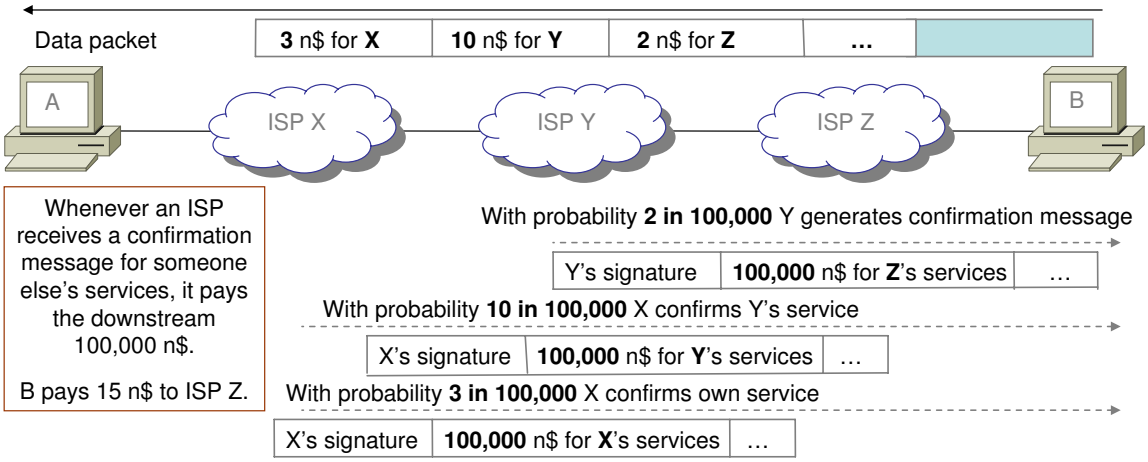

Figure 3: Each ISP on the path of a data packet confirms the service of the ISP immediately upstream. Confirmations are generated only for sampled packets.

treatment of the packets by the routers along the path. There are many ways to organize such an offering, but here we will only briefly sketch two extreme approaches: a lightweight one and one with strong QoS guarantees. The lightweight approach is inspired by Odlyzko's Paris Metro Pricing [29]: the ISP defines multiple levels of "better than best effort" service with progressively higher prices and higher priorities. Since the ISP makes no guarantees of service quality, such a service can be implemented with Diffserv [13], without any admission control or policing. Prior work [34, 37] shows that through a combination of provisioning and pricing, the ISP can ensure that at any time an eager user can find a level of service that is high enough priority to give his application the desired service quality. With this approach the endpoints have to find the right levels of service for all ISPs along the path. Given prior work on ISP-assisted active measurement [36], on using earlier measurements to predict performance [41], and on sharing passive measurement information with other endpoints [35], we believe that it is possible to solve this problem, but do not address it here.

ISPs can also offer strong QoS guarantees to flows of traffic if they implement call admission, policing at the edges and some form of authentication for data packets. Nanopayments can be used to pay setup charges and per-minute charges. When the application has no data to send but wants to retain the reservation, data packets with nanopayments but with no payload can be sent.

3.1.2 *Alternate paths*

ISPs can also offer to carry *NetPaS* data packets along paths not used by best effort traffic between the same endpoints. For example, an ISP would not carry best effort traffic between its peers, because the peers do not pay it to do so. But ISPs may carry packets between their peers if senders use *NetPaS* nanopayments to pay them. It has been shown repeatedly [33, 18, 20] that for many pairs of endpoints, the path taken by best effort traffic is sub-optimal in terms of many performance metrics. Thus data packets purchasing service along an alternate path may experience lower delays, lower loss rates or better bandwidth.

3.1.3 *Better service for incoming packets*

NetPaS can help a user purchase improved services on the reverse path also. If the user trusts the endhost at the other end, she may send micropayments and the other end can spend the amount buying improved services for data packets it sends to the user. But if the other end is not trusted to receive *NetPaS* micropayments (see Section 4.1.3), the user may purchase improved service on the reverse path with the help of the ISP at the other end. The participating endpoint can send a micropayment or a nanopayment to the ISP. This requires the ISP at the other end to keep per flow state, which is feasible at access routers.

3.1.4 *Internet roaming*

NetPaS can also help roaming users if their home ISP and the ISP they connect through provide some simple services. The foreign ISP does not need to authenticate the user, nor does it need to have a pre-arranged roaming agreement with the home ISP. It only needs to allow the roaming user to open a connection to its home agent which authenticates him. The home agent can send network payments to the foreign agent who makes them available to the roaming user. These payments can also cover the cost of best-effort Internet access.

3.2 Uses for micropayments to endpoints

Before discussing the uses for *NetPaS* micropayments it is worth discussing whether they offer a new capability at all. Large Internet companies offer payment services that handle micropayments. For example Ebay's PayPal charges 5% plus $0.05 per transaction for micropayments between its users [32], and Amazon's Flexible Payment System (in

limited beta at the time of submission) charges 20% of the transaction amount, with a minimum fee of $0.0025 for payments below $0.05 and 1.5% +$0.01 for those above [3]. We believe that *NetPaS* can offer two advantages: convenience and coverage. Integrating nanopayments for network services with micropayments for services from endpoints may be convenient to end users. Furthermore, existing payment services such as those above can handle micropayments efficiently only if both ends have an account with them, but *NetPaS* also handles the case when the two endpoints have accounts with different ISPs. We believe that it is more feasible to attain wide coverage by leveraging the ISPs' existing relationships with billions of users than by having a single company cover the entire population. We note here the topology of the overlay that carries micropayments can be richer than the topology of interconnections for data traffic between ISPs. Thus two ISPs that do not connect to transport data can have an agreement for transporting micropayments through a tunnel between their accounting systems. Furthermore organizations that do not offer data transport services (such as Ebay and Amazon) can become part of the overlay for micropayments and intermediate micropayments to and from their users. In conclusion, we believe that *NetPaS* micropayments could significantly extend the abilities of existing micropayment systems.

The most obvious use for endpoint micropayments is to pay for "content" delivered over the network. This is especially true where content is organized as collections of small inexpensive pieces and where some users may not want to commit to a long term subscription. Micropayments may also be used for improved versions of freely available content – a higher quality tune, a video clip with better resolution, a version without advertisements, etc. But this is not the only possible use. Some anti-spam defenses such as "Bonded Sender" [1] were based on the idea of requiring senders of email messages to include a small payment. With *NetPaS* micropayments this could be done as part of SMTP. The ability to incorporate micropayments in future network protocols may lead to surprising new uses for them.

The ability of endpoints to efficiently pay each other can be used for transacting non-network services or goods. For example, it could become possible to buy books on Amazon using *NetPaS* payments. A further extension of this ability arises by turning Internet-enabled mobile devices into digital wallets: the owner of the device could use *NetPaS* payments to purchase a subway ticket, or coffee. Similar initiatives already exist. For example NTT DoCoMo's Osaifu-Keitai [14] service allows cell phone owners to pay for a variety of services, but both the buyer and the merchant need to have a contract with NTT. *NetPaS* could offer broader services by allowing transactions to occur even if the two (mobile) endpoints have contracts with different ISPs.

4. TRUST AND SECURITY

Since *NetPaS* manipulates money, the trust of end users and ISPs is especially important for its acceptance. We discuss here how *NetPaS* protects against stealing, fraud and freeloading. Our discussions of various threat scenarios introduce some elements of the system not described in the technical overview and motivate design choices. Table 1 summarizes the threats discussed. Due to space limits we present some of the threats only in the technical report version of the paper.

