

# Path-Coupled Configuration of Passive Measurements

Frederic Raspall  
NEC Europe Ltd.

Network Laboratories Heidelberg  
E-mail: raspall@netlab.nec.de

Marcus Brunner  
NEC Europe Ltd.

Network Laboratories Heidelberg  
E-mail: brunner@netlab.nec.de

Juergen Quittek  
NEC Europe Ltd.

Network Laboratories Heidelberg  
E-mail: quittek@netlab.nec.de

Miquel Martin  
NEC Europe Ltd.

Network Laboratories Heidelberg  
E-mail: miquel@netlab.nec.de

## Abstract

*Passive and active technologies are available for measuring hop-by-hop properties of traffic along its path through the Internet. Passive technologies can measure these properties accurately, but configuring them for the measurement of a particular traffic flow at all hops requires significant overhead for measurement configuration. This problem does not apply to active measurements, such as traceroute, because probing packets automatically follow the same path as the traffic flow to be measured. However, active techniques measure properties/conditions of the injected traffic, which may differ from those of the traffic of interest. This paper describes an approach that combines the advantages of both worlds. We use path-coupled signaling for configuring passive hop-by-hop measurements along the path of a traffic flow of interest. We identify the advantages and disadvantages of the approach and describe our prototype implementation.*<sup>1</sup>

## 1 Introduction

There are several applications including traffic engineering and SLA monitoring for which it would be desirable to know the properties of a specific traffic flow along its path through the network. The knowledge of hop-by-hop loss ratio, delay, jitter and other properties experienced by the traffic of interest could be used, for example, to identify which hops contribute most to loss, delay and jitter. While existing active methods can provide path-related measurements (for example, traceroute measures hop-by-hop delay), they measure properties of injected traffic and not nec-

<sup>1</sup>This work was partially funded by the EU within the EU IST INTERMON project.

essarily of the traffic of interest. Passive measurements can provide more accurate information about the traffic of interest. Their disadvantages are that they first need to acquire topology information in order to know which node to configure, and they are expensive in terms of configuration. Each node on the path needs to be individually configured to measure the particular traffic of interest and to report measurement data to a desired data collector.

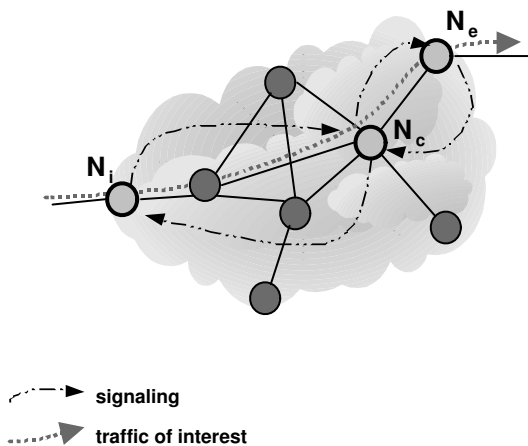
This paper describes an approach that overcomes these disadvantages of passive measurements. It reduces the configuration overhead and automatically determines the nodes on the data path to be configured. The idea is to use a path-coupled signaling protocol for measurement configuration. Path-coupled signaling messages are sent along the same path as the traffic flow to be measured. They carry configuration information to the measurement probes, at those nodes on the path which support this particular signaling protocol.

The IETF Next Steps In Signaling (NSIS) WG is currently developing a so-called NSIS Transport Layer Protocol (NTLP) that is a hop-by-hop transport protocol Carrying signaling information along a data path. The signaling information itself is carried by NSIS Signaling Layer Protocols (NSLPs) that serve specific purposes. Currently, the NSIS working group develops two NSLPs, one for QoS resource reservation and one for middlebox control. Our approach aims at an NSLP for traffic measurement. Similar work has been proposed in [1].

The general idea is described in Section 2. Section 3.1 describes an existing measurement architecture that we developed and how we extended with the use of NSIS. The implementation itself is described briefly in section 4. It uses the upcoming IETF standards for the NTLP and the Packet SAMPLing (PSAMP) protocol.

