Analysis of Tools for the Characterization of Latency and Physical Bandwidth of Internet Links[*]

Andrés Navarro[1], Marifeli Sedano[2], and Bernardo Alarcos[1]

Abstract-- **In this paper we present a comparison between several path characterization tools based on both analytical and simulation work. We start with a functional and analytical characterization of *pathchar*, *clink*, *pchar*, *nettimer*, and *a-clink*. Based on this characterization, several deficiencies are identified for the different tools, many of which are shared between the non-active tools. For this reason, the *clink* tool is selected as a representative for the non-active tools. *Clink* is then analyzed in a deeper manner in different simulation scenarios: a) under ideal conditions, b) with cross traffic, c) including the forwarding time at the nodes of the network, and d) combining these last two effects. After the analysis, we have confirmed that the problems we had identified are inherent to the end-to-end calculation technique used. These problems are, therefore, shared by all the tools which use that same technique for their calculations: *pathchar*, *pchar*, and *a-clink*. The *clink* behaviour is then compared with the performance of the active tool *a-clink* under the same simulation conditions. This will allow us to determine that error propagation, the extra load introduced on the network, and the time needed to achieve the characterization, are all reduced using programmable network technology.**

I. INTRODUCTION

IP network administration requires analysing the network behaviour beyond the network actually under the control of the network manager. This means obtaining performance and configuration data of network regions of which the network manager does not have control, and therefore lacks detailed configuration and real-time information which he may collect for his own network. To be able to gather some characteristic parameters of a network which is not under ones administrative control, many link characterization tools have been developed. These tools try to give approximate values for some network parameters, such as the link's bandwidth and propagation delays.

In 1988 Van Jacobson developed *traceroute*. This tool provides the IP address of each of the routers on the path from the source to a given destination. Later on, other tools have appeared. Some of these latest tools can give information about latency of links, their bandwidth, queuing times in each router, etc., while other just characterise the link with the smallest available bandwidth (the bottleneck link) along a path. Only the tools which characterize the latency and physical bandwidth of every link in a path have been taken into account. From this last group only some of the most relevant have been selected in order to make our analysis: *pathchar*, *clink*, *pchar*, *nettimer*, and *a-clink*.

A powerful concept we can apply to Internet analysis tools is programmable networks [1][2][3]. The traditional networks used so far confine themselves to the passive carrying of packets between different points. The nodes of the network ("passive" or "non-active" nodes) are mainly in charge of analysing the route that packets must follow and properly routing them. The main advantage introduced by programmable networks is the possibility of user-configurable processing of packets at network nodes, which become "active" nodes. This is done independently of the network protocols used. This means that nodes become generic execution environments. In programmable networks specific code can be executed at the active nodes of the network to process the different kinds of packets defined. In this environment the development of link characterization tools is proposed that benefit from the advantages supplied by programmable networks. These active tools will try to correct or just to improve the deficiencies found in the non-active ones.

In this paper[*] we make a detailed analysis of the problems introduced by the different measurement techniques, implementations and assumptions used in some relevant link characterization tools, represented by *clink* [4]. We also perform a comparative review of this tool and its corresponding version for programmable networks, *a-clink*. It is difficult to achieve this analysis over real networks, since there are many factors which cannot be directly controlled. Because of this, it is essential to employ simulation environments where all the factors are directly controllable. The performed tests will allow us to determine the accuracy of the tools and to derive conclusions from the working problems they show.

[1] Department of Automatic. Alcala University.
E-mail: {andres, bernardo}@aut.uah.es
[2] Department of Telematic Systems Engineering. (DIT). Technical University of Madrid.
E-mail: marifeli@gsi.dit.upm.es

[*] This work has been funded by MCYT under project AURAS (http://matrix.it.uc3m.es/~auras)

The rest of the paper is organised as follows. First, we present a review and analysis of the several relevant link characterization tools. Then, we provide a justification for the selection of *clink* as a consistent representative of the non-active tools. Afterwards, we describe the simulation analysis performed over *clink* and *a-clink*, using the network simulator *ns-v2*. We continue with the comparative review of the results obtained from the simulations, and finally we present our conclusions and future work.

II. REVIEW OF LINK CHARACTERIZATION TOOLS

Pathchar [5] uses the *Packet Delay* technique to calculate the bandwidth and propagation delay of each link along a given internet path. This technique is based on deriving conclusions from the delays that packets experience. In order to facilitate the understanding of some problems within this tool, we will briefly describe the fundamentals of *pathchar*.