4.1 Threats posed by hijackers

Today malicious hackers can hijack many endhosts and small routers and we believe that this threat will persist for a long time. We show how *NetPaS* can ensure that they cannot steal money from the owners of the hijacked systems through network payments.

4.1.1 Stealing content providers' money

A hijacker could take control of a server used by a content provider and leak micropayments. It is easy to defend against this threat by allowing these servers only to receive micropayments, not to send them also.

4.1.2 Stealing end users' money

A hijacker taking control of a home computer with the ability to send micropayments could leak money. In Section 5.3 we describe ways of controlling *NetPaS* payments at the endhost that make it hard for a hijacker to send arbitrary micropayments. But the network also offers protection. It can cap the damage by rate limiting the micropayments. Also, the logs of confirmation messages reveal the destination of the micropayments making it possible to track down the attackers and prove their guilt.

The ISP can also offer a micropayment filtering service to further protect users. The accounting system can turn the ability to send micropayments into an "off by default" [8] feature by only accepting payees with a proper cryptographic certificate. The PKI infrastructure used by secure web sites can be extended for this purpose. The procedure for certifying payees ensures the safety of this solution and multiple procedures with different levels of safety are possible. For example, there may be a certification authority for "merchants with a low number of user complaints" one for "entities who can be held accountable by U.S. law enforcement" and one for "entities from countries where regulations against stealing network payments are enforced". A risk averse user may instruct her ISP to only allow payees with the first type of certificate, while a user who prefers more flexibility may allow all three types of certificate. The micropayment filtering services also protect against phishers impersonating a site that normally accepts micropayments and against attackers that manage to change the payee information on such a site.

4.1.3 Micropayment laundering networks

If malicious hackers could assemble large networks of hi-

Threat	Solution	Discuss
Stealing content providers' money	Disabling the sending of micropayments from servers	4.1.1
Stealing end users' money	ISP filters payees based on certificates and rate limits payments	4.1.2
Network payment laundering	Few computers can both send and receive micropayments	4.1.3
Certified merchants stealing	ISP freezes suspicious accounts	4.1.4
Stealing through nanopayments	ISPs protect customers by blacklisting hackers' ISP accomplice	4.1.5
Payee denies receiving money	Payee must sign confirmation for micropayment	4.2.1
Freeloading	Routers check that nanopayment covers cost of service	4.2.2
Replaying confirmations	Timestamps and unique identifiers for confirmations	4.2.3
Sender disavows network payment	Authenticated channel between sender and ISP	4.2.4
ISPs stealing micropayments	Payee does not disclose private key to ISP	4.3.1
Extra nanopayment confirmations	Neighbor detects using single downstream payment counter	4.3.2
Skim downstream nanopayments	Check signatures in random confirmations, check path in all	4.3.3
Shift downstream nanopayments	Neighbor detects using per-ISP downstream payment counters	4.3.3
Payments bypass middle ISP	Nanopayment confirmations generated by all ISPs along path	4.3.4
Withholding confirmations	Local dispute resolution with neighbor ISP	4.3.5
Incriminating neighbor	Prudent incident response procedures	4.3.6
Compromised accounting servers	ISP limits damage with protection measures in other servers	4.4.1
Floods with high-priority packets	Senders' ISPs limit damage by rate limiting payments	4.4.2
ISPs degrade best effort service	Competition, (threat of) regulation for monopolies	4.4.3
ISP provides bad service	Endpoint switches to other paths	4.4.4
Confirming dropped packets	Limited by the need to collude with neighbors	4.4.5

Table 1: Threats against *NetPaS*, solutions to them, and sections where we discuss them.

jacked computers with the ability to send micropayments to each other, they could use them to hide their tracks when leaking payments much like they use stepping stones today [42]. Such a network requires computers able to both send and receive micropayments, but we expect that few computers will need to do both. The ability to send *and* receive should be granted only to properly protected systems, and their logs should be scrutinized for signs of laundering.

4.1.4 Trusted merchants turned bad

While the certification process can weed out most of the untrustworthy payees, if it is possible to get a large one-time payoff, some certified merchants may collaborate with hijackers to steal money through micropayments. The operating procedures of the ISP they connect to can offer protection. Note that the turncoat payee only gets the money that results from the scam when the ISP pays the balance of the micropayments at the end of the billing cycle. But the ISP may freeze the account (temporarily) when it sees suspicious increases in the balance and it receives complaints from remote ISPs about micropayment stealing. The existence of such safeguards may be one of the things taken into account when payees are certified.

4.1.5 Leaking nanopayments

Hijacked systems or trojaned applications can send streams of packets purchasing expensive service from a remote ISP that colludes with the attackers. These packets do not flow through the accounting overlay, and bypass all protections against stealing with micropayments. It is possible to preventively rate limit at routers the nanopayments to untrusted remote ISPs. The logs allow the victims' ISPs to detect which ISP colluded with the attackers, and once found it can be blacklisted. Routers can enforce in the data plane that all packets purchasing service from the blacklisted ISP are dropped.

4.2 Threats posed by dishonest endpoints

4.2.1 Payee denies receiving micropayment

The payee may try to deny that it received a micropayment. But *NetPaS* requires the payee to sign the confirmation message. The payee may use an invalid key and claim that someone else generated the confirmation. None of the ISPs on the path of the confirmation message check the signature, nor do they need to have the payee's public key. But the payer normally downloads the payee's public key before sending the micropayment and it can detect if the signature on the confirmation does not match. The payer can ask its ISP to cancel the micropayment and submit the public key of the payee as supporting evidence. All ISPs on the path can confirm that the signature does not match the public key provided by the payer. Since the payee identifier in the micropayment message is a cryptographic hash of the payee's public key, the ISP can check that the payer has the correct public key.

4.2.2 Sender skimps on nanopayments

An overly thrifty end user may put into data packets nanopayments that are smaller than the price of the service requested. But each ISP's border routers would normally check that the nanopayment matches the cost of the service and packets with lower nanopayments may receive an inferior service or be dropped.

4.2.3 Replaying confirmations

A payee (or a dishonest ISP) may replay micropayment confirmations to make the payer pay repeatedly. To protect against this threat confirmations have an identifier which combined with the timestamp and has to be unique for each payer-payee pair. The ISPs discard duplicates and out of date confirmations.

4.2.4 Sender disavows network payments

Since the sender does not sign its data packets purchasing improved service, or its micropayments, he can try to claim not to have sent some of them. If this threat is a concern for the ISP, it can authenticate the user and require that all packets carrying network payments be sent over a trusted channel that makes it hard for the sender to repudiate them. Note that the end user's agreement with the ISP would typically stipulate that the user pays based on the total nanopayments in the packets sent, not based on confirmations. Hence the end user can not contest the amount due for nanopayments.

4.3 Threats posed by cheating ISPs

Since ISPs run the *NetPaS* accounting system they may be tempted to manipulate it to receive payments larger than the cost of services they performed. But most service providers are companies that want to stay in business for decades to come and if they get caught committing accounting fraud the legal and public relations consequences can be dire. Hence *NetPaS* does not use expensive mechanisms to ensure that they cannot commit fraud, but instead makes it easy to expose fraud and track down cheating ISPs.

