Implementation of Differentiated Services over ATM

Informatikprojekt

Arik Dasen

Universität Bern, 1999−2000

Inhaltsverzeichnis

1 Abstract

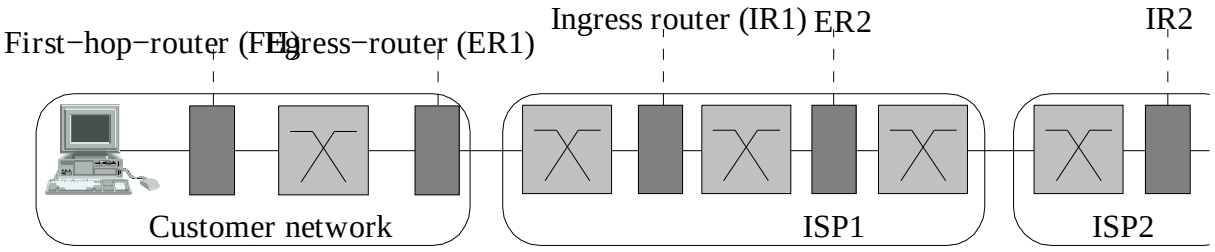
This report has been created within the context of the project "Differentiated Services over ATM" which is currently being performed between NEC Europe Ltd. and the University of Berne. "Implementation of Differentiated Services over ATM" has been my "Informatikprojekt" (studentproject). Until now, a Differentiated Services implementation on Linux has been realized [4]. This implementation is independent of any link layer charcteristics and does not take these into account. This report describes a concept how the DiffServ implementation can take advantage of ATM functionality if ATM is the used link layer. The paper also describes two possible implementation of this concept.

2 Introduction

The implementation of Differentiated Services (DiffServ) such as Premium and Assured Services requires to implement additional traffic conditioning components within the IP routers, in particular within the boundary routers of a DiffServ domain. Such traffic conditioning functions are among others classification, marking, metering, shaping, and policing. Traffic conditioning puts additional burden to the routers and may limit their performance. In particular, shaping is one of the most expensive operations. Instead of implementing these traffic conditioning functions in software, hardware implementations can be used in order to overcome potential performance bottlenecks. The approach described within this paper uses available ATM hardware for these purposes. In particular, shaping and policing functions are already implemented in ATM switches and ATM network interface cards (NICs). We discuss whether ATM can be used to implement the most popoular DiffServ services Premium Service (Expedited Forwarding, EF) and Assured Services (Assured Forwarding, AF). We also describe the current implementations, the steps planned for the future and alternatives to them.

3 DiffServ over ATM Scenario

Figure 1 depicts a typical DiffServ scenario, in which all hosts and routers are interconnected via one or more ATM switches. The sending host is connected via any link technology (e.g., Ethernet) to the first−hop router. Boundary routers (ingress and egress routers) are interconnected via at least two ATM switches. This makes sense if one ATM switch has to implement certain traffic conditioning functions for exactly one DiffServ domain.



Drawing 1 DiffServ over ATM Scenario

4 EF Service over ATM

4.1 Normal EF Operation

In the case of EF service, the FH in the customer network has to classify EF flows, mark them with an appropriate DSCP and shape the flow to the pre−negotiated EF peak rate. The ER has to shape the whole EF traffic to the rate negotiated with ISP1. If the incoming EF flows have already been classified as such before, ER might apply its shaping function to all EF packets without doing a re−classification. IR1 at ISP1 then performs policing based on the profile pre−negotiated with customer 1, while ER2 again performs shaping based on the profile negotiated among ISP1 and ISP2. IR2 again performs policing functions. Note that policing can be implemented as shaping without buffering.

4.2 EF Service over ATM (first approach)

4.2.1 Architecture

This section describes how available shaping and policing functions in ATM switches can be used to set up a DiffServ implementation which avoids the deployment of shaping and policing (software) functions within IP routers.

First, the FH classifies all EF packets according to the defined profile information configured within FH. In the next step, the various flows must be shaped in order to get conforming flows. For shaping at the FH, we set up a number of PVCs between FH and ER1. For each PVC we create an Classical IP over ATM interface at both of the routers. Each of these logical ATM interface address pairs share a common IP subnetwork. The IP over ATM interfaces can be configured with a shaping rate. This configuration must be done via the command line interface.

For each flow to be shaped, we use one of these PVCs as an CBR PVC and configure the ATM switch between FH and ER1 so that the ATM switch performs CBR shaping on this PVC. The configuration is done by conventional network management functions. For the ATM switch we are using our own web−based management tool [5] which allows to setup, modify and delete ATM PVCs.

Alternatively, this shaping can be supported by the ATM driver/NIC at the FH. Within the FH, the forwarding behavior must be changed so that the FH forwards all packets of a certain EFflow over the corresponding PVC. This requires that the forwarding process considers in addition to destination addresses flow information such as IP source addresses, DiffServ Codepoints (DSCPs), port numbers or even higher protocol information for forwarding. Note that normal forwarding considers IP destination addresses only.