Pathchar sequentially analyzed the links along the path, beginning with the closest one to the host running the tool. To analyze the first link, *pathchar* sends several packet streams, all with TTL set to 1. It then proceeds to analyze the second link, sending again several packet streams, but with TTL set to 2. It proceeds likewise to analyze each of the following links until all the ones in the path have been analyzed. The TTL value forces the router at the end of the link being analyzed to send back a "TTL-exceeded" packet for each packet sent by *pathchar*.

For the analysis of each link, *pathchar* sequentially sends several packet sets, each set composed of 32 equally sized packets. The packet size *s* of each sequential set is increased by 32 bytes, setting *s* to 64 bytes in the first set, and setting *s* to the path MTU in the last set. Within each packet set, *pathchar* records the time from the sending instant of each packet up to the instant in which the corresponding "TTL-exceeded" packet ("error packet") is received.

Pathchar makes the following analysis between two nodes. Before a packet leaves the *(n-1)*-th node, it waits in the queue to depart through the outgoing link. The time spent on travelling across the *n*-th link –transit time– is a linear function of the packet size (*s*). In this function the two parameters that intervene are latency (d_n) and bandwidth (b_n):

$$Transit\ time = d_n + s / b_n$$

At the *n*-th node, the packet waits again in the queue until the router processes it and generates the error packet. The error packet waits in the *n*-th node queue, and goes back to the *(n-1)*-th node. The error packet therefore also experiences a link transit time of $d_n + error / b_n$, where *error* is the size of the *TTL-exceeded* error packet (56 bytes). Finally, the error packet waits in the *(n-1)*-th node queue. The round trip time (*rtt*) from node *n-1* to node *n* and coming back is:

$$rtt_n = q_1 + (d_n + \frac{s}{b_n}) + q_2 + forward + q_3 + (d_n + \frac{error}{b_n}) + q_4 \qquad (1)$$

In this equation the q_i values are random variables that represent queuing times. *Forward* is the time spent by the forwarding element at the *n*-th node processing the received packet and generating the corresponding error packet.

In order to simplify this expression, *pathchar* makes the following assumptions:

- Forwarding time is negligible.
- Error packet size is small enough not to consider the value *error / b_n*.
- If we do a high enough number of tests for a given path, some of the samples sent might have a round trip time with negligible queuing times.

Removing these unimportant terms, equation (2) is the base of the analysis performed by *pathchar* to estimate the features of the links.

$$rtt_n = 2 \cdot d_n + \frac{s}{b_n} \qquad (2)$$

In order to calculate the estimated latency and bandwidth values of each link over a path, *pathchar* proceeds as follows. For each sample size, *pathchar* uses the smallest *rtt* value obtained to estimate the delay with zero queuing time. These smallest *rtt* values and their corresponding values of *s* define points that, according to the model represented in equation (2), should fall within a straight line. Therefore, a linear regression is used over the cloud of points in order to obtain the *latency* and *bandwidth* estimations. The estimated *latency* corresponds to half the value of *rtt,* when packet size is 0. The estimated *bandwidth* is the inverse of the slope of the straight line obtained from the linear regression. The delay values obtained for each hop in a path are cumulative and *pathchar* finds out the parameters of the subsequent links by differentiation. In the case of latency, its value at the *n*-th link is half the difference between the latency value at the current link and the value at the *(n-1)*-th link. For the *n*-th link's bandwidth, the value is the inverse of the difference between the slopes of the straight lines adjusted for the *n*-th and *(n-1)*-th links.

This measurement method is intrusive, and the mathematical model is based on some hypothesis that are not fully exact. Consequently, its results include deviations from actual values. The main problems we find in *pathchar* are:

1. *Pathchar* introduces significant extra load on the network (e.g. for a path with an MTU of 1,500 octets, *pathchar* would inject over 9 Mbits of traffic to analyze **each link** along the path.)
2. *Pathchar* is unable to detect links composed of parallel channels (inverse multiplexing). Due to this,

the bandwidth it characterises is the channel's bandwidth instead of the link's one.
3. Characterization fails if the route from the source to the destination changes during the execution time.
4. If the size of the sent sample packet is bigger than the *MTU* of a link, the packet will be fragmented. The router will therefore send an error packet as soon as the first packet arrives, causing measurements to be wrong.
5. Some routers limit the sending rate of *ICMP* packets because the router-forwarding engine gives priority to the processing of other packets. There are also routers or firewalls that may filter the generation or forwarding of these packets.
6. The values of time used to characterise the link parameters belong to the user level. If the time period measured by one of the employed timers includes a context switch in the Operating System, the obtained measure will also include the time spent on the context switch. A process usually obtains 10 ms from the processor clock between two context switches. Therefore, if the link's *rtt* to be estimated is higher than this value, the estimate will not be right.
7. As differentiation is used to calculate the data in adjacent links, there is an error propagation effect. This means that problems to characterise the *n*-th link of a path leads to a wrong characterization of the following links.

