Internet Service Delivery Control with Mobile Code

Manuel Günter and Torsten Braun
Institute of Computer Science and Applied Mathematics
University of Berne
Neubrückstrasse 10, CH-3012 Bern, Switzerland
`http://www.iam.unibe.ch/~rvs/`
`[mguenter|braun]@iam.unibe.ch`

Abstract

The trend towards value-added Internet services causes network providers to deploy new network based quality-of-service and security services. Today, however, the customer has only limited means of controlling the service delivery. For example the network security guaranteed by virtual private network providers cannot be checked with a traditional static approach. This paper presents a novel approach for controlling new IP services using mobile code, and motivates the approach with two examples of new IP services proposed by the Internet engineering task force (IETF).

1 Introduction

The rapid growth of the transport capacity of the Internet and the global trend towards liberalisation of the telecommunication market forces the Internet service providers (ISP) to look for new revenues beyond pure connectivity offerings. Therefore, ISPs that control their own network try to introduce new Internet services including quality features such as premium transport or traffic privacy. Since ISPs have control over a (albeit small) portion of the Internet, they can deploy new technologies such as Differentiated Services (DiffServ) or the Internet Protocol Security architecture (IPSec) to enhance their network service. The deployment of such services brings some challenges, for example: How can enhanced services be set up dynamically? How to provide services that demand for collaboration between providers? In earlier work [GBK99] we propose a generic architecture to cope with these problems. However, there are two problems to solve that go beyond the deployment of the service, namely:

1. How to convince the user of the presence of the new quality. For example the privacy of a communication is not easily demonstrated. From the user's

point-of-view this is equivalent to the question how can (s)he control if (s)he gets the quality the provider promised.

2. For services involving several providers the question is how to find out who is responsible when the service quality is less than guaranteed to the customer.

It is in the interests of the customers and of honest providers that the customer is able to verify the steady quality of a service and to locate problems when they occur. We refer to this process as *service delivery control* (SDC).

For today's Internet services, there is only very limited support for service delivery control. If a customer happens to detect a problem (which is usually when the customer needs that service badly and does not get it), phone-calls between administrators, local measurements, and manual browsing of log-files will eventually lead to the identification of the problem source. Unfortunately, it is also not uncommon that the involved parties will suspect each other and repudiate any guilt. Note, that this problem not only concerns the relation between customer and provider but also between providers themselves. It is to be expected that the problem becomes worse when new and more expensive network services are deployed that require provider collaboration.

This paper proposes a generic service delivery control architecture based on mobile code. Mobile code allows us to test the service where it is delivered. Thus misbehaviour can be located, even if the provider that causes the problem tries to hide the tracks. Furthermore, mobile code offers a flexible way to deploy new tests for new network services.

The next section will position our work in the field of mobile code research and motivate our approach. In section 3 we will present the architecture of the infrastructure necessary to perform service delivery control with mobile code. Then, we will back-up our arguments by studying two emerging network services proposed by the Internet Engineering Task Force: section 4 describes a virtual private network service and how the privacy that it provides can be controlled. Section 5 discusses service control for differentiated services across multiple domains. Section 6 describes related work in the area of network measurements and section 7 concludes this paper.

2 Mobility and Service Delivery Control

2.1 Terminology

Today, there are many different trends in the research of mobile code. At the application level the most prominent instance of mobile code is called *mobile agents*

[Whi94, CHK97]. These are program instances that are able to move self-directed through a network to locally perform a task in behalf of their sender. Different mobile agent platforms have been proposed e.g. for the programming languages Java [LO98, VB99, Fün98] and Tcl [Gra98]. The term agent is also occupied by other research communities, namely the artificial intelligence research (intelligent agents) [MJ99] and the software engineering community (software agents) [WJ99]. Both communities have influenced the mobile agent research, so a mobile agent is also a paradigmatic software abstraction and includes autonomous behaviour (intelligence). Mobile agents are proposed for different tasks such as network search (more recently e-commerce [HGF^{+}99]), network management [BGP97] and network intrusion detection [JMKM99].

On the network level, the emerging mobile code technology is called *active networking* [TSS^{+}97, CBZS98]. The mobile code is often referred to as *capsule* and is directly integrated into the network traffic packets. Thus, the code flows directly on the communication path that is subject of the code's computation and it can be executed on a per-packet granularity. Here, the abstraction and intelligence aspect is secondary. The focus is on the interaction with the network infrastructure. Active network packets access the networking functionalities of the routers they pass (e.g. forwarding and routing) and change these functionalities for packets or classes of packets. Furthermore, performance is a crucial issue, since the code should be able to manipulate data at the line speed (in todays backbone network this can be up to several gigabits per second).