## 2 An Architecture for Path-coupled Measurements

Our architecture is built on top of the IETF NSIS approach. The NTLP carries signaling information from a sending node  $N_i$  to a destination node  $N_e$ . The path to  $N_e$  is determined by IP routing and is assumed to be the same for all packets from  $N_i$  to  $N_e$  (which unfortunately does not hold in some rare scenarios). At each node  $N$  on the path, which supports NTLP, the NSLP payload of the signaling message is processed before forwarding the (potentially modified) message. The NTLP further ensures that after the signaling message reaches its destination, a signaling reply message sent back to the original source visits again all nodes that processed the original message but in reverse order. Figure 1 depicts a scenario with  $N_i$ ,  $N_e$  and an intermediate node  $N_c$  supporting our NTLP and NSLP. Note that not all nodes may support the protocols nor the measurements requested.



**Figure 1. General scenario for path-coupled configuration of measurements**

We use NTLP messages for configuring traffic measurement at the visited nodes. Therefore, we have defined an NSLP that can carry measurement configuration requests, for example a five-tuple describing a micro flow or other parameters specifying the attributes of the traffic to be measured, and the address of a data collector to which measured data should be reported. When a signaling message is processed at a node, the following actions are performed:

1. A node not supporting NTLP and NSLP simply forwards the message.
2. Otherwise, it either accepts or rejects the request depending on both its current status (CPU load, available

memory, existing number of requests, etc.) and on access control policies.

3. If accepted, the request is stored but not yet activated.
4. If accepted, a counter in the signaling message is incremented indicating the number of acceptances on the path.
5. Then, the message is forwarded.
6. The request is stored for a pre-defined time
7. If a signaling reply message arrives before the request timed out, then the requested measurement will be activated at the node. This typically involves the setup of a new filter according to the description of the traffic to be measured. Note that a reply message is expected from the recipient of the original signaling message. This may contain an identifier of the request and the number of accepted requests.

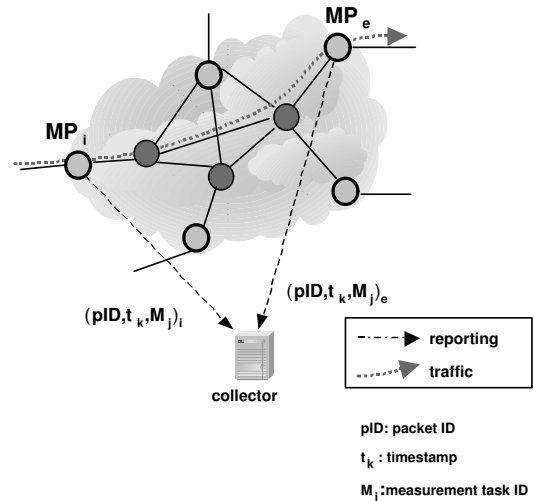
This way, different traffic flow measurements can be configured. For hop-by-hop loss measurement, each node can be requested to count packets for a specified flow and to report counters to a data collector. There, losses per hop can be computed. For hop-by-hop delay and jitter measurement, each node can be configured to report packet timestamps for all packets of the flow. They can be used at data collectors for calculating per-hop delay, total delay, per-hop contributions to jitter, and total jitter. Also accurate loss measurements can be made this way.

Configuring passive traffic measurements in this way has advantages and disadvantages. While advantages include:

- The nodes on the data path to be configured are automatically discovered.
- The nodes on the path that do not support NTLP and NSLP do not have any negative impact on measurements conducted by other nodes.
- Only nodes on the data path are configured, no nodes outside the path conduct unnecessary measurements.
- Nodes on the path only measure the traffic for which there was an explicit request. This reduces resource consumption at the nodes.
- The collectors to which measurement data is sent can be arbitrarily located. They can be co-located with nodes on the data-path or not.

Some disadvantages are:

- The scalability of the approach is limited. If all flows in a network are to be measured, then configuring passive measurement by conventional means is preferable.
- While different levels of granularity may be possible, the specification of the traffic of interest must be carefully done to ensure that only the traffic following a specific path is measured.
- The measurement can only be requested from a node on the path. A central network management station cannot generate appropriate path-coupled signaling messages.