Clink, [4] is based on the same working principles as *pathchar*. *Clink* is based on the same assumptions needed for *pathchar*. And it also has the problems (2, 4, 5, and 7) derived from the data pickup technique used. But *clink* includes some improvements trying to solve some of the detected problems (1, 3, and 6).

In order to do so, *clink* reduces the load introduced on the network by decreasing the number of samples sent per packet size to 8. However, the steps between two consecutive tests with different packet sizes are reduced to 16 bytes. These two effects reduce the total number of packets sent to half the number in *pathchar*. Another improvement is that if there is more than one path between two points, *clink* shows the results of the first path that it is able to characterise. Finally, *clink* also allows including kernel timestamps, and this solves the problem of the accuracy of the results.

Pchar, [6] introduces two main modifications. The first one is that it includes new mathematical algorithms (minimum squares and Kendall statistical tests) to calculate the link's characteristic parameters. The second modification is that *pchar* extends the information provided by the tool, characterising new parameters as well as latency and bandwidth. Nevertheless, *pchar* still suffers from the problem of a high load introduced into the network to perform the characterization. As well as *clink*, *pchar* allows the use of kernel timestamps.

Nettimer, [7][8][9] unlike the other three tools previously described, is based on the *Packet Pair* technique. This technique works with two packets instead of using just one. It includes multiple working modes. Three modes characterise the bottleneck link on a path (RBPP, SBPP and ROPP techniques) and another one characterises the physical bandwidth of all links on the path (*Packet Tailgating* technique). The main differences between *nettimer* and other tools based on packet delay techniques, are that *nettimer* reduces the load introduced into the network. This becomes possible owing to its adaptive calculation algorithms. These adaptive algorithms stop sending sample packets once certain level of convergence is reached in the characterization. Another difference is that *nettimer* is able to characterise links with parallel channels. *Nettimer* solves the problems relating to *ICMP* messages by using *TCP_FIN* and *TCP_RESET* packets. *Nettimer* has some limitations to take into account. The first one is that *nettimer* is not able to characterise a fast link behind a slow one if the bandwidths ratio exceeds 37,5. Another limitation is that the nodes of the network must be FIFO for packet tailgating technique to work. Another point to consider is that errors in the employed measurements can provoke worse alterations in the obtained results than in the case of packet delay technique. As a result, *nettimer* cannot characterise distant links belonging to long paths. The technique used by nettimer works under the following assumptions: the two packets used are sent close enough in time so that they queue together, routers used FIFO-queuing, transmission delay is proportional to packet size and routers are store-and-forward.

A-clink, is based on programmable network technology and tries to solve some of the problems found on *clink*. *A-clink* first sends an *explorer* packet which finds the active routers present on a path. After that *a-clink* launches several *clink* tools at the same time in each active router on the path between source and destination. This path is divided into segments delimited by active routers. In this way, each active router estimates the parameters of the links in its segment and sends back the results to the source node. In this way, the number of links estimated by each tool is smaller. As a result, the error propagation is limited to the segment between two active routers. The characterization time and the traffic offered to the network are also reduced.

A. Selecting the tools to be simulated

With regard to the performance of the non-active tools evaluated before, we conclude that there are no significant differences in the estimation of link's bandwidth, except for *nettimer*. This tool is less aggressive considering the load introduced into the network because of the adaptive algorithms. But this tool has limitations when it is characterising fast links behind slow ones. Besides that, error propagation does not allow characterising far links over a long path. Finally, *nettimer* suffers a severe restriction, because routers must be FIFO, and this is an increasingly erroneous assumption. On the other hand, *clink* is the next less aggressively tool of the non-active ones, while providing results that are as accurate as the ones of the other non-active tools.

Based on the above considerations, we have selected *clink* as the non-active tool and *a-clink* as the corresponding one for programmable networks. The comparison criteria used are the same as stated in [9].

III. SIMULATION ENVIRONMENT

To achieve our objectives, we have simulated the *clink* and *a-clink* on *ns-v2.1b7* [10] network simulator (referenced as *ns-v2* in this paper).

In order to simulate *clink* we modified the basic behaviour of the simulator when processing packets with TTL field equal to zero at the nodes of the network**.** The other essential point to be modelled is the processing time spent in routers. This time is composed of the time to analyse the received packet and other additional times spent generating the *ICMP* error message (when necessary) and routing the packet on the proper link. After reviewing different studies [11] we have opted for modelling the processing time as a linear function of packet size. At the same time, we have endowed *ns-v2* with support for programmable networks.