Still, there is no solid line between mobile agents and active networking. For example the active networking testbed ANTS [WGT98] can also be seen as a mobile agent testbed, since capsules are Java objects, and the code is not included in network data packets but is dynamically loaded upon need. The approach that we describe in this paper cannot be classified clearly as mobile agents or active networking. Service delivery control agents examine network services down to the structure of forwarded network packets. Furthermore, the SDC agents don't necessarily need to be intelligent. They can just collect whatever data the customer is looking for and send it back to a customer application at home. Also, the performance of the SDC agents is an issue since one of their goals can be to monitor the network at wire speed. For these reasons, SDC agents seem to be an application of active networking. However, the control agents do not necessarily need to travel in capsules but can be transferred out-band of the data traffic, like mobile agents do.

2.2 Why Must the Service Delivery Control be Mobile?

Mobile Code has the questionable reputation of being a solution in search of a problem (J. Ousterhout). In this section we will argue why mobility is necessary for service delivery control agents.

Generic interface. No provider will open its network administration system to the customers. By providing an agent platform at relevant sites (see section 3) the provider can give access in a controlled fashion. From the customer's point-of-view the control agents can perform whatever tests that the customer thinks is necessary. It is easy to introduce new tests for new network services.

Cross checking. A misconducting provider can easily fool a customer that relies on the measurements published by the provider. End-to-end measurements can in some cases indicate to the customer that the service is not delivered as guaranteed. But in the case of a multi-provider service such measurements cannot identify the misconducting provider. SDC agents can be sent out to perform active measurements by producing and measuring traffic at different sites. Mobility allows the agents to virtually 'track-down' the problem source.

In order to bring more substance to these theoretical advantages, we need to specify the architecture of the supporting infrastructure. Then we can demonstrate these theoretical advantages on concrete examples.

3 A Supporting Infrastructure for Service Delivery Control Agents

Like any other network monitoring system, the SDC agents need a supporting infrastructure. In this section we discuss the required components.

3.1 Location of the Control Points

The Internet is a heterogeneous network, it consists of thousands of administrative domains. The interior network of these domains is administered in different ways and consists of different kinds of networking technologies such as Frame Relay, ATM, MPLS or Sonet. This may render the access to the traffic inside of the domain very difficult (e.g. for optically switched technology). The least common denominator is the Internet Protocol (IP). The IP traffic is exchanged between the domains at so-called peering points, according to peering- or service level agreements. While the network engineering and management of the interior network of the domains is usually hidden, the peering points are by their nature open (at least to the peer). For service delivery control the peering points are thus of high interest. Note, that is suffices to track down a problem to a provider. Once the problem is found to relate to a given administrative domain, it is up to its administration to further locate the problem in the inside of their network, using the network management system of their choice. Therefore, the SDC agent nodes should be located at the peering points. This guarantees, that the control has access to the IP traffic and that the control can relate identified problems to a specific provider. Of course, a provider can also offer additional control points in the inside of its network as an additional service to its customers or for its own service control purposes.