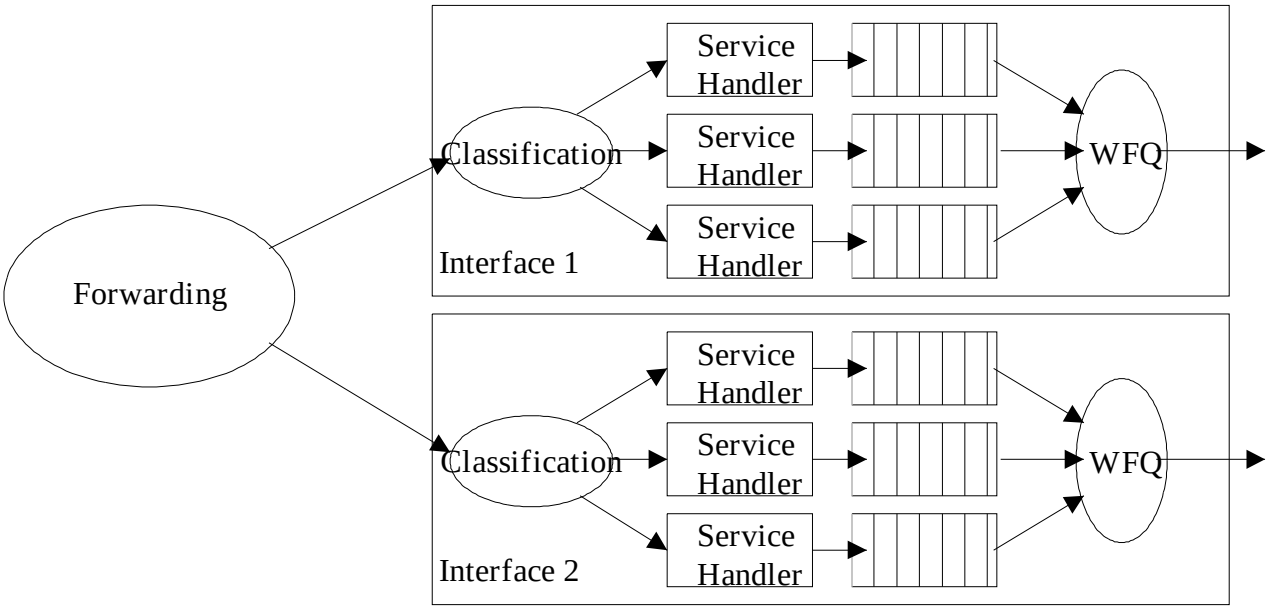
Another shaping step must be performed at the edge between the customer network and ISP1. For that purpose, ER1 takes all packets from the EF PVCs and forwards them over a single common PVC towards ISP1. The ATM NIC at ER1 and/or the ATM switch at the border of the customer network performs shaping according to the rate negotiated between the customer and ISP1, while the ATM switch at the border of ISP1 performs policing functions. The same behavior applies at the boundary between ISP1 and ISP2 in routers ER2 and IR2. In all cases, the ER has to classify the EF service packets and to forward them over the PVC that has been established for EF traffic to the IR. This means that the ER forwarding behavior must be changed again in order to consider the DSCP for forwarding.

In our implementation approach, an incoming packet is first classified and forwarded to a certain output interface / device. DiffServ queuing components for this device are selected in order to mark the packet according to the service the packet should get. After forwarding the packets in the first part of the router implementation, they are marked appropriately and transmitted over a dedicated

ATM PVC in the second router implementation part.

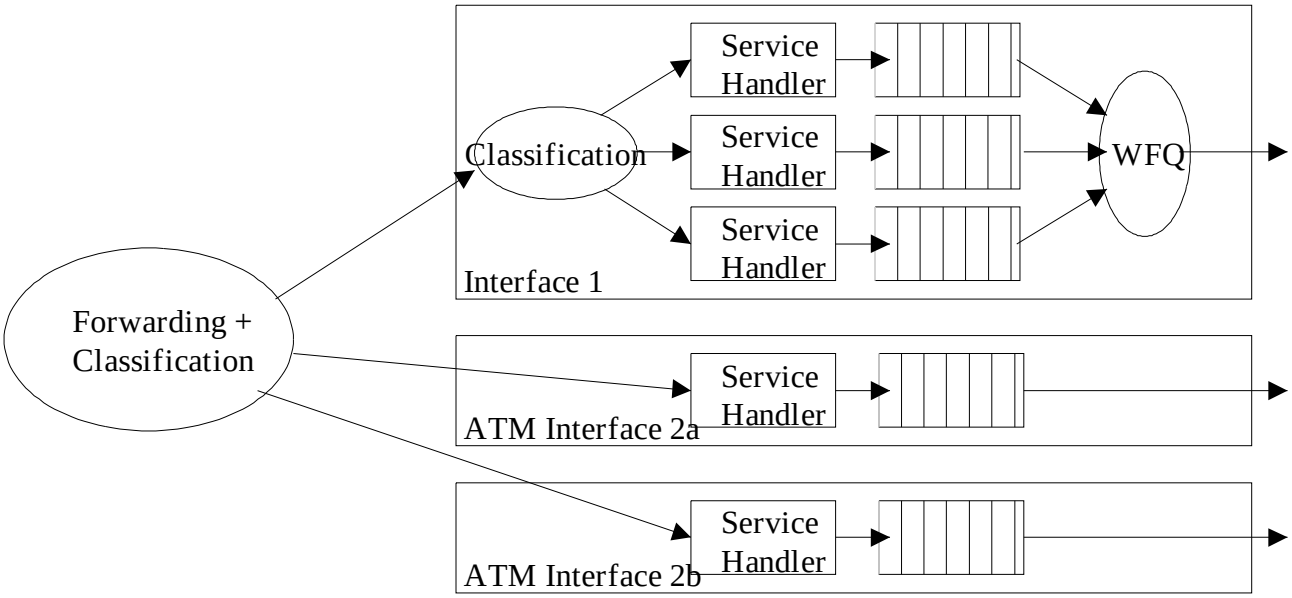
In addition to router configuration, the ATM switches between two routers have to be configured appropriately, ie. the PVCs used for DiffServ traffic have to be setup and the associated shaping and policing functions have to be configured, e.g. by network management functions.