We can separate our simulations into two sets. The first set is focused on achieving a performance review of *clink* tool when the ideal working conditions shown in equation (2) exist. The reason to do so is to try to establish the conditions for which the obtained results are reliable. The goal of the second set is achieving a performance review of the tools for traditional and programmable networks in scenarios under the conditions depicted in equation (1). These conditions can be easily found on the Internet links. In order to determine the influence of each factor on the obtained results we have tested the tools in the following scenarios: in presence of forwarding time, in presence of cross-traffic and in presence of a combination of these two effects.

A. *Set 1. Performance review of* clink *tool*

In order to achieve our goals, we have simulated *clink* over the network topology depicted in Figure 1 with all the nodes been non-active. There we have characterised the links of a path constituted by twelve hops. In this topology where there is neither cross-traffic nor forwarding time, we have varied the order of magnitude of the link's parameters that *clink* characterises (physical bandwidth and propagation delay). This review aims to:

- Analyse if there is any relationship between the order of magnitude of the variables being present in the mathematical model used by clink and the obtained results.
- Analyse the ability to characterise links with different values of bandwidth and propagation delay.
- Analyse the effect of transferring the limitations to the subsequent links when characterising a link.

With this network topology we have created four simulation scenarios where we have tested the performance of *clink*. We wanted to know the functioning of clink when working with different orders of magnitude of link bandwidth and propagation delay parameters: kilobits and milliseconds, megabits and milliseconds (shown in Figure 1 in normal, not bold, font), kilobits and microseconds and finally megabits and microseconds.

B. *Set 2.* clink *and* a-clink *evaluation over a real network topology*

The first goal of the analysis described in this set is to determine the influence of the factors considered as negligible in the model described in equation (1). Those factors are the queuing times q_i and the forwarding time. We also analyse the predominance of each factor in the provided results. Finally, we analyse the advantages that programmable network technology provides to link characterization tools.

To analyse these effects we have executed both tools in several scenarios over the same network topology used in set 1. However, we have changed the values of bandwidth and propagation delay to be similar to what may appear in a path of the Internet (shown in Figure 1 in bold font).

The four scenarios where we have tested how both tools work are the following ones:

- **Scenario 1.** This test is similar to the one performed in set 1.
- **Scenario 2.** There is forwarding time in the nodes of the network. This simulation allows us to determine the effect of forwarding time on the ideal model described in equation (2).
- **Scenario 3.** There is cross-traffic in the links of the network topology. This simulation allows us to determine the effect of q_i on the ideal model described in equation (2).
- **Scenario 4.** The two situations described above happen at the same time. This simulation allows us to determine which of the two introduced factors are dominant.

C. *Simulation factors under study*

Forwarding time is composed of two main times. The first one is the time that the router spends analysing the received packet, called **process time**. The second one is the time that the router spends performing the required action over the packet. This action could be generating and routing the *ICMP* error message back to the source if it exists, which needs some **ICMP time**, or routing the received packet by the proper link, which needs some **routing time**. In the simulation we have modelled a fixed