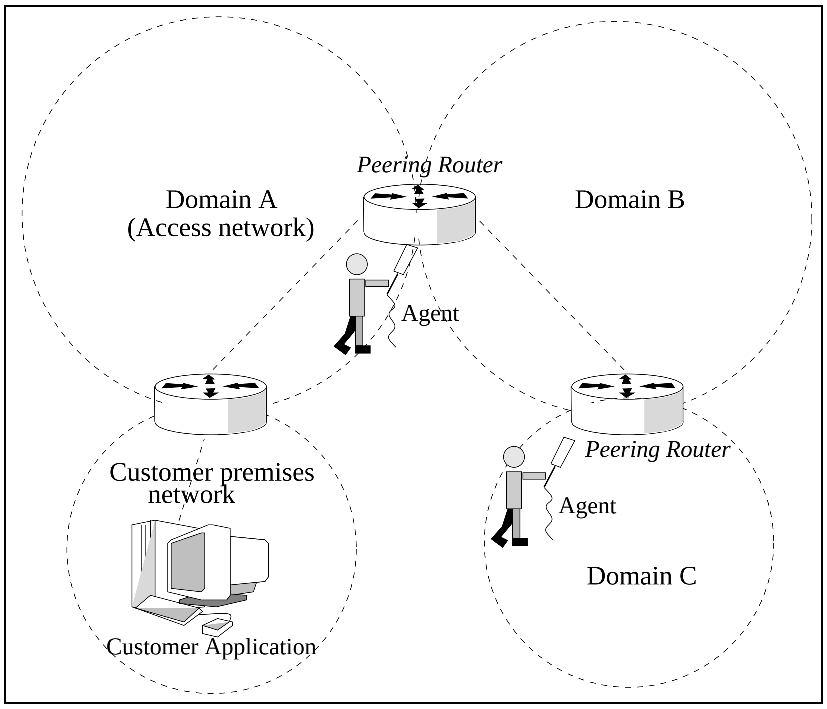

Figure 1: Measuring at peering points.

Figure 1 shows SDC agents which were sent out by a customer application running on a machine owned by the customer. The customer application also coordinates the agents, processes their feedback and forwards the results to the user. The agents migrate to the peering points to perform particular local checks on the service.

3.2 Node Architecture

The SDC agents should be able to perform any kind of passive measurements, however they should not be able to eavesdrop or analyse traffic of other customers. To perform active measurements, the agents should be able to send traffic, but again, this should be traffic related to the specific customer. Spoofing of foreign IP addresses or denial-of-service attacks should not be facilitated. Given these requirements we foresee the following node architecture as depicted in figure 2: At the peering router, there is a *T-component* that serves as a high-performance and configurable packet copying mechanism. The T-component can be configured to copy network packets according to filtering rules based on IP packet information such as source and destination address (see section 3.3). It adds a high-accuracy time-stamp to the packet. For optimisation it can be told to only copy the packet headers. The T-component forwards the requested packet copies to the *Node environment*. Note, that for security reasons the agents do *not* have direct access to the T-component. If the peering router runs under UNIX, the `tcpdump` [JLM89] command can be used as T-component.

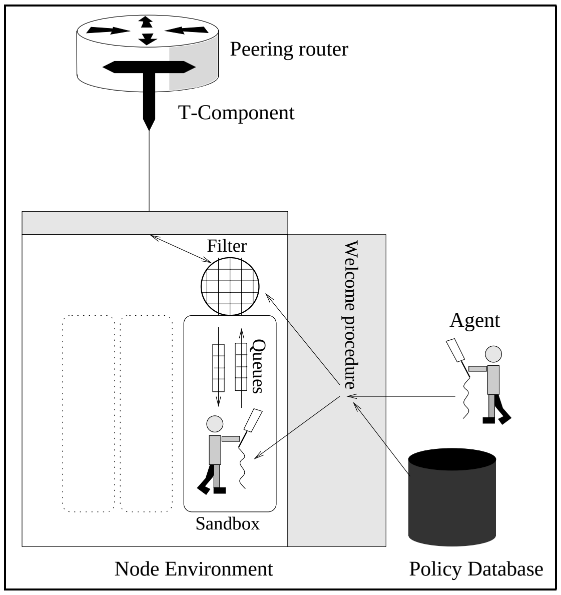

Figure 2: The node environment.

The *node environment* is hosting and executing the SDC agents. It does not have to run on the peering router. In fact, no provider will want to run foreign code on such a crucial machine, so the node will probably run on a separate machine in a 'demilitarized zone', but has to be connected to the T-component. Customers push their agents into the node. The agent does not necessarily have be encrypted, but a strong authentication protocol is needed. This ensures that the node can properly authorise the agent. When the agent arrives at a node, it has to undergo a welcome procedure. After the authentication, the agent asks the node for resources (CPU time, memory and specific traffic). The agent also specifies a packet filter for the bypassing packets it is interested in (see section 3.3). Based on policies, the node authorises the agent for these resources. It provides an execution environment (user-level thread in a 'sand-box'). The agent execution environment also contains an inbound and an outbound packet queue, which is the only way the agent can receive and sent packets.

3.3 Authorisation and Filtering

The agents are effectively separated from the network by filters on the in- and outbound queue. The agents carry a definition of the desired filters. The node environment has policies describing what filters are appropriate for what agent. We propose a mechanism using the mathematical cut between two sets. Both the agent and the node describe the filtering rules as a set of integers (addresses, protocol numbers etc.). We foresee three kinds of wildcards: an integer range, an integer list and 'any' (matching everything). While the agent holds a filter describing what specific traffic it is looking for, the node holds the most general filter it allows for that agent. The node uses the mathematical cut of both filters. The agent can query if the resulting filter is empty (matches no packets at all) or not equal to what is has requested, and react upon this (e.g. terminate gracefully). The node holds generic filters in its policies so that it does not need to keep a filter for each potential customer.