Figure 2 depicts the current DiffServ implementation architecture without ATM support [4]. In the ‚normal" DiffServ implementation architecture the packets are forwarded to a device, classified, processed by a service handler, which may include marking, shaping, or policing, and finally queued for transmission. Finally, a queuing mechanism such as weighted fair queuing (WFQ) or priority scheduling has to collect the packets from the different queues and schedule them for transmission over the output interface.



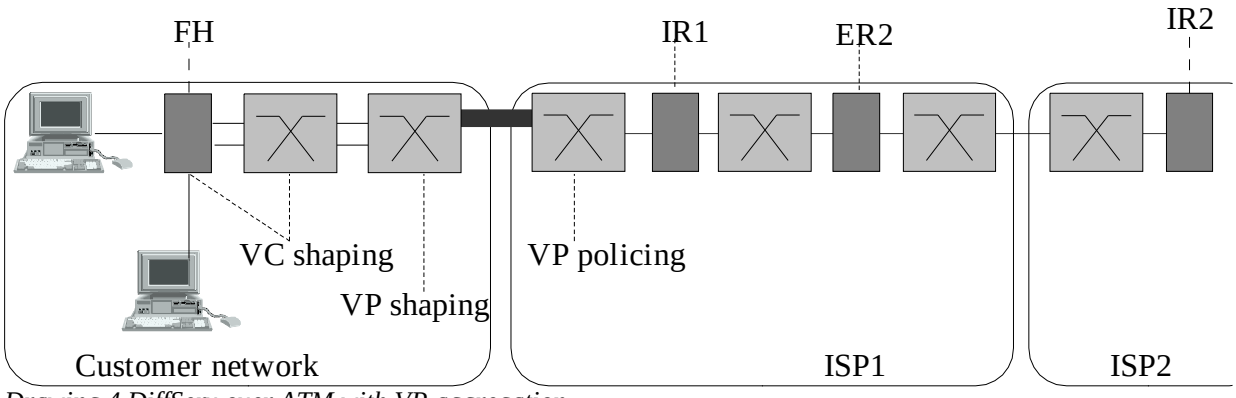
Drawing 2 DiffServ Implementation

Figure 3 shows the same architecture if the second interface is replaced by an ATM interface. In this case, the traffic is classified earlier and certain packets are directly forwarded to particular logical ATM interfaces. Since only one flow is processed by a logical ATM interface, no further classification is required. In addition, no output queuing such as WFQ is required since the packets to be transmitted over this logical interface come from a single queue. The service handlers used with this interface can be used to mark the packets to be transmitted accordingly.



Drawing 3 DiffServ Implementation over ATM (1)

The usage of ATM even allows to eliminate egress routers if each flow is shaped by an FH router. Figure 4 shows the modified scenario. Here, the FH and/or the next downstram ATM switch perform VC shaping for each EF flow. The aggregate shaping at the DiffServ domain border is then achieved by using an ATM switch which bundles several EF flows into a single PVP. On the other side, VP policing should be configured at ISP1. Figure 4 is only valid for traffic from the customer network towards the ISP. A central router will be needed for incoming traffic from ISP1 into the customer network.



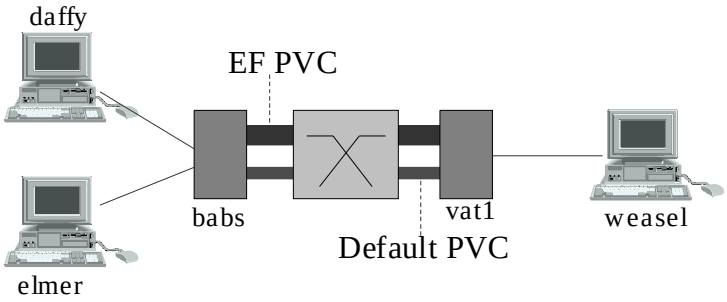
Drawing 4 DiffServ over ATM with VP aggregation

4.2.2 Router Setup

For the first of the two parts of a DiffServ over ATM router implementation iproute2 [3] can be used. This package has been installed under Linux version 2.2.9 and allows to select the outgoing router interface dependent on other parameters than destination addresses such as DSCP, port numbers, protocol identifiers and IP source addresses. For IP over ATM, the ATM−Linux package (0.59) has been used [1].

4.2.2.1 ATM Interface Selection with Iproute2

For illustration and verification of the concept, a test scenario has been implemented using iproute2. The scenario is depicted in Figure 5. Two hosts (daffy (130.92.66.141) and elmer (130.92.66.142)) are connected to a FH router (babs (130.92.66.137)) which is interconnected via a default ATM PVC (2 Mbps) and an ATM PVC (5 Mbps) for EF traffic to another router (vat1 (10.1.2.14)). Both PVCs are setup bidirectionally. The latter router is the directly connected to the receiver host (weasel, 10.1.2.2).



Drawing 5 DiffServ over ATM Implementation Scenario

Two TCP flows have been created, flow 1 from daffy to weasel, and another flow 2 from elmer to weasel. The two routers babs and vat1 have been setup in order to separate the two flows in both directions over the two PVCs 0/91 and 0/92 respectively. After setting up the Linux−ATM package [1][2], the IP over ATM interfaces have been configured on both of the routers.

```
vat1:
> ifconfig atm0 10.2.91.1 netmask 255.255.255.0 up
> ifconfig atm1 10.2.92.1 netmask 255.255.255.0 up
> atmarp −s 10.2.91.2 0.91 qos cbr,aal5:pcr=2Mbps
> atmarp −s 10.2.92.2 0.92 qos cbr,aal5:pcr=5Mbps
babs:
> ifconfig atm0 10.2.91.2 netmask 255.255.255.0 up
> ifconfig atm1 10.2.92.2 netmask 255.255.255.0 up
> atmarp −s 10.2.91.1 0.91 qos cbr,aal5:pcr=2Mbps
> atmarp −s 10.2.92.1 0.92 qos cbr,aal5:pcr=5Mbps
```