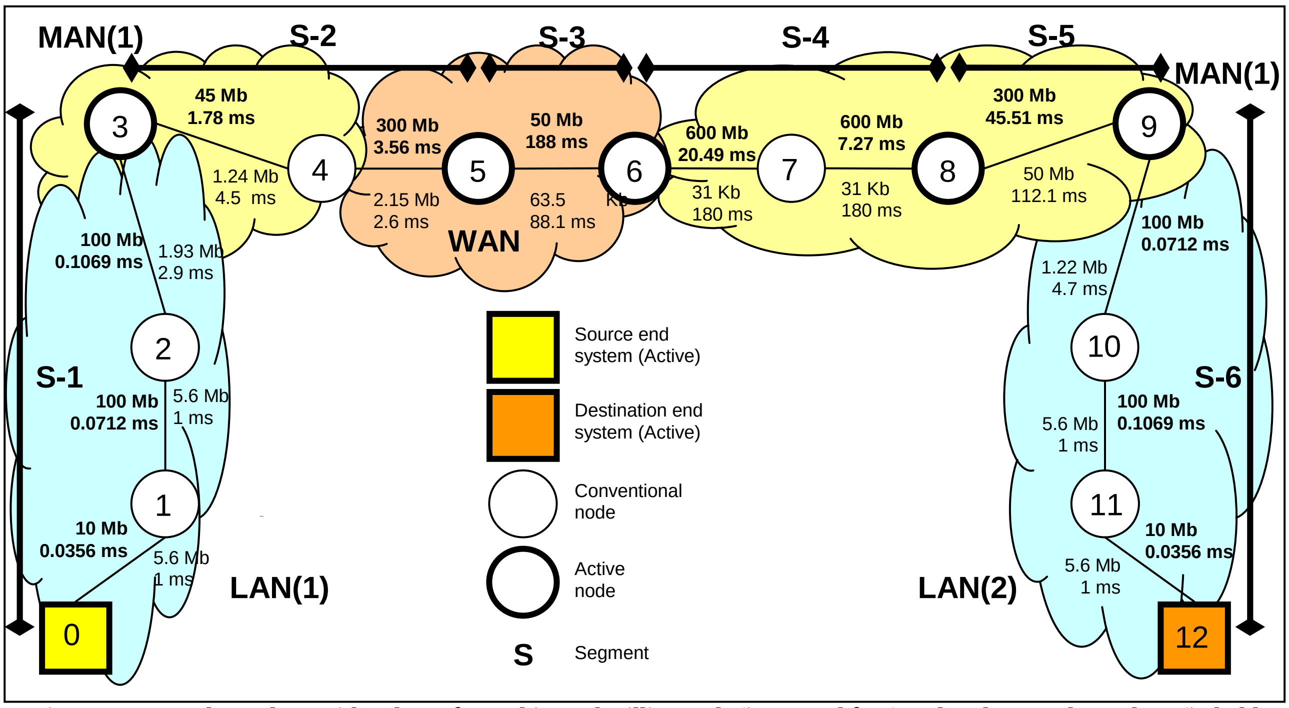

Figure 1. Network topology with values of megabits and milliseconds (in normal font) and real network topology (in bold font)

ICMP time [12] for generating the *ICMP* error message and a *process* and *routing time* proportional to the size of the received packets at nodes [11]. However, these values are just qualitative because there are not precise data about router's behaviour.

Queuing algorithms. We have used the RED (*Random Early Detection*) [13] queuing algorithm on the nodes of the network. The parameters of RED nodes used in our simulation are those proposed and used in [13].

Cross-traffic. After analysing different studies [14][15] of modelling self-similar traffic on Internet links, we have used ON/OFF [16] traffic sources. These sources have a constant sending rate during ON periods and the distribution of ON/OFF periods is given by a pareto distribution. The average link bandwidth consumed by the cross-traffic that we have introduced in our simulation is summarised in table 1. The other parameters that characterise a pareto source on a ns-v2 network simulator are summarised in table 2. The first row in the table 2 corresponds to the characterization of the sources present in all links except for the intercontinental link, whose parameters are represented in the second row.

Type	Mean Value	Direction
LAN(1)	15%	LAN to MAN
MAN(1)	50%	Bi-directional
WAN	80%	Bi-directional
MAN(2)	50%	Bi-directional
LAN(2)	15%	MAN to LAN

Table 1. Cross-traffic loads used in the simulation. Mean value is the percentage of link bandwidth consumed by cross-traffic

Size	"ON" Period	"OFF" period	Shape
700 bytes	7.5 ms	2.5 ms	1.5
700 bytes	8.5 ms	1.5 ms	1.5

Table 2. Pareto source parameters

The constant sending rate during ON periods has been adjusted so that the mean value of cross-traffic load introduced in the link is determined according to the following equation:

$$\overline{Load} = \frac{Rate \cdot \overline{ON}}{\overline{ON} + \overline{OFF}} \qquad (3)$$

IV. ANALYSIS OF RESULTS

A. clink *performance review*

After simulating *clink* in the four scenarios described in the set 1 of section 3, we realise that the order of magnitude of latency has no effect on bandwidth characterization. Latency affects displacing the straight line along the *x-axis* in the equation (2) without varying the slope. Due to this, the value of the characterised bandwidth is not influenced by the value of latency nor its order of magnitude. However, we notice that the order of magnitude of bandwidth affects the value of the characterised latency. In our simulation there is no more traffic on the network than the traffic generated by our tool. This is why the forwarding time and the q_i queuing times are zero. Furthermore, an error packet is generated as soon as a sample packet is received. So, equation (1) remains as follows:

$$rtt_n = \frac{s}{b_n} + \frac{error}{b_n} + 2 \cdot d_n \qquad (4)$$

The only difference with the model described by equation (2) is the transmission time of the error message (*ICMP_time_exceeded*). This transmission time is responsible for the error introduced in the latency characterization. It is also responsible for the dependency between bandwidth and latency.