For example the university of Berne (Switzerland) owns the network 130.92.0-.0/16. Suppose that the system administration of the university is interested in traffic that a specific institute with the subnet 130.92.66.0/24 sends across the trans-atlantic link to the US. Thus, the administrator would send an agent to the peering point between the Swiss University Internet Provider (SWITCH) and its US peer MCI Worldcom. The agent would be signed by the system administration of the university and contain a filter specifying the source address with a range wildcard which ranges from -2107883008 to -2107882753 (2-complement of the smallest and the biggest 32 bit address in 130.92.66). Suppose the node would contain a policy saying that the system administrator of the university of Berne is allowed to see any traffic originating form their network, thus its source filter entry is the range wildcard -2107899904 to -2107834369 (130.92.0.0/16). The mathematical cut of the generic node filter and the agent's filter will deliver the desired packets to the agent, since the range specified by the agent is a subset of the generic range assigned by the node. Note that the cut between lists and ranges is always empty or lists or ranges which facilitates the object-oriented implementation of the filters.

Our prototypical and object-oriented filter implementation features filtering for source and destination address, protocol number, the maximum number of packets to receive and the length of payload to be delivered to the agent. It supports the briefly described wildcard mechanism.

3.4 Security Issues.

The security of the proposed infrastructure bases on three concepts. First and foremost, the agents must authenticate themselves with strong cryptography. We do not foresee to implement this mechanism ourselves but rather rely on existing and stable technology such as PGP [Zim], or built-in mechanisms of available agent platforms. Authentication allows the node to relate each agent to a customer, which is responsible for the behaviour of the agent. Second, the agents do not run on the controlled network devices but rather on a dedicated general-purpose computer. Third, the agents run in a sand-box. They have no direct access to neither node nor network resources. Their only communication mechanism uses the in- and outbound queues which are controlled by node filters. The cutting of agent filters with a default filter provided by the node assures in a convenient way that the agents cannot eavesdrop or spoof other peoples traffic.

This architecture is the basis for service delivery control with mobile code. Since the architecture is non-intrusive and only relies on basic agent mechanisms such as authentication and an execution sand-box, we believe that state-of-the art agent technology [LO98, VB99, Fün98, Gra98] can provide most parts of such a platform, and that such a platform can be deployed in the Internet. The only missing key component is thus the T-component (and the agents themselves). The next two sections describe two specific applications of the platform.

4 Controlling a Virtual Private Network Service

Virtual private networks (VPN) for the Internet [FH98a, FH98b] provide a transparent and secure mechanism to interconnect remote sites with IP (see figure 3). IP packets are encapsulated in new IP packets when entering the Internet (tunnelling). The payload of the new packet (the original packet) is encrypted. Virtual private networks over the Internet are a cheap and secure alternative to leased line based private corporate networks. They take advantage of the ubiquitousness of the Internet and the trend towards Intranets[1]. The Internet Engineering Task Force (IETF)

[1]Intranets: Corporate networks based on IP technology.

proposed the VPN standard IPSec [KA98], which is supported by many vendors nowadays. However, VPNs and especially their cryptographic mechanisms are difficult to understand and manage [GBK99]. Therefore, service providers begin to offer VPN services where they setup and manage the tunnel end-points for their customers. However, the security is transparent to the customer. The customer believes, that all the IP traffic that is leaving the network is encrypted. But encryption is computational intensive. How can the customer be sure, that the provider is really performing the IPSec protocol properly and not just e.g. compressing the payload?