The following iproute2 commands are required to configure the routers. Iproute2 maintains different tables containing routes which can be accessed with "ip route". With "ip rule" we can define some forwarding rules based on source address, DSCP, etc. which can then be associated with those tables.

```
babs:
> ip route add 10.1.2.0/24 via 10.2.91.1      (default route)
> ip route add 10.1.2.2 via 10.2.92.1 table 1 (route in table 1)
> ip rule add from 130.92.66.141 table 1
```

The first line describes the default route for packets to be transmitted from babs to subnet 10.1.2.0/255.255.255.0. The second line adds another route for packets destined to 10.1.2.2 (weasel). In addition, the third line makes sure that the source address of these packets must origin from 130.92.66.141 (daffy).

Since we have performed our tests using TCP connections, we should also separate the reverse traffic. Therefore, we have to configure the router vat1 in a similar way as babs.

```
vat1:
> ip route add 130.92.66/24 via 10.2.91.2
> ip route add 130.92.66.141 via 10.2.92.2 table 1
> ip rule add from 10.1.2.2 table 1
```

The first line again describes the default route to network 130.92.66.0/255.255.255.0. The second line adds a route to daffy (130.92.66.141) via the second PVC. Only packets from weasel (10.1.2.2) are sent via this second PVC.

4.2.2.2 Integration of IProute and DiffServ Implementation

Another way to provide the functionality described above can be achieved by integrating iproute2 with ipchains or with the DiffServ implementation developed at University of Berne [4]. This is achieved by installing a classifier/marker either in the sending host or in the first hop router.

If the marking is done in the sending host, packets are marked first by setting the DSCP / ToS byte using a DiffServ classifier/marker in the sending host and routing the packets dependent on the DSCP in the router using iproute2. The DiffServ implementation at the sender has to be configured by the following commands:

```
> tc qdisc add dev eth0 root handle 1: serv_handler file test.tbl
test.tbl:
0xb8 130.92.66.141 255.255.255.255 10.1.2.2 255.255.255.255 0x00 0 100000
```

In this case, the Premium service flow is sent with DSCP=0xb8 to the first−hop router. The pakkets now have a ToS−byte set to 0xb8 before the routing desicion is done at the router. The iproute2 package provides ToS−based routing. In our case, it must be configured by the follwing commands which cause Premium Service packets to network 10.1.2.0 being forwarded via a special PVC.

```
> ip rule add tos 0xb8 table 1
> ip route add 10.1.2.0/24 via 10.2.92.1 table 1
```

Alternatively, the packets can be classified and marked in the first−hop router. In this case, two options exist:

1. Ipchains can be used for classification and marking before iproute2 forwarding.
2. The packets are classified/forwarded by iproute2 and marked by the DiffServ implementation.

If the packets are classified by the first−hop router, the classification and marking can be achieved by the router configuration command below using ipchains. In this case, all packets arriving at interface eth0 from daffy to be sent to weasel are tagged with DSCP=0xb8 for Premium Service. The packet is marked by ipchains [6][7] and can then be routed by iproute2 using ToS−based routing with the configuration command as described above. Ipchains can change the ToS field with the "−t tos_and tos_xor" parameter. The rule has to be inserted in the input−chain as we want to mark the packet before the routing decision is made.

```
> ipchains −A input −s daffy −d weasel −i eth0 −t 0x01 0xB8
```

If we want to use the DiffServ implementation for marking in the first−hop router, we have to install a full iproute2 classification of the IP packets in the first−hop router as described in the subsection ‚ATM Interface Selection with Iproute". Iproute2 selects the ATM interface associated with the special ATM PVC for Premium Service. The service handler queuing discipline of the Diffserv implementation is then responsible for marking packets. We have to ensure that each packet which is transmitted over the ATM interface reserved for Premium Service is marked with the Premium Service DSCP. The configuration of the DiffServ implementation is then very simple:

```
> tc qdisc add dev atm1 root handle 1: serv_handler file test.tbl
test.tbl:
0xb8 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 0x00 0 100000
```

4.2.2.3 Performance Measurements

For the performance measurements we used ttcp and ftp on the three involved end systems. The results below show that over both PVCs approximately 80 % of the configured bandwidth have been achieved. The difference is due to ATM cell (10 %) and AAL/IP header overheads. However,

the results show clearly that forwarding dependent on source addresses and shaping over ATM works well.

```
daffy:
>ttcp −t −s −n 400 −f m weasel
ttcp−t: buflen=8192, nbuf=400, align=16384/0, port=5001  tcp  −> weasel
ttcp−t: 3276800 bytes in 6.20 real seconds = 4.03 Mbit/sec +++
ttcp−t: 400 I/O calls, msec/call = 15.87, calls/sec = 64.53
ttcp−t: 0.0user 0.0sys 0:06real 0% 0i+0d 0maxrss 0+2pf 0+0csw
elmer:
>ttcp −t −s −n 400 −f m weasel
ttcp−t: buflen=8192, nbuf=400, align=16384/0, port=5001 tcp −> 10.1.2.2
ttcp−t: 3276800 bytes in 15.17 real seconds = 1.65 Mbit/sec +++
ttcp−t: 400 I/O calls, msec/call = 38.84, calls/sec = 26.36
ttcp−t: 0.0user 0.0sys 0:15real 0% 0i+0d 0maxrss 0+2pf 0+0csw
```

4.3 EF Service over ATM (second approach)

4.3.1 Architecture changes

There are two main reasons why the first approach is not satisfactory. First, the above explained architecture does not fit well in our existing DiffServ implementation as some classification has to be done within the routing decision. Second, there's a big waist of (private) IP addresses and logical interfaces as for each PVC one interface and a pair of IP adresses (IP−subnet) has to be used. To avoid this two problems we focused on a second approach. Ipchains or iproute2 is now longer used as all classification is now done by the DiffServ implementation. The new architecture does fairly differ from the architecture without ATM as shown in Figure 2. It can be used on multiple interfaces (atm0, eth0) at the same time.