The confirmations for *NetPaS* nanopayments are signed by both the ISP generating them and the one whose service is being confirmed. The role of these signatures is to discourage fraud and they only become important when tracking down the source of various types of cheating. *NetPaS* does not need a PKI to manage certificates for the keys used by ISPs to sign the confirmations, but each ISP can set up a secure web site with the all public keys it uses. This makes it easier to establish the ISP's innocence in some fraud scenarios.

4.3.1 Stealing micropayments

ISPs may try to steal micropayments by not forwarding them towards the payee and generating a confirmation. But the payer will detect the signature mismatch and can convince all ISPs upstream of the fraudster that the micropayment should be canceled.

4.3.2 More nanopayment confirmations

A cheating ISP can introduce into the accounting system extra confirmations for its services or for those of ISPs downstream. If this goes undetected, the cheater effectively steals the amount in the confirmations from the the ISPs of the senders of the fraudulently confirmed packets. But the ISP upstream of the cheater keeps payment counters that signal that the volume of payments in the confirmations does not match nanopayments in the packets and the fraud is detected quickly. To detect this type of fraud a single payment counter for all downstream nanopayments suffices. Note that since the confirmations are generated at random, ISPs can cheat a very small amount without setting off alarms due to mismatches with payment counters. But for realistic settings, increases as small as a tenth of a thousandth of the volume of nanopayments can be detected (see Section 6). Hence for the rest of this section we will assume that comparisons against payment counters are reliable indicators of fraudulent increases in confirmation volumes.

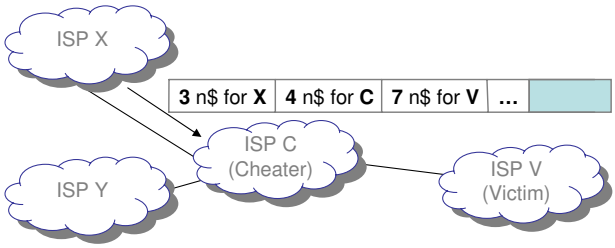

Figure 4: ISP C may try to embezzle (part of) the nanopayment to ISP V it receives from ISP X.

4.3.3 Embezzling downstream nanopayments

In *NetPaS*, ISPs are entrusted with the payments for the services of other ISPs further downstream. One concern is that a cheating ISP can embezzle some of that money. Since this does not create any mismatches between the total volume of nanopayments in the packets received by the cheater and the volume confirmed, embezzlement cannot be detected by the upstream using a single payment counter. We use the example from Figure 4 to show how *NetPaS* protects against two types of embezzlement.

Skimming downstream nanopayments involves dropping packets but still generating confirmations for the services they purchase. This way the ISP does not need to pay the downstreams, but it gets paid by the upstream for its services. Another variant of this type of embezzlement is for the cheater to reduce the downstream nanopayments (which may result in inferior downstream service), but put the original amount in the confirmations[1]. For the example from Figure 4, the cheating ISP C could change the account-

[1]Since we use sampling probabilities proportional with the amount of the nanopayment, the cheater would also need to generate extra confirmations to collect the difference between the original and the modified nanopayments.

ing header of the packet it receives from ISP X so that the nanopayment to the victim ISP V is reduced to 2 nanodollars and generate a fraudulent confirmation for 5 nanodollars' worth of services from V. If ISP C uses a fake key to sign the fraudulent confirmation, random checking of signatures at ISP X can expose it. If ISP C replays an old but valid confirmation for services performed by ISP V for a similar packet, the replay protection mechanisms (the checking for recent duplicate confirmation, and the checking of the timestamp in the confirmation) used by ISP X will detect the cheating. A third thing C may try is to send a new and valid confirmation for V's services that applies to a packet that came through Y. Normally this confirmation would not flow through ISP X, so replay protection mechanisms cannot detect that it is fraudulent. But confirmations also include the accounting header of the packet that triggered them, so if C does not change it, X will detect that the confirmation is fraudulent because it is for a packet that did not go from X to C. If C changes the accounting header contained in the confirmation, X can detect the signature mismatch.

Shifting downstream nanopayments is another type of embezzlement which involves changing the recipient of the downstream nanopayments. For the example in Figure 4, if C colludes with Y, it can reduce the nanopayments in the packet going through V, and send a confirmation for its own services using a packet that goes from X to C to Y. Since Y and X collude, all the signatures on the confirmation will be valid, and the confirmation will be a new signature flowing on the reverse path of the data packet being confirmed. Since X keeps a *separate* confirmation counter for each downstream, it will notice that the volume confirmations for C's services exceed the volume of the nanopayments for C in the data packets. Even if C colludes to shift the nanopayments to other ISPs, this form of cheating will be detected because X keeps separate payment counters for each downstream ISP.

For some variants of embezzlement, we rely on checking the authenticity of the signatures in confirmations. This is only possible if the accounting server has and trusts the public keys used by the downstream ISPs signing the confirmations. ISPs can set up a secure public web server with the keys, but they do not have to. ISPs not concerned about being the victims of embezzlement need not go through the minor expense of setting up a secure web site that uses a public key certified by an authority trusted by other ISPs. But if such an ISP confirms the services of an ISP that is the victim of embezzlement, the fraud cannot be detected based on its signatures. This is the reason why *NetPaS* also requires the ISP whose services are being confirmed to sign the confirmation message. So any ISP that is concerned about embezzlement can set up a site with the public keys it uses to ensure that embezzlement is detected as long as there is an honest ISP upstream of the cheater.

4.3.4 *Bypassing an intermediate ISP*

Two colluding ISPs may generate confirmations for the services of an intermediate ISP, but bypass it. For the example from Figure 3, ISP X and ISP Z can collude and tunnel the confirmations for ISP Y directly to Z. Hence Z gets paid for Y's services, but does not pay Y because Y's accounting system never records the confirmation. But Y can easily detect this form of cheating by comparing the volume of its services reflected in packets it delivers to X with the volume of payments in the confirmations. If X under-confirms the actual volume of Y's services, Y can request X to fix the problem and if the dispute escalates it can even stop delivering to X data packets purchasing improved service.

Note that if we used confirmations by endpoints to confirm the service for all ISPs along the path, it would not solve this problem as ISP Y could not tell the difference between data packets lost inside ISP X and data packets for which X tunneled the confirmations to Z. This is the main reason that in *NetPaS* confirmations are generated along the path. But this type of cheating is not a threat for micropayments where we use such endpoint-only confirmations. The path of micropayments does not have to follow the network topology and if X and Z trust each other enough to collude, they should just link their accounting systems and service micropayments without Y's help.

4.3.5 *Withholding confirmations*

An ISP may generate fewer confirmations than due for its upstream neighbor's services with the intent of causing financial loss to it. But this type of cheating can be promptly detected by the upstream with a payment counter for its services reflected in the packets it passes to the cheater. If the problem is not resolved promptly, the victim may stop accepting packet that indicate the cheater as the next ISP on the path.

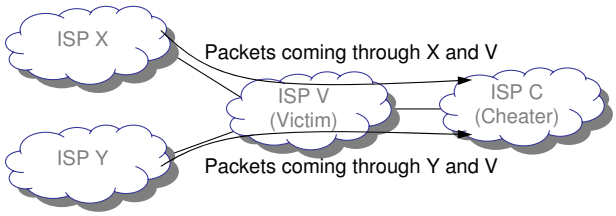

Figure 5: ISP C may try to incriminate its neighbor V of fraud by biasing its confirmations for the two streams of traffic.

