Inferno Nettverk A/S

Technical Report

Analysis of the *session limit* functionality in Dante version 1.3.2.4

August 6, 2012

Contents

List of Figures

1 Introduction

This report examines the behavior of the *session limit* functionality present in version 1.3.2.4 of the Dante SOCKS server implementation from Inferno Nettverk A/S. This version was only made available to a customer but the functionality described in this document will also be found in Dante 1.4.0.

Part of this functionality is new to the 1.3.2.4 version of Dante and makes it possible to control the resource usage of Dante by allowing the Dante administrator to limit the *number* of concurrent client sessions and the *rate* client sessions can be established at.

Individual session limits can be enforced on both the total number of sessions, regardless of what client IP address they connect to Dante from, and on a group of sessions coming from the same client IP address.

1.1 Session limit functionality

Dante is a SOCKS proxy server that typically receives requests from SOCKS clients that wish to communicate with target servers that are only accessible to the clients via the Dante server.

The two most common usage scenarios is the Dante server providing access control for sessions going *out* of a local network, or providing access control for sessions going *in* to a local network.

In this latter case, the Dante server might be accessible from any host on the public Internet, making *Denial-of-Service* (DOS) attacks a potential threat. This type of attack might be attempted by initiating a large number of connections to the Dante server, potentially reducing the amount of resources available to other users and increasing the resource load generated by Dante as it tries to handle the large number of connections.

Similar problems can also be the result of less deliberate behavior, where accidental configuration errors by legitimate users can also result in many connections being initiated in a short time frame from a single machine.

Any new session handled by the Dante server requires file descriptors, memory, CPU time, and depending on the existing load, the creation of new processes.

While the session functionality in Dante cannot eliminate DoS-like problems, it can limit the effects of a DoS attack, both by enforcing limits on the maximum number of sessions that can be established and the rate they can be established at.

Two types of limits can be imposed via the session functionality: maximum concurrent session limits, and rate of session establishment limits.

Limits may be specified in any of Dante's access control rules and can be specified both as a total limit for all clients matching the rule, and as a limit only on clients from the same IP address.

The per IP address limit can use either the SOCKS clients IP address, or any *hostid*[1] address set on the TCP session established by the SOCKS client to Dante.

2 Session limitation behavior

The different session limit types can be used in several different ways. To show the effect of these limits on client connections, a set of tests are undertaken using different server configurations and client loads. Each test follows the same general pattern: a client application connects to the Dante SOCKS server and requests multiple connections to a target server. The connections are requested at a constant fixed rate for as long as the application is active. Connections that have been opened are kept open until the application is terminated after a fixed number of seconds.

In the tests, two instances of the application are run, with each application instance connecting to Dante from different IP addresses, and requesting connections to different target servers. In all tests, the IP address each client connects to Dante from is fixed, and so is the address of the target server it requests a connection to.

There are no access control rules in the SOCKS server that would result in any of the requests being refused, meaning that without any session limitations each client would be successful in all connection attempts. The server configurations that are tested thus show how the session functionality can be used to limit the client behavior.

To observe the behavior of the clients and SOCKS server, the standard *tcpdump* network monitoring program is used to capture traffic both to and from the SOCKS server. The initial SYN packet used to initiate TCP connections are captured for the connections made by the client to the SOCKS server, as is the initial SYN packet made from the Dante server to the target servers. The timestamps of these packets are then used to visualize the system behavior.

[1] A *hostid* address is an address provided out-of-band and is typically used in usage scenarios with multiple layers of proxies to store the identity of the client that initiated the connection to the first proxy in the chain. Currently this functionality is not part of the standard TCP/IP specifications and the support for this type of functionality in Dante is based on a proprietary Linux kernel patch.

2.1 Maximum number of concurrent sessions

One of the simplest configurations are shown below, with a single *session.max* limit of 50 sessions specified in the *client-pass* rule:

```
client pass {
        from: 0.0.0.0/0 to: 0.0.0.0/0
        session.max: 50
}
```

At any time, at most 50 sessions will be allowed with this configuration, which in the test environment corresponds to 50 open TCP connections. When this number of active connections has been reached, the Dante server will block any new session attempts until one of the existing sessions are closed. As soon as the number of active sessions falls below 50 clients will be able to establish new sessions, until the limit of 50 is reached again.