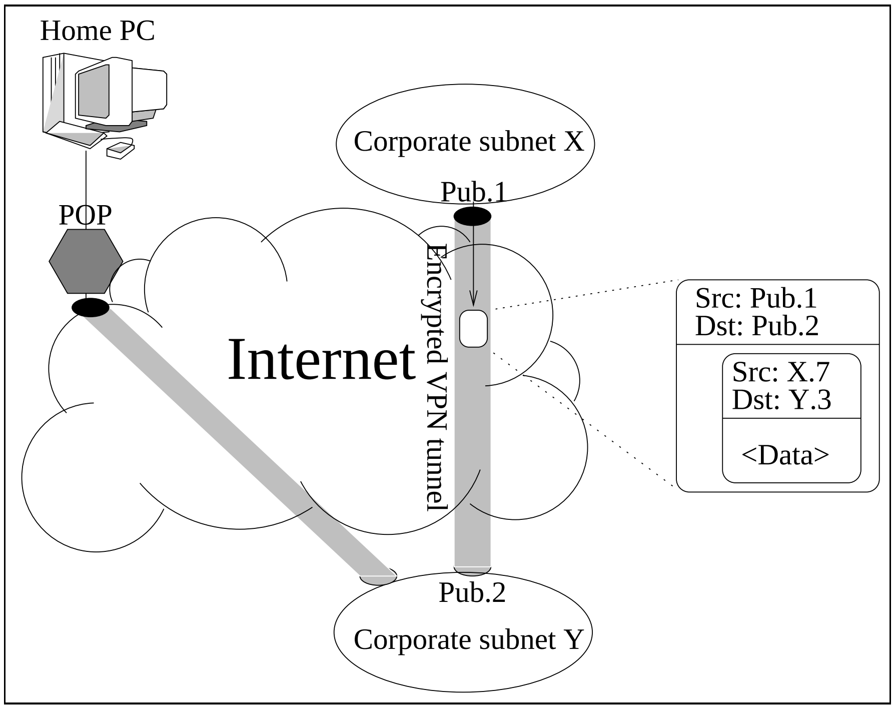

Figure 3: Application scenario for virtual private networks.

A VPN control agent. Given an agent infrastructure as described in section 3 we have many possibilities to check whether the VPN provider is performing as promised.

- The customer can occasionally send out agents to the egress peering points of the access network (see figure 1). These agents check for traffic originating from internal network addresses of the customer's network. The agents should not find any such packets if the service is working properly, since the packets should be encapsulated.

- The customer can send out agents that monitor the IPSec protocol activities. Agents can for example monitor the presence[2] of the key exchange protocol IKE (UDP to port 500). The agents can also analyse the packet structure to see if the proper tunnelling modes are used (examine the IP `protocol` field: 50 for ESP, 51 for AH [KA98]).

- The customer can validate the encryption using statistical tests. This is the most difficult task, thus we explain it in more detail.

Statistical encryption checks. The VPN delivery control agent has also (albeit limited) possibilities to validate the quality of the encryption. Such an agent requests from the node (some) packets that are IPSec encrypted (protocol=50, Encryption Security Payload). The nodes of VPN providers can accept that request, because the payload should be encrypted, so the privacy of the sender is not compromised. Paranoid providers may also limit such access to agents of their customers. The VPN delivery control agent can now apply statistical tests to the payload or to parts of it. The motivation behind this is that a good encryption scheme scrambles the bits so that they look random. Statistical tests can detect regularities in the payload which are a sure sign that no valid encryption scheme has been used.

So far, we implemented two simple statistical tests, namely (1) the byte frequency test which counts the occurrence of each byte value [-128..127] and tests for uniform distribution [Knu81]. (2) The run-test divides the byte stream in sequences of increasing (decreasing) bytes. It counts the number of occurrences of sequences with lengths 1,2,3,4,5 and 6 or more [Knu81]. For example the byte sequence (-33 104 | -45 3 34 | 7 | -19 | -93 1) contains two (increasing) runs of length one, two of length 2 and one of length 3. Both tests are evaluated by comparing the counts (bytes or run-lengths) with an expected distribution. The evaluation uses the well-known χ^2 statistics [Knu81].

The VPN delivery control agent requests encrypted payload and classifies the data into categories (byte values or run-lengths). After all packet data is classified and a significant[3] amount of data has been collected, the agent evaluates the test. Note, that all statistical tests only deliver probability values and not absolute values. The tests indicate the probability that the tested data was produced by a uniform distributed and independent random function. If the data is significantly off the

[2]The IKE protocol of IPSec is encrypted, therefore not much more than the presence can be monitored.

[3]This is dependent on the test, e.g. for the byte-frequency test it is about 3000 bytes.

expected distribution the agent can sent back an alert to the customer application. In case of doubt, the agent can also send back the last packet to be examined by a human expert.

The two proposed statistical tests are by far not the only possible tests. They suffice for our purpose since both tests can detect if compression instead of encryption is used ([BGKL00]). The genericity of the agent infrastructure allows the agent programmer to easily deploy other tests such as the Anderson-Darling test [PAMM98].