E – Link number	**S-1 – Scenario 1**
S – Segment number	**S-2 – Scenario 2**
Real – Parameter real value	**S-3 – Scenario 3**
	S-4 – Scenario 4

E	Real	S-1	S-2	S-3	S-4
1	10	10	9.625	10	9.625
2	100	100.048	47.544	100.048	47.544
3	100	99.962	99.087	99.962	99.087
4	45	44.988	25.659	46.287	25.490
5	300	300.716	231.885	227.724	212.121
6	50	49.988	27.211	55.884	27.373
7	600	598.619	377.688	174.693	645.801
8	600	602.149	559.281	-1071.281	539.040
9	300	299.070	289.385	930.558	233.412
10	100	100.046	37.378	95.570	36.690
11	100	99.989	98.862	76.866	113.986
12	10	10	9.009	10.240	9.461

Table 3. *clink* bandwidth results (Mb)

E	S	Real	S-1	S-2	S-3	S-4
1	1	10	10	9.625	10	9.625
2		100	100.048	47.544	100.048	47.544
3		100	99.962	99.087	99.962	97.584
4	2	45	44.999	44.757	45.141	45.652
5		300	299.917	232.054	248.201	200.655
6	3	50	50.002	49.699	47.979	49.839
7	4	600	600.043	559.756	600.715	560.817
8		600	599.289	559.719	615.587	571.045
9	5	300	299.833	289.795	297.122	293.544
10	6	100	99.984	98.833	99.984	98.833
11		100	100.028	98.805	100.028	98.805
12		10	10	9.008	10	9.008

Table 4. *a-clink* bandwidth results (Mb)

E	Real	S-1	S-2	S-3	S-4
1	0.0356	0.058	0.083	0.058	0.083
2	0.0712	0.073	0.130	0.073	0.130
3	0.1069	0.109	0.114	0.109	0.114
4	1.78	1.785	1.845	1.789	1.846
5	3.56	3.561	3.574	3.561	3.577
6	188	188.004	188.065	188.033	188.085
7	20.49	20.5	20.513	20.491	20.517
8	7.27	7.27	7.275	7.275	7.280
9	45.51	45.501	45.505	45.511	45.505
10	0.0712	0.073	0.134	0.073	0.131
11	0.1069	0.109	0.114	0.099	0.120
12	0.0356	0.059	0.115	0.058	0.215

Table 5. *clink* propagation delay results (ms)

E	S	Real	S-1	S-2	S-3	S-4
1	1	0.0356	0.058	0.0356	0.058	0.083
2		0.0712	0.073	0.0712	0.073	0.130
3		0.1069	0.109	0.1069	0.109	0.113
4	2	1.78	1.785	1.78	1.787	1.792
5		3.56	3.561	3.56	3.560	3.573
6	3	188	188.004	188	188.008	188.013
7	4	20.49	20.5	20.49	20.5	20.505
8		7.27	7.27	7.27	7.271	7.275
9	5	45.51	45.501	45.51	45.501	45.506
10	6	0.0712	0.073	0.0712	0.073	0.078
11		0.1069	0.109	0.1069	0.109	0.114
12		0.0356	0.058	0.0356	0.058	0.115

Table 6. *a-clink* propagation delay results (ms)

Since the error packet size is constant, the transmission time added to the *rtt* value of each sample packet is also constant. This added value depends directly on the bandwidth of the characterised link. As a result, the straight line that models *rtt* time is displaced upwards along the *x-axis* by the value of the transmission time of the error packet. Accordingly, the latency is the link's real latency plus half the transmission time of the error packet. The smaller the bandwidth of the characterised link, the more noticeable this effect becomes. It is particularly important when the transmission time of the error packet is greater than twice the link's latency.

The second feature we have analysed is the ability to characterise links with values of bandwidth and propagation delay of several orders of magnitude. Considering the obtained results we can remark that the smaller the bandwidth the better the characterization is. This is due to the fact that in links with small bandwidth the *rtt* line slope is a big value. Thanks to that, the mathematical calculations needed to obtain the bandwidth (inverse of the *rtt* line slope) from the *rtt* line require less accuracy than in those cases where the *rtt* line slope is smaller. This can be noticed in any of the links with bandwidths greater than 10 Mbps of the network topology. For latency we do not have calculus limitations, but for the limitation imposed by the transmission time of the error packet described before.

Finally, we have analysed the error transferred to the characterization of a link that follows a link with high bandwidth. This is a direct consequence of the situation described in the paragraph above. We confirm that the differentiation technique employed in *clink* leads to the fact that errors propagate to the subsequent links.

B. clink simulation over a real network topology

The results corresponding to the analysis and conclusions presented in this section are summarised in tables 3 (bandwidth values) and 5 (propagation delay values).

In ***scenario 1*** **(S-1),** as it has been concluded from the set of tests in the previous section, we notice that the obtained latency (table 5) is increased to half the transmission time of the *ICMP* error packet. We also notice that the limitations to characterise links with high bandwidth (table 3) cause an error propagation when characterising the following links as if they would also have high bandwidth values.