4.3.6 *Incriminating a neighbor*

A more insidious way for the cheater to hurt its neighbor is by generating flawed confirmation messages that seem to incriminate its neighbor. More specifically, the cheater can bias the probabilities of generating confirmations in a way that is not creating suspicion when they are handed over to the victim, but it gives the appearance of cheating further upstream. For the example from Figure 5, ISP C would confirm with higher probability the packets that arrive through

X and V and with lower probability those arriving through Y and V in a way that the total volume of confirmations for V's services is normal on the link between V and C. But on the link between X and V there are too many confirmations for V's services. Furthermore all confirmations bear a valid signature from V, so X can suspect that V is cheating. If C publishes its keys and uses fake keys to sign the fraudulent confirmations, V can detect this by checking C's signatures before signing and it gives V the opportunity to stop the fraud. Hence if C publishes its keys it must use them to sign the fraudulent confirmations and it incriminates itself also with this type of cheating. If C does not publish its keys and certificates, X would have no reason to suspect that it is involved in the apparent cheating by V. Thus ISPs that concerned that their neighbors may incriminate them will use whatever means at their disposal (e.g. threatening to severe the link between them) to make their neighbors publish their keys on a secure, trusted site.

But the cheater may be willing to give the impression that it is colluding with the victim, just to cause it damage. To prevent this type of cheating it is not enough for the victim to check whether its downstream neighbor (the cheater) confirms it services in an unbiased way *for each upstream ISP*, but it must check separately *for each upstream path*. In Figure 5, C could give on average unbiased confirmations for packets coming through X and V, but bias them towards one upstream of X (not shown) at the expense of another. Since this type of checking would require too many payment counters in the data plane, we do not adopt this solution. Instead we propose that when an upstream detects what looks like cheating downstream, it give the apparent beneficiary of the cheating the opportunity to rectify the situation before taking punitive measures. Hence when V finds out that X (or some ISP further upstream) detected that C is overconfirming its services, it can specifically measure the traffic it serves for the path in question and ensure that the confirmations do not exceed the value of the nanopayments in the traffic.

4.4 Remaining concerns

Some threats are not eliminated by the mechanisms presented in this paper. We argue that, for various reasons, these threats will not keep *NetPaS* from gaining the trust of users and ISPs: some of them have a small overall effect, some can be eliminated by mechanisms we do not discuss, some can be reduced significantly by mechanisms we discuss, and some are just new forms of threats that already exist in today's Internet.

4.4.1 Compromised accounting servers

If malicious hackers take control of accounting servers, they can disable protections, pollute confirmation logs, and steal a lot of money, especially if the compromise goes undetected for a long time. This is a very serious threat since such an episode can cause massive financial losses to the ISP whose servers are hijacked. We believe it is a manageable one because only a small number of accounting servers need to be secured. Accounting servers may be easier to protect than other types of servers because they need to communicate in well-defined ways with only a small number of computers and routers. The ISPs may also add some internal checks to their networks of accounting servers so that the accounting network is not completely disabled by the compromise of a few servers. Furthermore, the confirmation logs of neighboring ISPs can help investigators track money stolen by the hijackers.

4.4.2 Floods with high priority packets

Hijacked computers can flood a victim with high priority packets carrying nanopayments. Since these packets get better service than best effort packets, such attacks may achieve a more severe denial of service. Rate limiting nanopayments close to the sender may help reduce the magnitude of such floods. We expect that mechanisms for controlling nanopayments at endhosts (see Section 5.3) will much reduce the ability of hijackers to flood. Despite these protection measures, end users whose computers are used in such floods can lose some money. Thus if in-network anti-DoS defenses [28, 26, 38, 40, 6] are available, the ISPs who deploy them can use this as competitive advantage to attract customers from ISPs who do not.

4.4.3 ISPs degrade best effort service

End users may be concerned that ISPs could degrade the best effort service below the capacity of their devices to force them to buy more expensive services. But this is not a new problem and ISPs are already doing this today. Broadband providers offer services with progressively larger bandwidth caps and prices. These caps are implemented by rate limiting traffic below the capacity of access link to the user. We hope that competition and (the threat of) regulation will keep ISPs from unfairly degrading the service offered to best effort traffic.

4.4.4 Will the service match the payment?

End users may be concerned that ISPs could give packets a lower level of service than the one the sender paid for. This may happen if the ISP cannot meet its service level commitments due to capacity constraints. *NetPaS* has a limited view of what the obligations of ISPs are: as long as the packet is delivered to the proper next hop ISP at the proper exchange point, payment is due irrespective of the queuing delays incurred. Note that this problem cannot occur in the lightweight service model described in Section 3.1.1 where the ISP makes no promise about the absolute level of service. Furthermore, the endpoints can stop sending if the service quality is worse than expected, and if alternate paths are offered, they can shun the offending ISP in future transfers.

4.4.5 Confirming lost packets

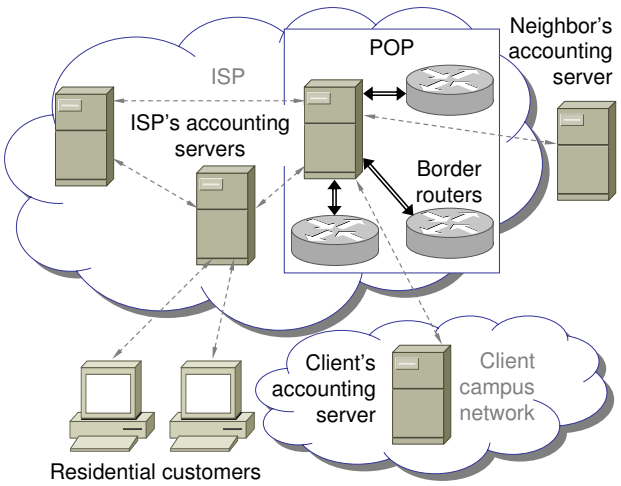

Figure 6: Each ISP's accounting network contains multiple accounting servers. Each POP has at least one accounting server that interacts with multiple edge routers.

There is one type of fraud an ISP can perpetrate without detection by its neighbors: it can generate confirmations for packets it looses. Since the downstream has to sign the confirmations and upstreams may detect fake signatures, the cheater has to conspire with its downstream. But lost packets should be rare as endpoints would cease to send or switch the service mix when packets do not make it to the other end. Also, this type of cheating requires tight coordination across organizational boundaries, and the ISP's neighbors may actually be its competitors, so such collusion would probably be uncommon. The last ISP can cheat this way without colluding with anyone as it typically confirms its own service. But this does not give it an incentive to drop packets, so even if this form of cheating becomes widespread it will not cause a degradation in service.

5. SYSTEM DESCRIPTION

We describe here the *NetPaS* accounting system together with the data plane changes to routers required to deploy it. Figure 6 sketches the structure of an ISP's accounting system. The resource constraints that informed the design of *NetPaS* are as follows: ensuring that packet processing tasks in the data plane can be implemented cheaply, minimizing the bandwidth overhead in terms of extra headers on data packets and extra packets, minimizing the processing requirements at the accounting servers due to cryptographic operations, and minimizing the amount of storage required.