4.3.2 Router Implementation

This second implementation is based on a sophisticated ATM queuing discipline which fits perfectly in our existing implementation of differentiated services. It can send traffic directly over a specified PVC without using any private IP adresses. Also, no special routing decision has to be made as this queuing discipline uses the standard outgoing interface (same as the other DiffServ components). This queuing discipline is realized as a module to the kernel in the same way as all the other DiffServ components.

Figure 6 shows the structure of our DiffServ implementation using the ATM queuing discipline for premium service in a FH router. Packets are first marked by the service handler. Then they are classified by the DS classifier which sends premium service packets to the ATM queuing discipline. There, some ATM related information is attached and the packet is sent to a FIFO or any other attached queuing discipline. The dequeue function of the ATM queuing discipline then encapsulates and sends the packets over the specified PVC and returns null. As incoming interface we still use a CLIP over ATM interface (atm0) which has the same IP address as the outgoing interface.

4.3.3 Router Setup

For further information on how to set up a DiffServ router refer to [4]. The following script shows a possible setup for a first hop router (FH) in the test scenario shown in figure 7:

```
#!/bin/sh
TC=/home/dasen/cvs/linuxds_iproute/tc/tc
DSTAB=/home/dasen/cvs/linuxds_dstab/dstab
DEV=eth0
FILE=/home/dasen/projekt/setup/ServHandler.tbl
```

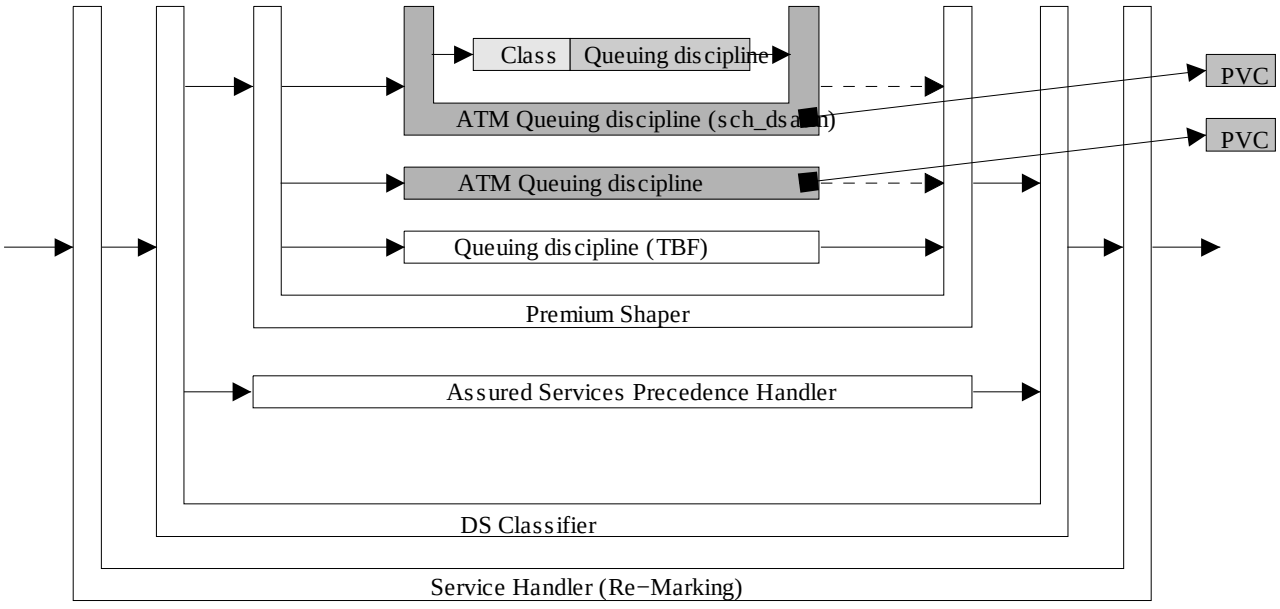
```
FILE2=/home/dasen/projekt/setup/PrecHandler.tbl
PVC=0.93
PVC2=0.94
TABID=1
TABID2=2

# Some important changes to the system
echo "1" > /proc/sys/net/ipv4/ip_forward
echo "0" > /proc/sys/net/ipv4/conf/all/rp_filter

# ATM interface
ifconfig atm0 130.92.70.92 netmask 255.255.255.255 up

#load ds table
$DSTAB l $TABID $FILE
$DSTAB l $TABID2 $FILE2

# DS Service Handler und Classifier
$TC qdisc add dev $DEV root handle 1: serv_handler table_id $TABID
$TC qdisc add dev $DEV parent 1:1 handle 2: dsclsfr asd1 0.5 asd2 0.2 \
 asd3 0.1 asd4 0.1 log_size 31

# DS ATM qdisc Premium Service
$TC qdisc add dev $DEV parent 2:5 handle 50: dsatm pvc $PVC \
 qos cbr,aal5:pcr=10mbps clip
```

ServHandler.tbl:

```
130.92.70.66 255.255.255.255 * 130.92.70.86 255.255.255.255 * * * ps 10000 1000
130.92.70.41 255.255.255.255 * 130.92.70.86 255.255.255.255 * TCP * as1 10000 1000
```