### 3 Path-coupled delay and loss measurements

The use of a signaling hop-by-hop protocol with tight control on the specific nodes being visited, such as NSIS, offers new possibilities in the way measurements can be taken in the network. In this section we explore an example case. First we briefly summarize an architecture for the measurement of One-Way-Delay (OWD) that we implemented and presented in [3], inspired in the work in [2]. Then we show how, by extending the elements in this architecture with NSIS functionality, new ways of measuring delay, jitter and loss can be supported and how new services could then be provided.

#### 3.1 One-Way-Delay (OWD) measurement architecture

The OWD system described in [3] aims at building transit delay matrices between any pair of ingress/egress nodes within an autonomous network using passive measurements, that is, by observing the traffic already flowing through the network. This OWD architecture is built using two different entities: *Measurement Points* (MPs) and a *collector*. MPs are located at the edges of a network where they timestamp a subset of the packets passing by, that is entering or leaving the network. The timestamps generated by the MPs are sent to a collector, which correlates this data to produce OWD estimates. The idea is to obtain a delay sample by subtracting the timestamps generated when observing the same packet at two different MPs. Figure 2 depicts the components in this architecture. For the OWD estimation to be correct, several restrictions apply in this process:

- The timestamps for the same packet generated at the two MPs need to be offsets from the same reference time, thus the MPs need to be clock-synchronized to some extent. Enough degree of synchronization can be achieved with the signal provided by a GPS receiver at a reasonable cost.

**Figure 2. OWD measurement architecture and its components for the estimation of delay matrices in a closed cloud**

- The timestamps used for the computation of OWD samples need to refer to the same packets, thus the packets need to be univocally identified by pIDs. These can be generated by applying hash functions to the invariant packet header fields and payload<sup>2</sup>. In a series of unreported experiments we show the existence of suitable hash functions<sup>3</sup> for pID generation.
- There is a certain probability of collision<sup>4</sup> inherent to the process of pID generation using hash functions. These collisions should be detected and corrected to avoid the computation of wrong delay samples, and we show also in [3] a possible way to achieve that.
- For scalability reasons, not every packet traversing a MP is to be measured. The system allows the selection of a fraction of the traffic by *subsampling*. However, not every sampling scheme is feasible, since the same packets must be selected at the two MPs, or, in other words, if a packet is selected at one MP, it should be selected at the other MP. This functionality can be achieved also by using a hash function and considering for measurement the packets whose hash value fall within some "selection range"<sup>5</sup>.

<sup>2</sup>by invariant we refer to those header fields that do not change from hop to hop. E.g. the TTL is not invariant. In order to ensure that a packet is identified by the same pID, the hash function must be applied to fields that do not change.

<sup>3</sup>computationally efficient and with low collision probability

<sup>4</sup>i.e. having two different packets with the same pID

<sup>5</sup>note that the hash function used and the selection range chosen must be the same at the two MPs.

- For some traffic to be measured, it must be observed at two MPs. If some traffic is only observed at one MP, no delay estimation for that traffic is possible. Also, the reporting of timestamps and pIDs for packets being observed at only one MP will waste bandwidth and processing resources at the MP and at the collector. This is actually the reason why the edges of the network are good places to put the MPs, since they confine a closed portion of a network.

A limitation of the original architecture is that there is a restriction on the location of MPs. They need to be deployed carefully so that all traffic crossing one of them will cross another one. Traffic being observed at only one MP leads to a waste of bandwidth and post-processing at the collector since no delay sample for that traffic will be possible anyway.

Also, the system described in [2] allows for the measurement of one-way transit times, but with limited control on the nature of the traffic being measured. That is, the selection of the traffic is "blind" since it relies on a *selection range* in the hash co-domain, and there is little control on the sort of traffic that will fall within that range. While this level control is enough for certain applications, it may not be for others. In [3] a filtering block was added to the scheme in [2] that, based on some traffic specification (measurement task) allowed the early selection of a specific traffic type. The problem is then to know in which MP to place a specific measurement task since it is not always easy to know the path a specific traffic will follow. That is, while some measurement tasks can be applied to all MPs in the cloud to estimate the transit delay matrices corresponding, for instance, to specific per-protocol aggregates or per-class-of-service aggregates, it is inefficient to do so when the targeted traffic (measurement task) aims at the selection of a flow following a specific path, or, in other words, crossing only one egress MP. Specifically if the selection of traffic is to be done based on the source or destination addresses<sup>6</sup> the knowledge of the MPs that the specific flow will traverse is required in order to avoid to configure all MPs. Configuring all of the MPs constitutes unnecessary overhead, wastes memory at the MPs and forces them to inspect and match the traffic against an unnecessary rule. Figure 3 describes these two limitations.