5.1 Data plane requirements for routers

NetPaS only requires data plane changes to edge routers[2] that handle packets requesting non-default services. If an ISP offers such service only to some of its customers or it has some neighbors with which it does not exchange such packets, the corresponding edge routers need not be modified. The packets requesting non-default services have an accounting header between the IP header and the data link header. Edge routers examine this header and make forwarding decisions based on it. How the ISP ensures that the packets receives the level of service it purchased depends on how the ISP structures its offerings (see Section 3.1.1). One possibility is to move these packets between edge routers through IP or MPLS tunnels using diffserv scheduling at internal routers.

[2]In this paper we use the term edge router do refer to all routers that have a link to a customer or another ISP. This includes a variety of border and access routers.

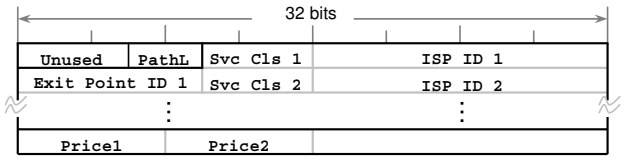

Figure 7: The accounting header specifies for each ISP on the path what services the data packet purchases and the price the sender pays.

The accounting header shown in Figure 7 lists all ISPs on the path of the data packet that offer it non-default service. For each ISP, a 6-bit service class identifies the type of service purchased, and for all but the last, the sender can specify the exit point in a 10-bit field. The exit point identifiers are local to each pair of connecting ISPs, and they indicate the peering point or the city in which the packet is handed over to the downstream ISP. If the sender leaves the field 0, the upstream can pick the most convenient exit point. For each ISP, the sender also specifies the price it pays for the service as a floating point number using 3 bits for the exponent and 5 for the mantissa. Thus the difference between consecutive values that can be represented is between 6.67% to 12.5% and the ratio between the smallest and the largest is almost 2^{32}. The size of the accounting header is 5 bytes per ISP rounded up to the nearest multiple of 4.

Data plane processing steps: Edge routers must perform the following eight *NetPaS*-specific processing steps. The first five are needed only at the ingress, the next two only at the egress, and the last step is required at both when suspected fraud is being investigated. 1) The router samples incoming packets based on the size of the nanopayment to the upstream ISP. It sends the accounting header and next 40 bytes of the sampled packets to the accounting server which generates a confirmation. 2) It drops the packet if any of the downstream ISPs are blacklisted. 3) It applies protective rate limiting based on total of remaining nanopayments using token bucket. 4) It looks up the forwarding action (e.g. the tunnel to send it through and the diffserv codepoint) and associated price information for the ISP's service. If the ISP is not the last one, it uses the next ISP, the exit point identifier, and possibly the service class for this lookup. If no value for the exit point is specified, the router picks the most con-

venient exit point and enters its identifier in the accounting header. If the ISP is the last one, it uses the IP destination address and possibly the service class. 5) The router checks the nanopayment for the ISP to ensure that it covers the cost of service. 6) For outgoing packets, the router updates the payment counter for its services and the one for total downstream payments. 7) It updates the payment counters for individual downstream ISPs. 8) If the packet belongs to one of the paths under investigation, the router updates a payment counter or sends the headers to the accounting server with some probability.

Feasibility of implementation: Step 1) is similar in complexity to operations standardized by IETF's PSAMP working group which generalize widely supported measurement features like the sampling used by Cisco's Netflow. Since we identify ISPs using a 16 bit number, the step 2) can be implemented through 64K flip-flops read in parallel and some combinatorial logic. Step 3) requires simple processing similar to what is implemented in today's routers. The only difference is the long timescale it operates at, so the number of available tokens for each interface should be saved into non-volatile memory when the router reboots. Step 4) for intermediate ISPs operates on a 32 bit quantity that is more structured than IP addresses and the corresponding forwarding table will also be smaller than the IP forwarding table. Hence the implementation cost of this step will be smaller than that for an IP lookup. Step 5) uses price information from the forwarding table. It is easy to implement prices that depend on the time of day by updating this information. We envision that rather than a fixed price per packet, ISPs will want prices that also depend on the size of the packet and on the total of the downstream nanopayments (which determines the load on the accounting system due to confirmations arriving from downstream). Since the path length cannot exceed 16, this is easy to compute using multiple integer adders. Multiplying the size with the per byte cost and the downstream total by the corresponding constant can be simplified by using costs that are powers of two (or small integers times powers of two). Step 6) is similar to updating the ubiquitous SNMP counters.

Steps 7) and 8) can be implemented in the data plane as described above. They may be expensive to implement at the highest speeds as up to 16 counters need to be incremented per packet for step 7) and step 8) requires relatively complex comparisons similar to those in packet classification. But both these steps are required only for detecting downstream fraud based on comparisons with confirmations triggered by very few sampled packets. Without much loss of accuracy for these tests, implementation costs can be reduced by performing steps 7) and 8) on many sampled packets processed locally with a "slow path" processor.

5.2 Accounting servers

The accounting servers perform five functions: generating confirmations for nanopayments, moving micropayments and confirmations, collecting billing data, detecting cheating, and investigating suspected fraud. The configuration information for an accounting server lists its neighbors, their password or key used for authentication, the relation of each neighbor to the server, and the routers the server is responsible for. All links of the accounting overlay are secure and both ends authenticate each other. The three main types of messages carried by the accounting overlay, shown in Figure 8, are the nanopayment confirmation, the micropayment, and the micropayment confirmation.

Nanopayment confirmations are generated based on packets sampled by edge routers. The accounting server responsible for the edge router builds a confirmation, signs it, and sends it to its overlay neighbor from the ISP whose service is being confirmed. This neighbor (the beneficiary) checks the signature and adds its own. The accounting servers move the confirmation towards the packet's source. Upon receiving the confirmation, each ISP compares the payment volume in the confirmations with the total volume of nanopayments for the beneficiary ISP reflected by the payment counters collected from routers. These servers also compare the confirmations against a buffer of earlier confirmations to detect duplicates. Upstream ISPs also randomly check the signatures of some confirmations.

Micropayment messages enter the accounting overlay from authenticated endpoints. The first accounting server performs rate limiting checks and, for some users, a cryptographic verification of the certificates of the payee. Once a payee is deemed safe, its identifier is cached and future micropayments do not trigger cryptographic processing. The accounting server obtains the payee's certificate from a public repository run by the ISP of the payee. Each accounting server on the path of the micropayment stores it (except for the description field) together with a pointer to the previous accounting server. Whenever a micropayment message enters an ISP, the accounting server checks whether the nanopayment to the ISP is sufficient. The micropayment may also be dropped if the total value of the micropayments from the upstream ISP reaches its "credit limit". The last ISP finds the endpoint associated with the payee identifier based on an earlier registration.

Micropayment confirmations are generated and signed by the payee. Each accounting server checks the received micropayment confirmations against the stored micropayments to ascertain their validity. They are also checked against recent confirmations to detect duplicates. Accounting servers do not check the signatures in micropayment confirmations, but if the payer finds the signature to be invalid and notifies its ISP, the micropayment is canceled along the path.