PrecHandler.tbl:

```
130.92.70.41 255.255.255.255 * 130.92.70.86 255.255.255.255 * TCP as1 * 10000 1000
```



Drawing 6 DiffServ Implementation with ATM Queuing Discipline for EF

4.3.4 Performance Measurements

All tests have been made the test−scenario shown in figure 7. As above we used ttcp for the performance measurements. Three different tests have been made. One with aggressive ATM traffic to insure that the CBR shaped PVC is not affected by any best effored ATM traffic:

Without other ATM traffic

```
ttcp−r: buflen=8192, nbuf=2000, align=16384/0, port=5001  tcp
ttcp−r: 16384000 bytes in 29.66 real seconds = 4.21 Mbit/sec +++
ttcp−r: 11316 I/O calls, msec/call = 2.68, calls/sec = 381.50
ttcp−r: 0.0user 0.3sys 0:29real 1% 0i+0d 0maxrss 0+1pf 0+0csw
```

With ~125Mbit/s unshaped ATM traffic

```
ttcp−r: buflen=8192, nbuf=2000, align=16384/0, port=5001  tcp
ttcp−r: 16384000 bytes in 29.66 real seconds = 4.21 Mbit/sec +++
ttcp−r: 11316 I/O calls, msec/call = 2.68, calls/sec = 381.50
ttcp−r: 0.0user 0.3sys 0:29real 1% 0i+0d 0maxrss 0+1pf 0+0csw
```

As expected, no difference could be seen.

The second test shows the performance using a CBR shaped PVC where shaping is performed by the ATM switch (5 Mbit/s):

```
ttcp−r: buflen=8192, nbuf=2000, align=16384/0, port=5001  tcp
ttcp−r: 16384000 bytes in 30.70 real seconds = 4.07 Mbit/sec +++
ttcp−r: 11316 I/O calls, msec/call = 2.78, calls/sec = 368.60
ttcp−r: 0.0user 0.2sys 0:30real 0% 0i+0d 0maxrss 0+1pf 0+0csw
```

The third test then shows the performance when the ATM NIC does all the shaping (different bandwidths).

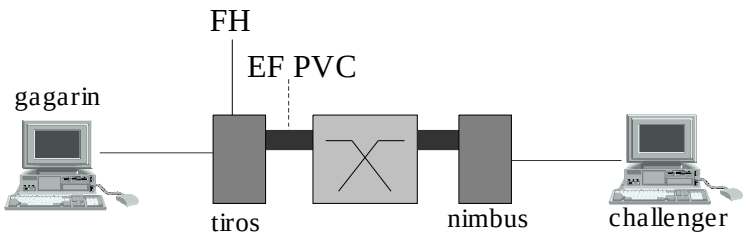
5Mbit/s:

```
ttcp−r: buflen=8192, nbuf=2000, align=16384/0, port=5001  tcp
ttcp−r: 16384000 bytes in 29.66 real seconds = 4.21 Mbit/sec +++
ttcp−r: 11316 I/O calls, msec/call = 2.68, calls/sec = 381.50
ttcp−r: 0.0user 0.3sys 0:29real 1% 0i+0d 0maxrss 0+1pf 0+0csw
```

10Mbit/s:

```
ttcp−r: buflen=8192, nbuf=2000, align=16384/0, port=5001  tcp
ttcp−r: 32768000 bytes in 28.27 real seconds = 8.84 Mbit/sec +++
ttcp−r: 22631 I/O calls, msec/call = 1.28, calls/sec = 800.49
ttcp−r: 0.0user 0.6sys 0:28real 2% 0i+0d 0maxrss 0+1pf 0+0csw
```

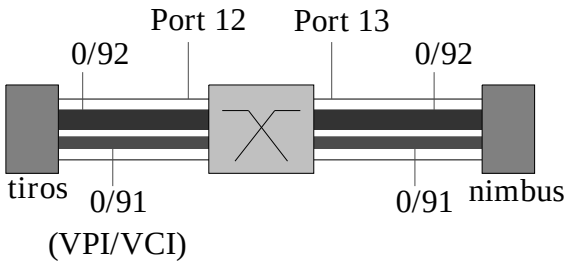
The results above are slightly better than those with the first approach. We get about 85−90% of the configured bandwidth and again, more than 90% can hardly be reached as we still have some overhead (ATM cell header, encapsulation).



Drawing 7 Test Scenario (2)

4.4 ATM Switch Configuration

In order to set up the scenario of Figure 5, two ATM PVCs have been configured at the ATM switch (a NEC ATMOIS 5SE ATM switch): PVC 0/91 is shaped to CBR with PCR of 2 Mbps, PVC 0/92 to CBR, PCR of 5 Mbps (Figure 8). The configuration has been done using the following comand line interface commands.



Drawing 8 ATM Configuration

Setting 2Mbps PVC with VPI/VIC=0/91

First, we need to setup a traffic profile for the default PVC. As we intend to establish a CBR connection we should specify only the peak cell rate. Peak cell rate is the maximum rate (in cells/sec) at which this virtual circuit can transmit. Since this PVC is a 2 Mbps line we can calculate the desired cell rate as follows :

```
 Cell rate = (BW in Mbps * 1000000 )/ (8 * Number of bytes in a cell)
           = (2 * 1000000)/ (8 * 53)
           = 4717
```

Now, to set up a CBR traffic profile we specify traffic type (for CBR it is 1) and a name of the profile (cbr2mbps).

```
# set profile 1 cbr2mbps
Peak cell rate (1-1412830[cell/s])?    4717
Profile data has been set.

Profile name:       cbr2mbps
Traffic type:       CBR
Peak cell rate[cell/s]:   4717
Sustainable cell rate[cell/s]:  -
Maximum burst size[cell]:       -
EPD:   off
```