The use of a path-coupled signaling protocol may be a way to overcome these limitations: when the specific type of traffic is to be measured, a signaling message is sent along the same path that the traffic will follow to configure only those MPs on that path with the proper measurement task.

<sup>6</sup>As it could be in the process of validating the SLA for a specific customer

### 3.2 Path-coupled delay and loss measurements

As already pointed out in the previous section, the usage of a path-coupled signaling protocol to selectively configure the measurement entities in a single path can be useful. In the case of the OWD architecture, the use of NSIS eliminates the restrictions on the location of the MPs to "enclose" a portion of the network, allowing them to be spread around. It also avoids the configuration of all MPs and thus the better usage of the network resources. In this section we describe how, besides extending the usage possibilities of some types of measurements, it allows for other types in a more end-to-end fashion.

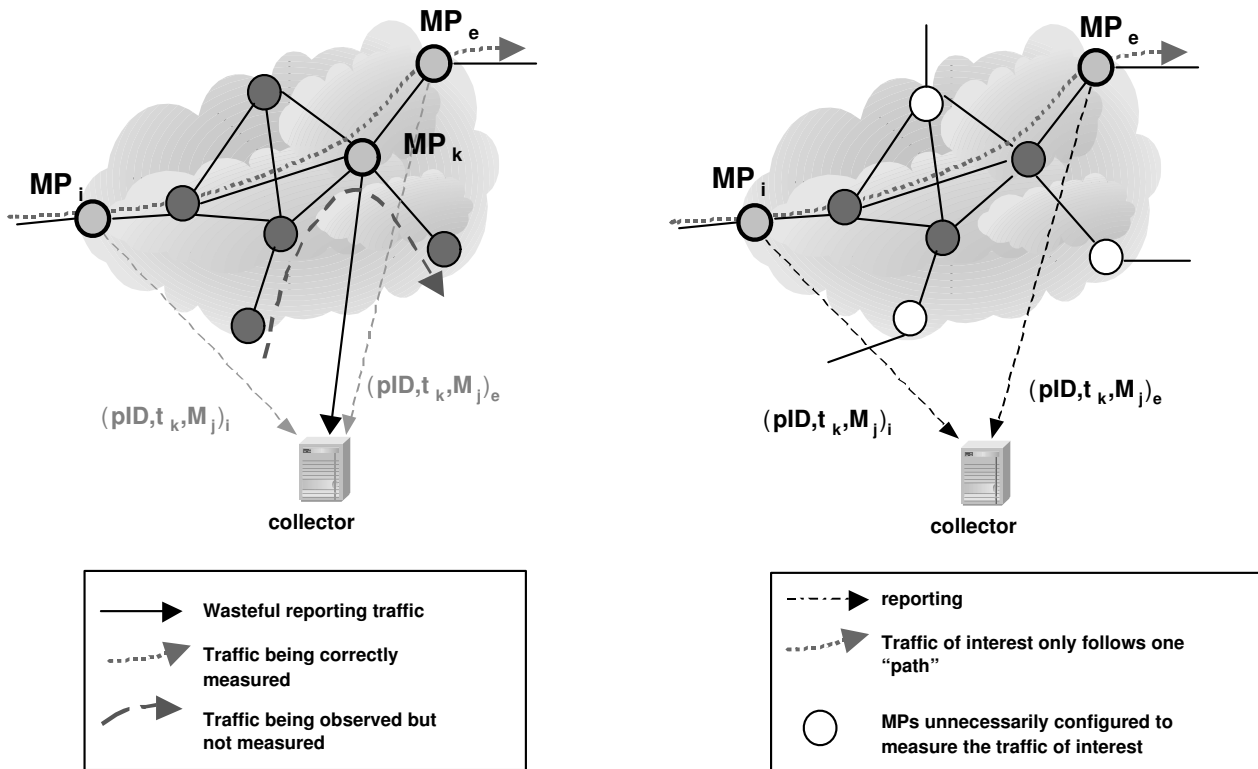
With NSIS, the MPs observe, timestamp and report only on those packets matching a specific filter, which is setup only at those MPs on the path that the traffic of interest will follow. The only restriction is that the filtering has to accept only the traffic flowing through the path where the filters are setup, which is the path followed by the signaling messages.