Accounting servers store micropayments and confirmations in hash tables for some time. To limit the memory required for these tables, *NetPaS* imposes expiration times based on the timestamps in these messages: expired messages are discarded and ignored. If a micropayment is discarded, nobody owes anyone any money, but if a confirma-

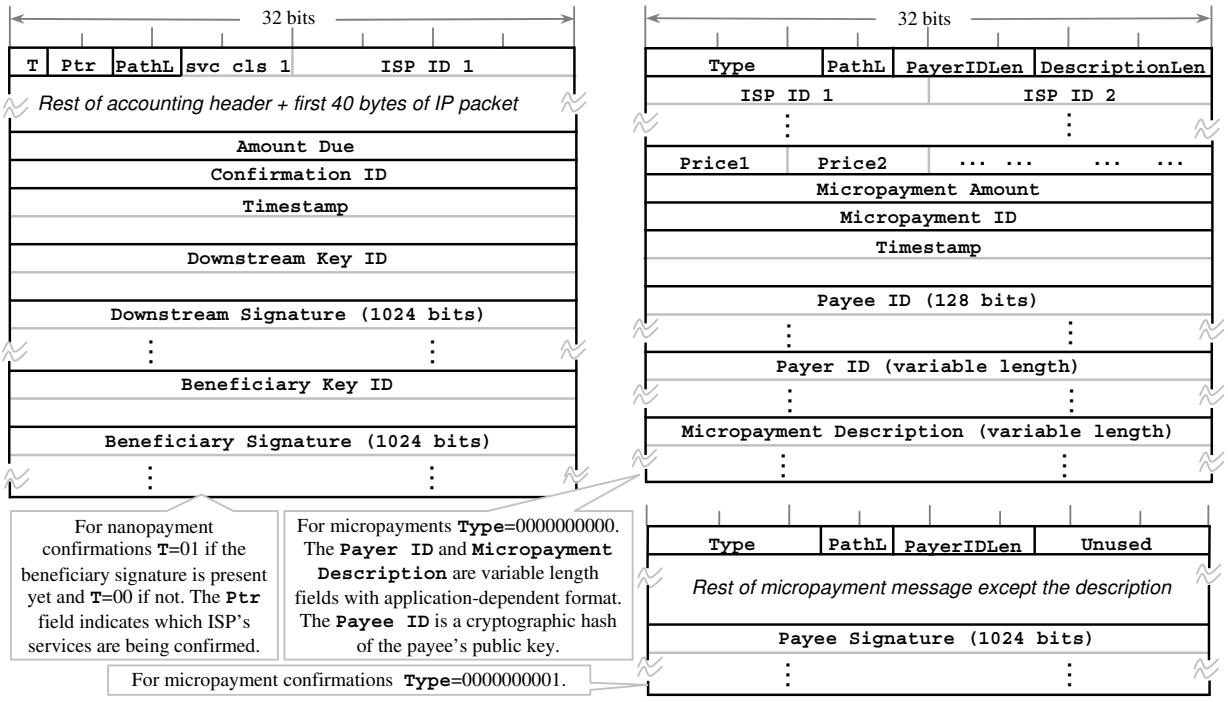

Figure 8: Accounting overlay messages: nanopayment confirmations, micropayments, micropayment confirmations.

tion is discarded, the ISP discarding it must pay the downstream without getting paid by the upstream. An ISP receiving an expired confirmation need not pay for it. If the expiration timeout were the same for all ISPs, the ISP receiving a confirmation about to expire would be forced to accept it without having enough time to get it to the upstream. We used staggered timeouts with each ISP on the way (and for micropayments, the payee also) getting an extra second which is enough to resend the confirmation a few times if it gets lost due to a random drop. Hence a payer making a micropayment through two intermediate ISPs knows that in the unlikely event that it does not receive a confirmation within 5 seconds, it will not be charged. Similarly, the payer has one second to detect a signature mismatch in the micropayment confirmation, after that, its ISP will not cancel the micropayment.

Once confirmations need no longer be buffered in memory, they go to persistent storage for a few hours or days so that they are available if a fraud investigation is started. They need not be stored until the end of the billing cycle as compact per-outside-link summaries suffice. The only piece of information that is absolutely needed in the summary is the balance of payments to and from the neighbor at the other end of the link over the time period covered by the summary.

5.3 Requirements for endpoints

Endhosts need new software to take advantage of the widening array of services enabled by *NetPaS*. Here we touch on three important questions that fall outside the technical details of the accounting system. How do endhosts learn about the available services? How do they decide what service is worth its cost? How can we protect network payments from hijackers taking control of the endhost? We discuss possible answers, but more research is required before actual solutions can be built.

An endhost module we call the *network cartographer* keeps track of the available services and assembles the combination of services from various ISPs that gives the desired end to end service quality. If it considers only improved service on the same path as best effort traffic, the network cartographer can be relatively simple. BGP routing tables from the ISP connecting the endhost can give the AS path to the destinations of interest. The menu of services offered by each ISP can be downloaded and cached. If the endpoint also wants to consider non-default paths, the cartographer needs to acquire more topological information. If ISPs take a lightweight approach to service quality, the cartographer also needs to keep track of current traffic conditions along the paths of interest. A combination of reporting by ISPS, active measurement and passive measurement can give the cartographer the information it needs. Information sharing with other endhosts in the same enterprise, or with endhosts throughout the Internet organized in a peer to peer network can improve the efficiency of the network cartographer. Many ideas from proposals that explore giving endpoints control about the path of the traffic [39, 24, 5] may be adapted.

For any given transfer, the network cartographer may assemble a range of end to end service alternatives with progressively better quality and higher price. The role of the *network secretary* is to decide which if any of the alterna-

tives best fits the user. We envision this secretary as a software module at the endhost that relies on a combination of pre-programmed heuristics, user preferences stored by various applications, and direct input from the user to make decisions. As the secretary learns from the user's answers, it becomes less and less obtrusive. The secretary can also take over decisions about small micropayments eliminating the mental burden of making decisions about small purchases. For example a secretary may end up with rules such as: the user is always willing to pay up to 50 cents per minute to achieve good quality videoconferencing when talking to people from the "business partners" contact list, but prefers to be prompted if the cost of videoconferencing with someone from the "family" contact list exceeds 10 cents a minute; the user is willing to pay 5 cents for comics from any site, and is always willing to pay for articles from "The Economist".

The ISP protects to a large extent its users from malicious hackers leaking network payments from endhosts after they is hijacked. But finer control can be achieved by protecting the network secretary from the hijackers and giving her the ability to block micropayment messages and data packets with nanopayments. One solution is to interpose it between the hijackable endhost and the network by having it reside on the modem connecting the endhost, or by running it on the endhost but outside the virtual machine in which the vulnerable operating system runs. Another solution is to run it on a protected external device connected to the computer (say a USB dongle) and have it sign all packets and messages with network payments. Session keys relying in efficient symmetric cryptography can be used, with checking occurring at the ISP's accounting server for micropayment messages and at the ISP's first router for data packets purchasing non-default service. Since the secretary must interact with various modules and applications on the endhost, it may also be vulnerable to hijacking, but since it implements much fewer functions than the operating system and the applications on the endhost, it is much easier to secure.

5.4 Customers with large networks

Enterprises with campus-wide networks or large data centers can interact with their ISP differently than individual end users. They can run accounting servers of their own. This gives them a centralized point for enforcing the organizations' rules on the uses of network payments. Also the ISP may delegate to them the authority of confirming its services for data packets sent to the organization's address space. While such clients cannot receive nanopayments, it is possible for them to make an agreement with their ISP to transfer portions of the nanopayments to certain service classes. Thus services such as the mirroring of nanopayments for purchasing improved services on the reverse path can be implemented inside large clients' networks.