The next step is to setup appropriate shaping profiles for this connection. Like traffic profile the shaping profile for a CBR connection can be configured with only the peak cell rate. To setup this CBR shaping profile we need to specify the slot (port) numbers (12 and 13 in our case) of the switch, shaper number (3 in both the cases) which should be between 1 and 8, and the peak cell rate (4717 cells). In this case shaping has been enabled in both directions, i.e. while traffic is passing from (i) babs to vat1 and (ii) from vat1 to babs.

```
# set shaper 12 3 4717
Shaping rate has been set.

# set shaper 13 3 4717
Shaping rate has been set.
```

Finally, we can setup the bidirectional CBR type PVC of 2 Mbps using pvc establish command. After setting the PVC the show pvc command can be used to show that the desired PVC has been setup with shaping enabled in both directions.

```
# pvc establish 0 1 12 0 91 13 0 91 0 3 cbr2mbps 0 3 cbr2mbps
# show pvc 12 0 91
```

```
Connection type: Bi-direction
Traffic type:    CBR
Line   VPI/VCI   Direction Line   VPI/VCI UPC       Shaper       Profile
12     0/91              ->     13           0/91 off     3            cbr2mbps
                         <-                         off    3            cbr2mbps

# show pvc 13 0 91
Connection type: Bi-direction
Traffic type:    CBR
Line   VPI/VCI   Direction Line   VPI/VCI UPC Shaper      Profile
13     0/91              ->     12           0/91 off     3            cbr2mbps
                         <-                         off    3            cbr2mbps
```

Setting 5Mbps PVC with vpi/vci = 0/92

Similarly a 5 Mbps CBR PVC can be setup using the following commands:

```
# set profile 1 cbr5mbps
Peak cell rate (1-1412830[cell/s])? 11792
Profile data has been set.

Profile name:      cbr5mbps
Traffic type:      CBR
Peak cell rate[cell/s]:   11792
Sustainable cell rate[cell/s]:  -
Maximum burst size[cell]:       -
EPD:   off

# set shaper 12 2 11792
Shaping rate has been set.
# set shaper 13 2 11792
Shaping rate has been set.
```

Again, after setting up the PVC we can demonstrate the establishment of the two-way shaped CBR connection by using the show pvc commands.

```
# pvc establish 0 1 12 0 92 13 0 92 0 2 cbr5mbps 0 2 cbr5mbps
# show pvc 12 0 92
Connection type: Bi-direction
Traffic type:    CBR
Line   VPI/VCI   Direction Line   VPI/VCI UPC Shaper Profile
12     0/92              ->      13     0/92       off    2      cbr5mbps
                         <-                         off    2      cbr5mbps

# show pvc 13 0 92
Connection type: Bi-direction
Traffic type:    CBR
Line   VPI/VCI   Direction Line   VPI/VCI UPC Shaper Profile
13     0/92              ->      12          0/92   off    2      cbr5mbps
                         <-                         off    2      cbr5mbps
```

5 AF Service over ATM

5.1 Architecture

For normal AF Service, a FH classifies the packets according to some profile with the correspondig AF class. In addition, the drop precedence is selected dependent on whether the flow exceeds certain bit rate limits to low, medium and high drop precedences. The same behavior is applied in the other routers with the difference that the packet classification is more coarse grained in the backbone. For mapping AF over ATM two options exist.

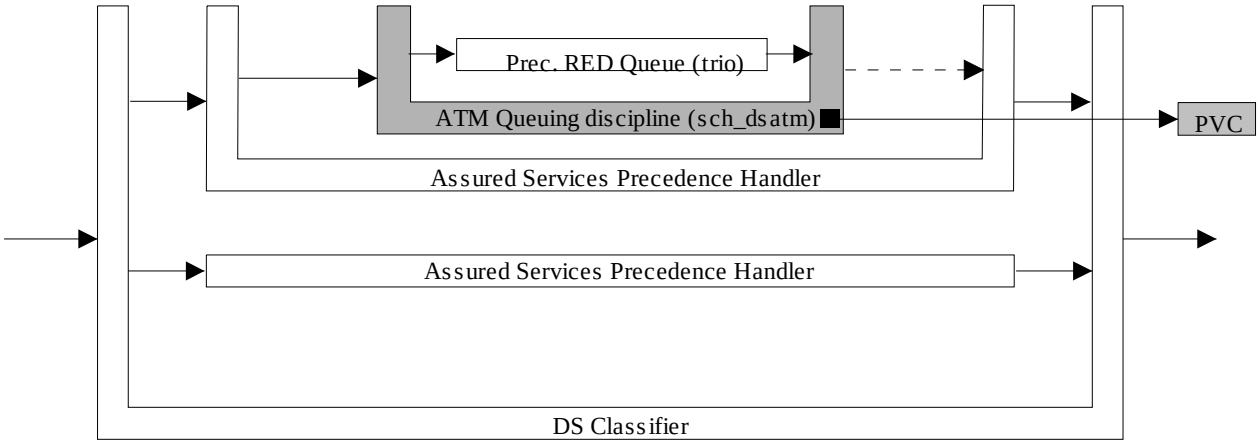
1. The first option keeps the AF router implementation with all its marking functions as it is. The AF traffic is forwarded over a certain AF PVC to the next hop. Again, the forwarding behavior must consider information such as IP source address, DSCP etc. in addition to the IP destination address. Low drop precedence packets should be mapped to CLP=0 cells, while medium and high drop precedence packets should be mapped to CLP=1 cells of this PVC. Of course, this mapping requires appropriate support via an API. The mapping ensures that ATM cells with CLP=0 are forwarded with higher probability than CLP=1 cells. The ATM PVC for AF packets can be setup with a peak rate equal to the negotiated rate. This option leaves the task of packet marking (which requires token buckets in the routers) within the routers. The implementation of this option can use the enhancements described for EF above. The configuration of the AF packet marking components (in particular the triple RED parameters) must match the ATM QoS parameters.
2. The second option tries to use the leaky bucket mechanisms implemented in ATM switches for correct marking of Assured Service packets. Two approaches are possible for the second option:
   The first one is a rather simple approach. Between an ER and an IR shaping and policing functions in the ATM switches could be used, i.e. the ATM switches mark non−conforming cells as CLP=1. At the IR, packets consisting of cells with CLP=0 should get low drop precedence while the other ones should get high drop precedence values. It has the disadvantage that only low and high drop precedence values are used, but not medium drop precedence. In addition, in the case of exceeding the peak bit rate slightly it might be that always a single cell of a packet is marked with CLP=1. This can then lead to marking more, or even all, packets with high drop precedence than in the normal DiffServ case.
   Another approach ist to mark packets in the IR with low drop precedence if the number of CLP=1 cells are below a certain threshold, while packets are marked with high drop precedence when the number of CLP=1 cells exceeds another threshold. All other pakkets are marked with medium drop precedence. Such a scheme must be made consistent to the marking schemes performed on a packet basis in IP routers. It also requires that a policing function based on CLP marking is available within the ATM switch.
   The implementation of both of these options requires to modify the CLP bits of the ATM cells resulting from AAL5 segmentation. This can only be achieved by modifying the ATM driver software, if segmentation has been implemented in software. The same is true for reassembly. If an ATM NIC performs reassembly, the CLP bits of the ATM cells will not be visible to the software driver on top of it. This means that this approach cannot be implemented over ATM NICs with ATM SAR functions implemented in hardware.