Note that the use of filters does not eliminate the need of identifying the packets with pIDs, since packets (and thus, timestamps) can be lost or reordered. In addition, the use of filters restricts the nature of the traffic being observed and this can be exploited to reduce the collision probability in the process of generating pIDs with a hash function. Also, the reporting of the filter identifier makes it easier to detect possible collisions. If the traffic of interest visits the MPs that have been configured<sup>7</sup>, we can assume that if a pair  $(pID, t_k)$  is reported by some but not all of the MPs, a packet loss has occurred. When using filters, if they are restrictive enough there is no need to subsample since the amount of traffic being measured is small. This has the implication that the overall number of losses for the traffic of interest can be computed as compared to the case when no filtering but subsampling is used, in which the detection of losses is possible only for those packets falling within the "selection range".

Based on this, the combination of NSIS and OWD allows for the measurement of not only the delay experienced by the targeted traffic along a segment of a path, but also for total losses, which in turn allow for the estimation of the jitter at the collector. In order to report on the number of bytes lost, we extended the report messages to include the length of every packet being reported. This additional information also allows to better detect the collisions, at the collector: two reports with the same pID but different length correspond to two different packets colliding in the pID domain.

Consider the situation depicted in figure 4, where three MPs ( $MP_i$ ,  $MP_c$  and  $MP_e$ ) are configured to filter and measure some traffic of interest ( $M_j$ ). Obviously the sys-

<sup>7</sup>which can be enforced by NSIS



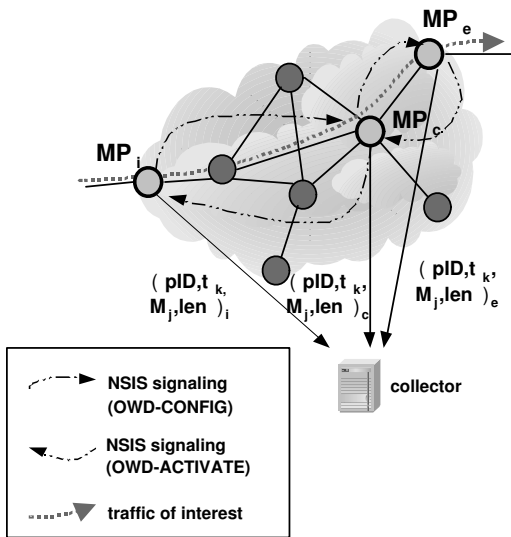
**Figure 3. Two of the limitations of the OWD architecture. Left: usage in an open environment may lead to the reporting of traffic observed at only one MP. Right: for the measurement of some types of traffic, all MPs need to be unnecessarily configured unless routing information is available.**

tem will not report on any loss occurring before  $MP_i$ . If a tuple for a specific pID is reported from  $MP_i$ ,  $MP_c$  and  $MP_e$ , then it is possible to generate delay samples from  $MP_i$  to  $MP_c$ , from  $MP_c$  to  $MP_e$  and thus from  $MP_i$  to  $MP_e$ . If a tuple for a specific pID is observed at  $MP_i$  and at  $MP_c$  but not at  $MP_e$  then the packet was probably lost between  $MP_c$  and  $MP_e$ . If a pID is only observed in  $MP_i$ , then the packet was lost in between  $MP_i$  and  $MP_c$ . A pID being observed at  $MP_i$  and at  $MP_e$  but not at  $MP_c$  could indicate that a problem had occurred at  $MP_c$  (e.g. the MP is overloaded), or that the packet followed a different path due to a change in the routing setup or due some load balancing mechanism. Since packet loss can be detected to some extent, the collector can consider them when calculating jitter. We are currently working in different ways of computing these estimates considering this new environment and that other situations than those described may occur. For example, the reception of a big number of reports from only one MP may indicate a change in the routing at some node. We are also finalizing the definition of the architecture by defining different modes of operation: one