6. EVALUATION OF FEASIBILITY

Deploying and running an *NetPaS*-like accounting system involves expenses on hardware, software, personnel, power, cooling, and probably a few more. Most of these costs we cannot yet estimate accurately. In this section we focus on the cost of hardware: routers and accounting servers. Since the processing steps routers edge routers need to perform in the data plane are not more complex than for today's routers, we expect that router costs will not go up significantly. For the accounting servers, two types of resources are critical: processing power for cryptography and memory for buffering confirmations and other data. Since only summaries of confirmations need to be stored for a long time, hard disk space is not an issue. By comparing the cost of these resources against the rate at which they can generate income for the ISP we will see that even for conservative configurations they "pay for themselves" in minutes.

Server processor	Performance (confirmation/min)		Server list price
	sign	sign+verify	
Pentium Core2 Duo 2.66GHz	49,232	37,336	$1,095
UltraSparc T1 8 core 1.00GHz	634,334	500,808	$4,945

Table 2: Performance using 1024 bit RSA keys.

Parameters: The sampling rate and the size of the smallest micropayment affect the load on the accounting servers. We choose a hundredth of a cent for the size of the smallest micropayment, which is much smaller than needed for the uses we can currently think of. We choose the sampling rate to ensure that the difference between the total price of the services the ISP performed P and the amount it is paid $\widehat{P}$, is relatively small. The parameter defining the rate of smart sampling [15] is a threshold T, and on average a new packet is sampled whenever the total amount of new nanopayments to the ISP reaches T dollars. We set T to a tenth of a cent, so for every dollar's worth of traffic we will get around $1,000$ confirmations. The standard deviation of the amount the ISP is paid is $SD[\widehat{P}] \lessapprox \sqrt{PT}$. For a small ISP making $P =$\$$100,000$ per month on improved services the deviation is $SD[\widehat{P}] =$\$10, or 0.1%. For a larger ISP making 10 millions the deviation is 0.01% or \$100. We consider such deviations acceptable.

Processing power: The most processing-intensive operation done by accounting servers is the cryptographic signing of the confirmations for nanopayments. The ISP whose services are confirmed performs one verification and one signing, and its downstream one signing. In our experiments reported in Table 2 we measured the rate at which two different processors can generate confirmations. Given that each thousand nanopayment confirmations earn the ISP a dollar, we see that at full utilization, both systems generate enough money to cover their hardware cost in minutes (30 minutes for the Pentium, 10 minutes for the SUN processor which

has built-in support for cryptography). The only other cryptographic operation the accounting servers perform regularly is the verification of payee certificates when a micropayment to a new payee shows up. Assuming that an accounting server verifies 100 new payees per month for each of the $10,000$ end users it handles, it needs only do 1 million verifications per month which does not pose a problem.

Memory: The most memory-intensive operation done by accounting servers is the storage of confirmations (and micropayments) in large hash tables to protect against replay attacks. With an average path length of 2 for a micropayment, an entry needs to be stored up to 7 seconds. With an average path length of 4 ISPs for packets purchasing improved service, the nanopayment confirmation needs to be stored up to 4 seconds. To be conservative, we assume that both types of confirmations will be stored for one minute. Assuming that the hash tables can store 2 confirmations/KB of memory, we find that at full utilization, the accounting server can process around 2 million confirmations per minute for every GB of memory. But how much money does an intermediate ISP receive for processing a confirmation? Assuming that for micropayments its fee is 10% for the smallest possible values, a confirmation brings a thousandth of a cent (the smallest micropayment is a hundredth of a cent). The intermediate ISP does not get paid directly for transporting nanopayment confirmations, but it can include in the cost of servicing data packets a modest surcharge of 1% of the downstream nanopayments. Since each nanopayment confirmation confirms a tenth of a cent worth of downstream service, the ISP would receive a thousandth of a cent. Hence at full utilization, the accounting server earns $\$20$ per minute for each GB of memory, so given memory prices of under $\$100$ per gigabyte [21], the accounting server's memory pays for itself in at most 5 minutes. Accounting servers verifying payee certificates also need an in-memory cache of payee identifiers. But since these are 32 bytes each, even 100 distinct payees for each of $10,000$ users only adds up to 32 MB of memory.

Floor space: In many data centers and POPs floor space is a scarce resource. We compute the number of accounting servers needed based on the peak traffic rate and the peak cost of servicing traffic. Assuming ISPs charge 18 cents/GB (the highest retail bandwidth price used by Amazon's Elastic Compute Cloud), and that traffic is symmetric, the commodity server from Table 2 can handle the nanopayment confirmations for up to 15.7 Gbps in both directions, and the SUN server which fits into one rack unit for 207.3 Gbps. Given the sizes of routers able to handle such volumes of traffic, the accounting servers account for a small fraction of the data center floor space.

7. RELATED WORK

Digital cash and micropayments are the subject of a vast and growing body of work that centers around on cryptographic algorithms [11, 27, 9]. Anonymity and privacy are two major concerns of this line of work, and *NetPaS* differs by offering weaker guarantees as the endpoint's ISP has a detailed view of the endpoint's spending. However, the identity of the users is not disclosed by *NetPaS* to the other endpoints they communicate with, or to other ISPs participating, hence we believe that the degree of privacy *NetPaS* offers is sufficient for most scenarios. The big advantage of *NetPaS* over solutions from this line of work is that it requires no cryptography in the data plane at all, hence it is feasible to associate very small payments even with individual data packets. Furthermore the extra visibility the ISP has allows it to implement payment filtering that protects against stealing by hijackers.

Payment services such as Ebay's PayPal [32] and Amazon's Flexible Payment System [3] are more akin to *NetPaS*. *NetPaS*'s main advantage is that it does not require all participants to set up accounts with a single organization and transactions are possible between participants with accounts at different ISPs. Furthermore, *NetPaS* does not require any authentication in the data plane inside the network.

Congestion-based pricing for the Internet has been considered in simplified settings [25, 19, 31, 29]. In MacKie-Mason and Varian's "smart market" proposal [25], users include "bids" within packets which indicate their maximum willingness to pay the ISP for access. In Odlyzko's Paris Metro Pricing [29], an ISP network is divided into several service classes each offering best effort service but at different prices. Similarly to these proposals, *NetPaS* allows end users to purchase improved service during times of congestion. Unlike them, we focus on the multi-ISP case and propose a safe and trustworthy accounting framework that can intermediate payments to remote ISPs.

Feldman et al. [17] discussed the use of recursive contracts to achieve good service quality along a path with multiple ISPs which is similar to how *NetPaS* composes non-default service by individual ISPs. Their focus is on a game-theoretical analysis of how the contracts affect ISPs' behaviors (dropping or not dropping traffic) and prices, whereas our focus is on building an accounting framework to ensure accurate charging.

Argyraki et al. proposed [7] an *accountability* framework for Internet traffic. Their feedback reports on individual packets are similar in some ways to the confirmation messages we propose. But since *NetPaS* solves another problem we use confirmations very differently. Andersen et al. proposed a different type of accountability framework [4] and our use of self-certifying payee identifiers is similar to their self-certifying addresses, but other details of the two proposals are very different.