Implementation and Evaluation. We have implemented a prototypical agent written in the programming language Java to perform the described tests. However, performance tests indicated some limitations, since statistical tests require significant computation. Running on a Sparc ULTRA 5 with a 269 MHz CPU, the agent could test 1.5 Mbps encrypted data with the run-test and 1 Mbps data with the byte-frequency test. In case the agent wants to monitor a line with higher throughput, it can choose not to analyse every byte in every packet, since sample testing will also suffice to detect misbehaviour.

It is important to note, that traditional customer premises and stationary control programs are under no circumstances able to perform the VPN checks we described in this section. The application would need to have insight in what is going on inside of the Internet, that goes far beyond from SNMP [CFSD90] or web-based network management entries. This necessity of code mobility is not VPN specific as the example of the next section shows.

5 Controlling Differentiated Services

DiffServ is a light-weight and scalable QoS mechanism proposed by the IETF [BBC$^+$98]. A single byte (DiffServ Code Point (DSCP), formerly called TOS) in the IP header is used to code different per-hop behaviours (PHB) that an IP packet can experience. Inside of a network, all IP traffic using the same code point is called a DiffServ behaviour aggregate and is treated the same way. Since there are only a handful of PHBs, the DiffServ architecture scales also to large core networks. To provide DiffServ across multiple administrative domains the DiffServ architecture proposes automated bandwidth brokers [TWOZ99] to negotiate service level agreements (SLA) between different autonomous systems. These agreements describe the volume of DiffServ traffic that can be exchanged between two domains and the price for that traffic. If all the domains between two end users have engineered their networks properly and have established SLAs for the DiffServ volume expected, the DiffServ architecture can guarantee end-to-end QoS. However, today's network engineering mechanisms work with overprovisioning or introduce large signaling overhead [GB99]. Therefore, a provider may deliberately try to over-book its SLAs in order to save money. In case the misconducting provider looses in-profile premium packets due to internal network congestion and insufficient SLA provisioning, it can always argue that it has never received the packets or that another provider down the stream has lost the packets. If the customer has only end-to-end measurements available (s)he cannot detect which of several providers causes the problem.

Given the infrastructure described in section 3, the suspicious customer can send out traffic measurement agents. These agents report back current statistics, and allow the home application to track down a possible problem. Note that for this purpose, mobile agents are not really necessary. Publicly readable measurement information databases (e.g. SNMP) would also do. However, it is much more difficult for the misconducting provider to manipulate local measurement agents than to manipulate local SNMP tables. For the later, some simple `put` commands will cover the traces. For the former the provider needs to analyse the agent code to understand how to trick it into sending false data. The analysis must be done online, since the customer can upload the agent at any time. Even if the provider manages to trick the agent, the customer has tracked down the problem to either the misconducting provider or the neighbour provider, on which the former tries to put the blame on. With mobile agents we can now put both providers under the test using active measurements to single out the bad guy.

Active measurements. Instead of just measuring what is being sent end-to-end (passive measurements), the agents provide the customer with the unique ability to actively generate test traffic from within the network. To test a suspect provider the agents can surround the network of the provider migrating to the closest node that is not under the control of the provider. From there, they can inject small amounts of test traffic. This traffic may use source addresses of the domain of the customer (and the DSCP to be tested) to seamlessly merge with the regular traffic of the customer. The measurement results will now reveal if the provider causes the problem or not. Note, that a provider might manipulate the active measurement being conducted on its neighbour. In general, however there are more than one neighbours to the tested provider, thus such manipulation can be detected once the different measurement results are compared. The measurements have thus to be coordinated. This issue, however, is ongoing research. Nevertheless, it is obvious that the described active measurements combined with the mobility of the SDC agents is a much more powerful tool for performance tests than static methods such as SNMP requests.

Implementation and Evaluation. We implemented agents for local jitter and bandwidth measurements. To perform such measurements the agent needs only to requests the packet headers and the timing information provided by the T-component (see section 3.2). The local jitter can be calculated by evaluating timing information of consecutive packets. For through-put calculation the packet length field provides the necessary information. Since the agent evaluates packet headers only, Java agents perform sufficiently for backbone wire speed.

The implementation work is in an early stage. We foresee to implement agents to account packet loss, delay (coping with the clock skew problem) and packet corruption as well.

6 Related Work

We already referred to work in the area of active networking and mobile agents. None of this work deals with service delivery control. In this section we compare our approach to traditional ways of service delivery control, which is based on measurements.