In ***scenario 2*** **(S-2),** the values of latency in all links are higher than the values obtained in scenario 1. The obtained latency is also higher than the real value because of the *ICMP* time simulated on nodes, which is constant with packet size. Regarding the values of bandwidth, they are all smaller than the real value. This is due to the process and routing times that are present in all nodes, which are proportional to the packet size. These added times make the *rtt* line have a higher slope. Consequently, the link bandwidth (the inverse of the line slope) is smaller than the real one. After this simulation we observe that the most damaging and noticeable effect on the forwarding time is the process time because of its dependency on the size of received packets.

In ***scenario 3*** **(S-3),** the values of latency are not altered much, and the increase is in the order of microseconds. In the case of bandwidth, when the traffic load introduced is small as in LAN(1), the characterization is perfect. It is even correct when the link is loaded with cross-traffic in the reverse direction of *ICMP* packets. The effect of cross-traffic is significant in MAN(1), where we have the accumulated effect of the 15% LAN cross-traffic load and the 50% MAN cross-traffic load. In this case we have the effect of cross-traffic combined with the error propagation to subsequent links (45 Mbps on link number 4, characterised as 46.287 Mbps and 300 Mbps on link 5, characterised as 227.724 Mbps). In the intercontinental link and the subsequent links of MAN(2), we can more easily see these effects. In these links we have noticed that the combination of the effects described above is more damaging when the introduced traffic load is higher. We have also observed that the main error is not noticeable in the characterization of the link that supports high cross-traffic load, but in the subsequent links. This is due to the accumulation of queuing times in the *rtt* time of the samples. So the 600 Mbps link 7 is characterised as 174.693 Mbps, the 600 Mbps link 8 is characterised as -1071.281 Mbps (this negative value is due to the differentiation employed in packet delay technique) and the 300 Mbps link 9 is characterised as 930.558 Mbps. This confirms that with a high network load and for links that are far away from the source end system, an approximate characterization of the links is impossible due to error propagation.

In ***scenario 4*** **(S-4),** we have combined both effects. We notice that for links near to the source end system and with small traffic load in the network, the main effect that introduces error in the characterizations is the forwarding time. Nevertheless, for links that are far away from the source end system and with high traffic load in the network, the error propagation caused by random queuing times makes their correct characterization impossible. This is because of the accumulation of the error introduced by sample packets waiting at the link's queues. So we realise the prevalence of cross-traffic over forwarding time error in far links.

C. a-clink *simulation over a real network topology*

The results corresponding to the analysis and conclusions presented in this section are summarised in tables 4 (bandwidth values) and 6 (propagation delay values).

In ***scenario 1*** **(S-1),** we observe that the problems are the same as for *clink* under the same conditions. Now the effects of the error are limited to each characterization segment.

In ***scenario 2*** **(S-2),** we can observe that the bandwidth characterization of the first link in a segment is very close to the real value. That happens because we are only including the first router forwarding time. Thanks to this, we manage to isolate the effects produced by the abrupt changes in the speed of the links connected by the router. In case of latency the same effect occurs as in scenario 2 for *clink*.

In ***scenario 3*** **(S-3),** we find the same effects in every characterization segment as we find for *clink*. These effects take place if *clink* has been launched from the first node of the segment. We must remark that in the 50 Mbps intercontinental link, the effect of 80% traffic load is not very significant because now we do not have the queuing times of the previous links added to the characterization of this link. This allows us to confirm again the conclusion that the effect of a high traffic load is even more damaging in the next link characterization than in the link that suffers such high cross-traffic load. Furthermore, the characterization of the remaining links is absolutely right, unlike what happens with *clink* tool.

In ***scenario 4*** **(S-4),** the predominant effects described for *clink* are perceived again in the segments that include LAN(1) and MAN(1) networks. The characterization of those links that are far from the source is correct because they are included in the new segments when we use *a-clink*. The effect of the forwarding time is still predominant over cross-traffic, because the error due to cumulative queuing times is isolated.

D. *Comparative review*

In respect to the effect of the forwarding time when characterising the links, we observe the first advantage provided by programmable network technology. The effect of processing time is reduced when the characterization paths made by *a-clink* locates in two different segments the links connected by a router that has high processing time. In this situation the characterised bandwidth tends to be lower than the real one. This is because *a-clink* starts a new characterization in such a router. In the case of latency the characterised value is also closer to the real one, as analysis time in the router is eliminated.

As it happened in the previous case, the propagation of possible errors is reduced by *a-clink* to just a characterization segment. This is because we get rid of the cumulative queuing times. As a result, the values of latency and bandwidth provided by *a-clink* for the links that follow the intercontinental link are completely reliable, unlike what happens with *clink*.