without filters, which may be used when performing measurements for traffic engineering purposes and other with filters and NSIS, to be used for instance for the monitoring of specific traffic types, for SLA validation or as a value-added service in a VPN scenario.

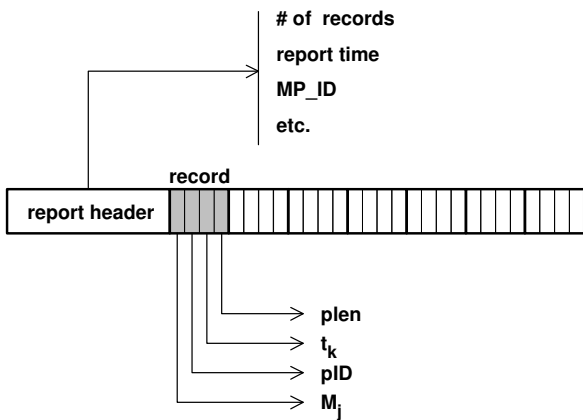
### 4 Implementation

We have extended our implementation of the MPs in the OWD architecture to provide path-coupled measurements. The MPs were implemented as Linux/PC boxes. The traffic being measured was captured by means of the well-known *libpcap* packet capturing library. The library provides copies of sniffed packets from a certain network interface. It also provides a timestamp, which is the one put by the kernel when the device driver queues the packet in the input queue (known as backlog queue). In order to synchronize the MPs we used GPS receivers to synchronize the PC's internal clocks, from which the kernel generates the timestamps. The hash function used to generate the pID's is the *mmh* function, which provides low collision probability at a small computational cost. The exporting



**Figure 4.** By enhancing the MPs with NSIS functionality, the delay in a single segment of the network can be estimated for a specific traffic. Also, loss and jitter can be estimated at the collector.

to a collector was done in a way inline with the ideas being proposed by the IETF PSAMP WG. As a transport protocol we used TCP to provide reliability and 6 depicts the structure of the report messages sent by the MPs to the collector.



**Figure 6.** Format of the report messages sent by the MPs

In order to support the hop-by-hop signaling the MPs need to capture the NSIS packets that traverse them, interpret them, and act upon them accordingly. This may involve manipulating the packet, dropping it or injecting it

back to the network so that it follows the path until the next node. To have this functionality we used the *LibIPQ* library that works with the Linux Netfilter architecture. By setting up appropriate match rules in the IPTables firewall, packets are sent to a kernel queue, which *LibIPQ* can access, effectively retrieving the packets from kernel space and into user space, where they can be read and manipulated. Afterwards, the packet can be released (modified or not) and dropped. We implemented an NSIS engine by using *LibIPQ* and raw sockets for the creation of NSIS messages. To support the configuration of measurement tasks at the MPs we defined two NTLP messages specific for OWD measurements. One of them is for the specification of the traffic to be measured, i.e. the measurement task (OWD-CONFIG). The other one (OWD-ACTIVATE) is for the activation of such a measurement task. When the NSIS engine receives a signaling message of type OWD-CONFIG it creates an entry in the set of measurement tasks. This entry may be further activated upon the reception of an OWD-ACTIVATE or cleared after some predefined timeout. We did a preliminary implementation of the NSIS engine and the NTLP, since their standardization is not completed yet. Figure 5 depicts the software components of our implementation.

## 5 Conclusions

Active measurements typically involve the injection of probing traffic to infer some metric related to the status of a path within a network, like end-to-end delay or available bandwidth. The result of the measurements with active techniques can be affected by this injected traffic or relate more to it than to the existing one. Passive measurements on the other hand, can discover properties like the composition of the existing traffic crossing a specific point in the network, providing richer information but with a limited scope and they require the configuration of equipment. There are situations in which passive techniques are preferable but they are difficult to configure since sometimes it is unknown the path that the targeted traffic will follow.