Network measurement is by its nature a distributed task. Even the old but nevertheless useful `ping` tool needs a source and a destination (to reflect the ICMP message). Today's measurements can be divided in two groups: *end-to-end measurements*, usually carried out by customers and *network traffic measurements* carried out by network providers.

Most end-to-end measurements on the Internet are performed at a small scale to find out local connectivity problems. Large scale measurements which can reveal insights in the Internet topology impose great logistical difficulties [Bol93]. A state of the art approach is to off-line distribute measurement daemons, that are run by the local system administration [Pax97]. The possibilities of mobile code is not exploited here.

Measurements inside of the Internet are carried out by almost any provider, in order to engineer their networks. Tools are often SNMP based, or specialised to the involved network equipment (e.g. Netflow [Cis00]). Often, providers hesitate to make any results of these measurements publicly available, because they fear to offer attacking points to their competitors. In academically influenced networks, network measurement data is sometimes available for via http, e.g. at the web site of the national laboratory for applied network research [NLANR]. However, such data is too aggregated for service delivery control. Architectures for fine-grained traffic data repositories have been proposed [KMKA99]. However, since they do not collect the data on a per-service and per-customer basis. All collected data enters the same database. Therefore, they need to scramble the origin of the data for privacy reasons. This will also render the data useless for some SDC applications.

To our knowledge, mobile service delivery control agents are a novel approach to an emerging problem of new Internet services, namely service delivery control. The most closely related work is probably the Q-bone measurement framework of the Internet2 [THD$^+$99]. However, this work uses provider-oriented stationary measurements which are not intended for the use by the customer. Furthermore, the measurements focus on the supervision of one specific network service, namely Differentiated Services (see section 5).

7 Conclusion

This paper presents the concept, the architecture and the motivation behind using mobile code for the on-line control of the delivery of Internet network services. The approach is classified as a hybrid approach between mobile agent technology and active networking. The paper identifies the advantage of mobile service delivery control agents: 1) the generic and secure interface to measurements, which can be carried out inside remote networks, 2) the ability to actively perform cross checks inside remote networks. The proposed agent infrastructure is kept simple including a threefold security concept, thus keeping the overhead of mobile code low. The paper illustrates the advantages by presenting two examples of new network services proposed by the Internet Engineering Task Force: IPSec based virtual private networks (VPN), and differentiated services (DiffServ). Specialised measurement agents can test both services to a degree that is not feasible with stationary measurement techniques.

Acknowledgements. This work has been performed in the framework of the project Advanced Network and Agent Infrastructure for the Support of Federations of Workflow Trading Systems - ANAISOFT (SPP ICS ElCom Project 5003-057753) which is funded by the Swiss National Science Foundation (SNF). The SNF also funded the project Charging and Accounting Technology for the Internet - CATI (CAPIV 5003-054559/1 and MEDeB 5003-054560/1) which supported this work in its early stages. The authors would like to acknowledge all contributions from their current and former project and working colleagues. The implementation environment for the SDC agents is funded by the SNF R'Equip project no. 2160-053299.98/1 and the foundation 'Förderung der wissenschaftlichen Forschung an der Universität Bern'.

References

[BBC$^+$98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services, 1998. RFC 2475.

[BGKL00] T. Braun, M. Günter, I. Khalil, and L. Liu. Performance evaluation of virtual private networks. Technical report, Institute for Applied Mathematics and Computer Science, March 2000.

[BGP97] Mario Baldi, Silvano Gai, and Gian Pietro Picco. Exploiting code mobility in decentralized and flexible network management. In *Proceedings of the 1st International Workshop on Mobile Agents*, Berlin, Germany, April 1997.

[Bol93] J.-C. Bolot. End-to-end packet delay and loss behavior in the Internet. In *Proc. SIGCOMM '93*, pages 289–298, September 1993.

[CBZS98] K. Calvert, S. Bhattacharjee, E. Zegura, and J. Sterbenz. Directions in active networks. *IEEE Communications*, 36(10), October 1998.

[CFSD90] J. Case, M. Fedor, M. Schoffstall, and J. Davin. A simple network management protocol (SNMP), May 1990. RFC 1157.

[CHK97] David Chess, Colin Harrison, and Aaron Kershenbaum. Mobile agents: Are they a good idea? In Jan Vitek and Christian Tschudin, editors, *Mobile Object Systems: Towards the Programmable Internet*, number 1222 in LNCS, pages 25–45. April 1997.