8. CONCLUSIONS

This paper explores an admittedly wild idea: changing the service interface of IP to allow senders to put into packets small payments for the receiver and for ISPs on the path. If used constructively, appealing possibilities open up: new

services can build on the ability to make (small) payments to endpoints and the ability to pay remote ISPs can serve as a key enabler of open, multi-ISP service quality. The resulting applications may be valued by end users and may drive the growth of the Internet by generating massive new revenue streams for ISPs and content providers. If used destructively, consequences can be dire: malicious hackers can steal money from hijacked computers, greedy ISPs can take payments intended for endpoints or other ISPs, end users can loose trust and abandon the network.

To protect against such misuse and chaos we propose *NetPaS*, an accounting system owned and operated by ISPs. Perhaps surprisingly, *NetPaS* manages to reach many important goals: malicious hackers cannot steal money from hijacked computers, fraud by ISPs can be quickly exposed, and the data plane of routers needs not perform any cryptography or authentication. By necessity, we leave many questions open: how endpoints learn about the non-default services, how the system interfaces with end users and applications, how peering strategies are affected, etc. Yet, we hope that our answers to questions we do address would persuade the reader that the wild idea of adding money to packets is worth exploring, and maybe, further down the road, treasure awaits.

9. REFERENCES

[1] Bonded sender. http://www.senderscorecertified.com/.

[2] Internet world stats. http://www.internetworldstats.com/, June 2007.

[3] Amazon. Amazon Flexible Payment Service. http://www.amazon.com/b/ref=sc_fe_l_2?ie=UTF8&node=342430011&no=3435361&me=A36L942TSJ2AJA.

[4] D. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker. Holding the Internet Accountable. In *Hotnets-VI*, November 2007.

[5] K. Argyraki and D. R. Cheriton. Loose source routing as a mechanism for traffic policies. In *Future Directions in Network Architecture*, August 2004.

[6] K. Argyraki and D. R. Cheriton. Active internet traffic filtering: Real-time response to denial-of-service attacks. In *USENIX Technical Conference*, April 2005.

[7] K. Argyraki, P. Maniatis, D. Cheriton, and S. Shenker. Providing packet obituaries. In *HotNets-III*, November 2004.

[8] H. Ballani, Y. Chawathe, S. Ratnasamy, T. Roscoe, and S. Shenker. Off by default! In *HotNets-IV*, November 2005.

[9] M. Belenkiy, M. Chase, C. Erway, J. Jannotti, A. Kupcu, A. Lysyanskaya, and E. Rachlin. Making P2P accountable without losing privacy. In *Workshop on Privacy in the Electronic Society*, October 2007.

[10] V. Cerf and T. Berners-Lee. A guide to net neutrality for Google users. http://www.google.com/help/netneutrality.html, 2008.

[11] D. Chaum. Blind signatures for untraceable payments. In *Crypto'82*, 1983.

[12] K. Cho, K. Fukuda, H. Esaki, and A. Kato. The impact and implications of the growth in residential user-to-user traffic. In *Proceedings of the ACM SIGCOMM*, September 2006.

[13] Differentiated services. http://www.ietf.org/html.charters/diffserv-charter.html.

[14] NTT DoCoMo. Osaifu-Keitai. http://www.nttdocomo.com/services/osaifu/index.html, July 2004.

[15] N. Duffield, C. Lund, and M. Thorup. Charging from sampled network usage. In *SIGCOMM Internet Measurement Workshop*, November 2001.

[16] D. Farber, G. Faulhaber, M. L. Katz, and C. S. Yoo. Common sense about net neutrality. http://www.interesting-people.org/archives/interesting-people/200606/msg00014.html, 2007.

[17] M. Feldman, J. Chuang, I. Stoica, and S. Shenker. Hidden-action in multi-hop routing. In *EC '05: Proceedings of the 6th ACM conference on Electronic commerce*, pages 117–126, New York, NY, USA, 2005. ACM Press.

[18] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. ZhangJ. IDMaps: A global internet host distance estimation service. *IEEE/ACM Transsactions on Networking*, October 2001.

[19] R. J. Gibbens and F. P. Kelly. Resource pricing and the evolution of congestion control. *Automatica*, 35, 1999.

[20] K. P. Gummadi, S. Saroiu, and S. D. Gribble. King: estimating latency between arbitrary internet end hosts. In *Internet Measurement Conference*, November 2002.

[21] John L. Hennessy and David A. Patterson. *Computer Architecture a Quantitative Approach*. Morgan Kaufmann, 4th edition, 2007.

[22] R. Kahn. An evening with Robert Kahn. http://vasarely.wiwi.hu-berlin.de/kahn_net_neutrality_transcript.html, January 2007.

[23] G. Kim. Broadband train wreck. http://www.ipbusinessmag.com/departments.php?department_id=5&article_id=12, June 2007. IP Business.

[24] K. Lakshminarayanan, I. Stoica, and S. Shenker. Routing as a service. Technical report, Computer Science Division (EECS) University of California Berkeley, January 2004.

[25] J. Mackie-Masson and H. Varian. *Public Access to the Internet*, chapter Pricing the Internet. MIT Press, 1995.

[26] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling high bandwidth aggregates in the network. *ACM SIGCOMM CCR*, 32(3):62–73, July 2002.

[27] S. Micali and R. L. Rivest. Micropayments revisited. In *Cryptography Track at RSA Conference*, 2002.

[28] J. Mirkovic, G. Prier, and P. Reiher. Attacking DDoS at the source. In *International Conference on Network Protocols*, November 2002.

[29] A. M. Odlyzko. A modest proposal for preventing Internet congestion. Technical report, AT&T Research Lab, 1997.

[30] Website Optimization. US jumps to 24th in worldwide broadband penetration. http://www.websiteoptimization.com/bw/0708/, 2007.

[31] I. Ch. Paschalidis and J.N. Tsitsiklis. Congestion-Dependent Pricing of Network Services. *IEEE/ACM Transactions on Networking*, 2000.

[32] PayPal. Micropayments. http://www.paypal.com/IntegrationCenter/ic_micropayments.html. January 2008.

[33] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of internet path selection. In *Proceedings of the ACM SIGCOMM*, September 1999.

[34] N. Semret, R. Liao, A. T. Campbell, and A. Lazar. Pricing, provisioning and peering: Dynamic markets for differentiated internet services and implications for network interconnections. *IEEE Journal on Selected Areas in Communications*, December 2000.

[35] S. Seshan, M. Stemm, and R. H. Katz. SPAND: Shared passive network performance discovery. In *Usenix USITS*, December 1997.

[36] J. Sommers, P. Barford, N. Duffield, and A. Ron. Accurate and efficient network-wide SLA compliance monitoring. In *Proceedings of the ACM SIGCOMM*, August 2007.

[37] X. Wang and H. Schulzrinne. Pricing network resources for

adaptive applications in a differentiated services network. In *IEEE Infocom*, April 2001.

[38] A. Yaar, A. Perrig, and D. Song. SIFF: A stateless internet flow filter to mitigate DDoS flooding attacks. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2004.

[39] X. Yang. Nira: a new internet routing architecture. In *Future Directions in Network Architecture*, August 2003.

[40] X. Yang, D. Wetherall, and T. Anderson. A DoS-limiting network architecture. In *Proceedings of the ACM SIGCOMM*, August 2005.

[41] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the constancy of Internet path propertiest. In *Internet Measurement Workshop*, November 2001.

[42] Y. Zhang and V. Paxson. Detecting stepping stones. In *USENIX Security Symposium*, 2000.