To summarize, it is more reasonable to use efficient ATM cards with SAR hardware implementations and keep the original IP enhancements for AF service. Otherwise, the introduced overhead on AAL level might increase to a similar level as for AF enhancements of the IP layer. Therefore, it might be the best if the AF implementation on IP level remains unchanged but sophisticated PVCs are provided for AF flows or aggregated flows in order to protect them from

aggressive best−effort flows. Our implementation of AF service over ATM is therefore identical with the EF service implementation over ATM with the difference that CLP marking of ATM cells is done dependent on the dropping precedence of the IP packets.

5.2 Implementation

As mentioned above we implemented the first of the two option for AF over ATM. Figure 9 shows the structure of the DiffServ implementation for assured services over ATM. The queuing dsicipline does not differ a lot from the queuing discipline for the EF described above. We actually use the same code for both services which we have enhanced according to the mapping of the dropping precedences to the CLP bit in the ATM cell header. As proposed in [8] nrt−VBR PVC should be used for AF sevices. In lack of any nrt−VBR support in the Linux−ATM driver for the Efficient ATM card functionality could only be tested (using CBR).



Drawing 9 DiffServ Implementation with ATM Queuing Discipline for AF

5.2.1 Router Setup

Again, for further information see [4]. The following possible setup script only initializes assured service class 1: (the first part of the script and the tables are shown in chapter 4.3.3)

```
# DS ATM qdisc Assured Service
$TC qdisc add dev $DEV parent 2:1 handle 10: prec_handler \
 table_id $TABID2 dscp as1
$TC qdisc add dev $DEV parent 10:1 handle 11: dsatm pvc $PVC2 \
 clip clp 0 1 1
$TC qdisc add dev $DEV parent 11:1 handle 12: trio limit 10 \
 low_begin 0.9 low_end 1.0 \
 medium_begin 0 medium_end 0.3 \
 high_begin 0.1 high_end 0.3
```

6 Known Problems, Issues, Outlook

6.1UDP−Traffic causes time−out error on interface

While testing the ATM queuing discipline with UDP traffic I’ve encountered a major problem. When sending UDP traffic with a bandwidth much higher than the shaped bandwitdh within the ATM PVC, an interface timing error will occur on the first ATM router, which results in rebooting the router. This problem does not occur with TCP, as TCP automatically adapts the bandwidth in the sender to the maximal available bandwidth.

There are probably two possible reasons for this error. First, there could be a bug in my queuing implementation (I hope not!) or second (amd more likely), this error occurs because the send operation of the ATM socket does not return immediatly and waits until the packed is physically sent (which can take a while when the queue in the ATM card is full). As this operation is called in the dequeue function, no other device can send in the meantime.

A possible solution might be, to drop the queuing mechanism in the dsatm−qdisc and to call the send operation of the ATM socket in the enqueue function.

7 References

[1] W.Almesberger: Linux−ATM, http://lrcwww.epfl.ch/linux−atm/

[2] W.Almesberger: atm/doc/usage.txt in atm−0.59.tar.gz in [1]

[3] A.Kuznetsov: iproute2, ftp://ftp.sunset.se/pub/Linux/ip−routing/iproute2−2.2.4−now−ss990824.tar.gz

[4] T.Braun, H.Einsiedler, M.Scheidegger and Karl Jonas: DiffServ Implementation Report, NEC Internal Technical Report NPDLE−HD−1999−02, September 1999

[5] R.Balmer, F.Baumgartner, T.Braun, M.Günter, I.Khalil: Virtual Private Network and QoS Management Implemenation, Technical Report, IAM−99−003, July 1999

[6] http://www.rustcorp.com/linux/ipchains/

[7] http://www.linuxdoc.org/HOWTO/IPCHAINS−HOWTO.html

[8] A.Dobreff, Differentiated Services in ATM−Networks, http://www.iam.unibe.ch/~rvs/publications/