[Cis00] Cisco. Network management. http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/, 2000.

[FH98a] P. Ferguson and G. Huston. What is a VPN - part I. *The Internet Protocol Journal*, 1(1), 1998.

[FH98b] P. Ferguson and G. Huston. What is a VPN - part II. *The Internet Protocol Journal*, 1(2), 1998.

[Fün98] S. Fünfrocken. Transparent migration of Java-based mobile agents: Capturing and reestabilshing the state of Java programs. In K. Rothermel and H. Fritz, editors, *Proc. Mobile Agents MA '98*, September 1998.

[GB99] M. Günter and T. Braun. Evaluation of bandwidth broker signaling. In *Proceedings of the International Conference on Network Protocols ICNP'99*, pages 145–152. IEEE Computer Society, November 1999.

[GBK99] M. Günter, T. Braun, and I. Khalil. An architecture for managing QoS-enabled VPNs over the Internet. In *Proceedings of the 24th Conference on Local Computer Networks LCN'99*, pages p.122–131. IEEE Computer Society, October 1999.

[Gra98] R. S. Gray. Agent Tcl: A flexible and secure mobile-agent system. Technical report, Darhmouth College, 1998. PCS-TR98-327.

[HGF+99] J. Hulaas, L. Gannoune, J. Francioli, S. Chachkov, F. Schütz, and J. Harms. Electronic commerce of Internet domain names using mobile agents. In *Proceedings of the Second International Conference on Telecommunications and Electronic Commerce (ICTEC'99)*, October 1999.

[JLM89] V. Jacobson, C. Leres, and S. McCanne. Tcpdump. available via ftp to: ftp.ee.lbl.gov, June 1989.

[JMKM99] W. Jansen, P. Mell, T. Karygiannis, and D. Marks. Applying mobile agents to intrusion detection and response. Technical report, National Institute of Standards and Technology, October 1999.

[KA98] St. Kent and R. Atkinson. Security architecture for the internet protocol, November 1998. RFC 2401.

[KMKA99] Akira Kato, Jun Murai, Satoshi Katsuno, and Tohru Asami. An Internet traffic data repository: The architecture and the design policy. In *Proc. INET '99*, June 1999. http://www.isoc.org/inet99/proceedings/4h/4h_1.htm.

[Knu81] D. E. Knuth. *The art of computer programming*, volume 2 Seminumerical Algorithms. Addison-Wesley, 2 edition, 1981.

[LO98] Danny Lange and Mitsuru Oshima. *Programming and Deploying Java Mobile Agents with Aglets*. Addison-Wesley, 1998.

[MJ99] Richard Murch and Tony Johnson. *Intelligent Software Agents*. Prentice Hall, 1999.

[NLANR] National Laboratory For Applied Network Research. Network analysis infrastructure. http://moat.nlanr.net/.

[PAMM98] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis. Framework for IP performance metrics, May 1998. RFC 2330.

[Pax97] Vern Paxson. End-to-end Internet packet dynamics. In *Proc. SIGCOMM '97*, 1997.

[THD$^+$99] B. Teitelbaum, S. Hares, L. Dunn, R. Neilson, V. Narayan, and F. Reichmeyer. Internet2 QBone: Building a testbed for differentiated services. *IEEE Network*, 13(5):8–16, September/October 1999.

[TSS$^+$97] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden. A survey of active network research. *IEEE Communications Magazine*, 35(1):80–86, January 1997.

[TWOZ99] Andreas Terzis, Lan Wang, Jun Ogawa, and Lixia Zhang. A two-tier resource management model for the Internet. In *IEEE Global Internet'99*, December 1999.

[VB99] Jan Vitek and Ciaran Bryce. The JavaSeal mobile agent kernel. In *Proc Symposium on Agent systems (ASA '99) and Applications and Symposium on Mobile Agents (MA '99)*, October 1999.

[WGT98] D. Wetherall, J. Guttag, and D. L. Tennenhouse. ANTS: A toolkit for building and dynamically deploying network protocols. In *IEEE OPENARCH '98*, April 1998. San Francisco.

[Whi94] James White. Telescript technology: The foundation for the electronic marketplace. General Magic White Paper, 1994.

[WJ99] Michael J. Wooldridge and Nicholas R. Jennings. Software engineering with agents: Pitfalls and pratfalls. *IEEE Internet Computing*, pages 20–27, May/June 1999.

[Zim] Phil Zimmermann. Pretty good privacy. http://www.pgpi.org/.