The distribution of the load introduced in the network to achieve links characterization is better with *a-clink*. This is because this characterization is performed from each active node to the next active node in the network. In case of *clink*, the links that are close to the source end system, that starts the characterization, support a high traffic load. The time spent for characterising a network topology is smaller in the case of *a-clink*. This is due to the fact that the characterization of the different segments, in which the topology is divided, is launched in parallel, once the characterization process is started.

V. CONCLUSIONS AND FUTURE WORKS

Based on the simulation results of the performance of the tools *clink* and *a-clink*, we can conclude that most of the detected errors of *clink* are inherent to the packet delay characterization technique used. These errors are shared by all the tools that use this same technique for their calculations: *pathchar*, *pchar*, and *a-clink*. One of the lines of improvement in this sense may be the improvement of the algorithms employed in processing the collected data, in order to avoid the amplification of possible errors, as it happens now with the election of extreme values for the calculations.

As for *nettimer*, it avoids the amplification problem, but it is based on the hypothesis of FIFO routers, hardly valid today, and it has severe limitations when characterising fast links that are later in the path than slow links.

a-clink shares the problems of clink in respect to error amplification and overload, but in a far less severe manner. This is due to the partition of the path in active segments, being able to analyse each path with fewer hops, and doing it in a parallel manner.

Finally, the usage of *ICMP* messages that are fully filtered in an increasing number of networks may bring these tools to become unusable. An interesting line of work could be to base their analysis on regular data packets combined with diagnostic or control packets from protocols at other layers (e.g., TCP control packets).

REFERENCES

[1] D.L. Tennnenhouse, J.M. Smith, W.D. Sincoskie, D.J. Wetherall and G.J. Minden. “A survey of Active network Research,” *IEEE Communications Magazine*, pp. 80-86, January 1997.

[2] A. T. Campbell, H. G. De Meer, M. E. Kounavis, K. Miki, J. B. Vincente and D. Villela. “A survey of programmable networks”, *Computer Communication Review*, 29(2):7-23, April 1999.

[3] J. T. Moore and S. M. Nettles. “Towards Practical Programmable Packets,” *Technical Report MS-CIS-00-12*, May 2000.

[4] Allen B. Downey, “Using pathchar to estimate Internet link characteristics,” *In Proceedings of ACM SIGCOMM*, pp. 241-250, September 1999.

[5] V. Jacobson, “Pathchar - A tool to infer characteristics of Internet paths,” *Mathematical Sciences Research Institute (MSRI)*, April 1997.

[6] Bruce A. Mah, “pchar: a Tool for Measuring Internet Path Characteristics,” http://www.employees.org/~bmah/Software/pchar/, 2000.

[7] K. Lai and M. Baker, “Measuring Bandwidth,” *Proceedings of IEEE INFOCOM*, March 1999.

[8] K. Lai and M. Baker, “Measuring Link Bandwidths Using a Deterministic Model of Packet Delay,” *Proceedings of ACM SIGCOMM*, August 2000.

[9] K. Lai and M. Baker, “Nettimer: A Tool for Measuring Bottleneck Link Bandwidth,” *Proceedings of USENIX Symposium on Internet Technologies and Systems*, April 2001.

[10] S. McCanne, S. Floyd, K. Fall and K. Varadhan et al, “Network Simulator, *ns*”, http://www-mash.cs.berkeley.edu/ns/.

[11] David Newman, “Cisco’s Catalyst 6500 raises the stakes,” *NetWorldFusion*, 2003, http://www.nwfusion.com/reviews/2003/1020cisco10gbe.html.

[12] R. Govindan and V. Paxson, "Estimating Router ICMP Generation Delays," *In Proceedings of Passive and Active Measurements (PAM)*, Mar. 2002, pp. 6-13.

[13] S. Floyd and V. Jacobson, “Random Early Detection Gateways for Congestion Avoidance,” *IEEE/ACM Transactions on Networking*, vol. 1,4: pp. 397-413, August 1993.

[14] V. Paxson and S. Floyd, “Wide-Area Traffic: The Failure of Poisson Modeling,” *IEEE/ACM Transactions on Networking*, 3(3), pp. 226-244, June 1995.

[15] M. E. Crovella and A. Bestavros, “Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes”, *IEEE/ACM Transactions on Networking*, 5(6), pp. 835-846, December 1997.

[16] W. Leland, M. Taqqu, W. Willinger and D. Wilson, “On the Self-Similar Nature of Ethernet Traffic (Extended Version),” *IEEE/ACM Transactions on Networking*, 2(1), pp. 1-15, February 1994.