In this paper we discuss the usage of a path-coupled signaling protocol to automatically configure for passive measurements only those nodes that the traffic to be measured will visit. We describe an example measurement architecture and how it can be enhanced by the use of a signaling protocol to extend its flexibility and functionality. We have implemented preliminary software prototypes of the components of the complete measurement system and are in the process of understanding how the data in this new setup can be better exploited to infer more on the conditions of the network and the ones experienced by the traffic being measured.

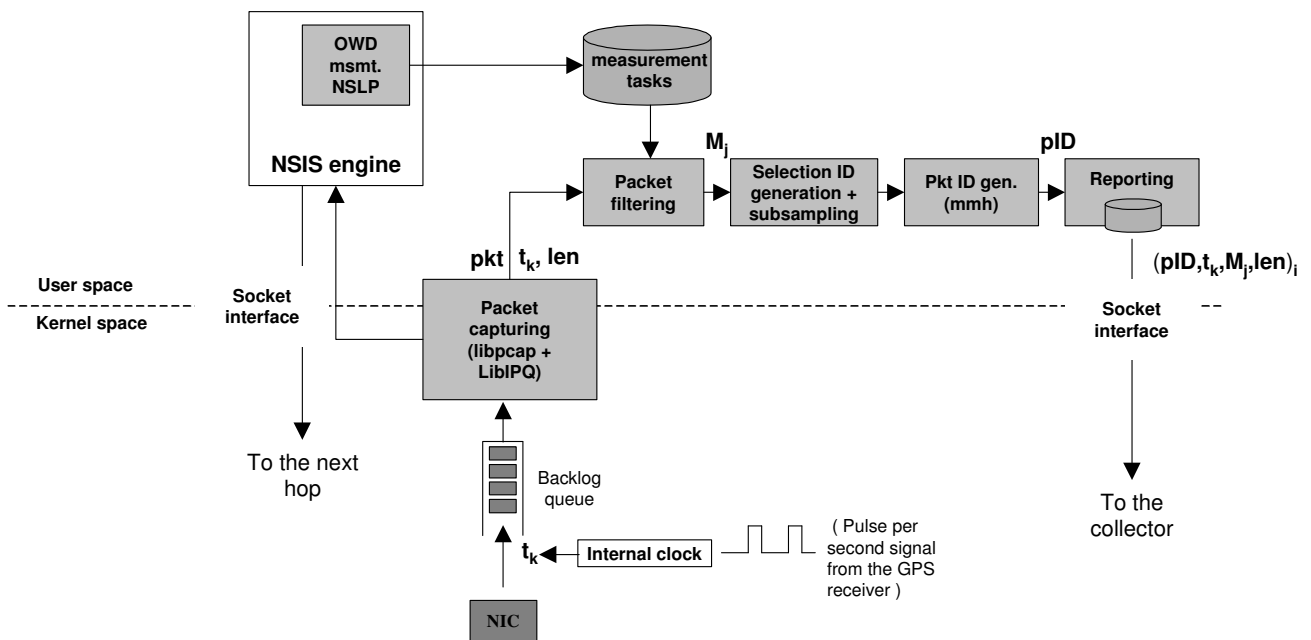


Figure 5. Software implementation of the NSIS-enhanced OWD Measurement Point in a Linux/PC.

References

- [1] Alban Couturier, "Signaling for qos measurement," Internet draft, May 2003.
- [2] Tanja Zseby, Sebastian Zander, and Georg Carle, "Evaluation of building blocks for passive one-way-delay measurements," Passive and Active Measurements Workshop, PAM, 2001.
- [3] S. Niccolini, M. Molina, F. Raspall, and S. Tartarelli, "Design and implementation of a one way delay passive measurement system," to appear in NOMS, 2004.
- [4] S. Niccolini, S. Tartarelli, F. Raspall, and M. Molina, "Analysis of hash functions and synchronization techniques for one way delay estimation," submitted to ICC'04, 2004.