The limit is placed in a *client-rule*, which will cause it to apply from the time the connection is received (a limit placed in a *socks-rule* would not apply until after completion of SOCKS protocol negotiation).

To test this configuration, two clients are used, each initiating new connections to the Dante server at a rate of three connections per second. Client A is started first and runs for 20 seconds. Client B is started 10 seconds after Client A and runs for 30 seconds. This gives an overlapping period of 10 seconds where both clients are active and attempt to establish new sessions. Both clients keep the sessions they have managed to establish with the Dante server open until the clients terminate.

Figure 1 shows the behavior that results with this combination of client behavior and server configuration. For the first 10 seconds, only Client A is active and the rate of new sessions established with the Dante server is the same as the attempted connect rate by Client A, three per second. After 10 seconds, Client B also starts connecting to the Dante server, and the combined rate of new connections is doubled to six per second.

In the beginning, Client B is also able to establish one session with the Dante server for each connection attempt, but the maximum limit of 50 active sessions is now quickly reached, at around 13 seconds after the test started, resulting in around 40 sessions with Dante for Client A and 10 sessions with Dante for Client B.

Both clients continue to open new connections at an unchanged rate as long as they are running, but neither client is able to establish any new sessions[2] via the

[2]By *session* we here refer to SOCKS sessions. The clients are naturally able to open TCP connections to the Dante server, but Dante immediately closes the connections after having determined that the session limit in the client-rule that matches the client has already been reached.

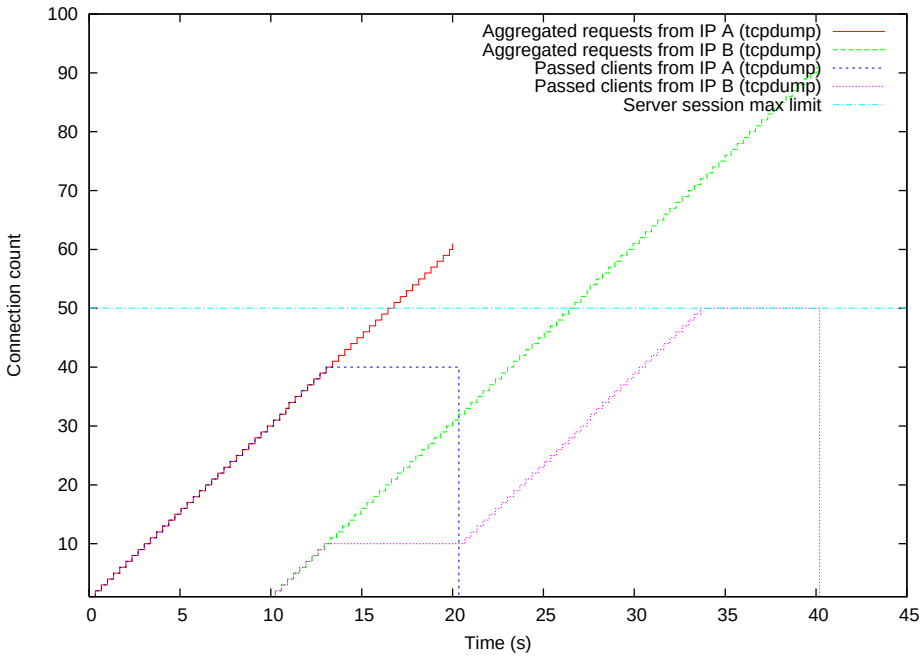

Figure 1: Effects of session max limit

Dante server due to the maximum session limit.

After having run for 20 seconds, Client A closes all connections to the Dante server, causing all of its sessions to be terminated. Because the total number of active sessions now falls below the maximum limit of 50, Client B is able to establish new sessions until the maximum limit is reached again. Client B maintains the connections it has been able to open until it terminates, but is unable to open any new connections during this time.

In short, the session maximum limit places an upper limitation on the number of sessions that that can be established via the Dante server. There are no limitations on connection rates or the number of connections that can be opened from any single IP address.

Significantly increasing the rate at which connections are initiated gives a similar result, but with less overlap between the clients. Figure 2 shows an example of this, with each client creating connections at a rate of 300 connections per second. The plot uses a logarithmic Y-axis for better readability.

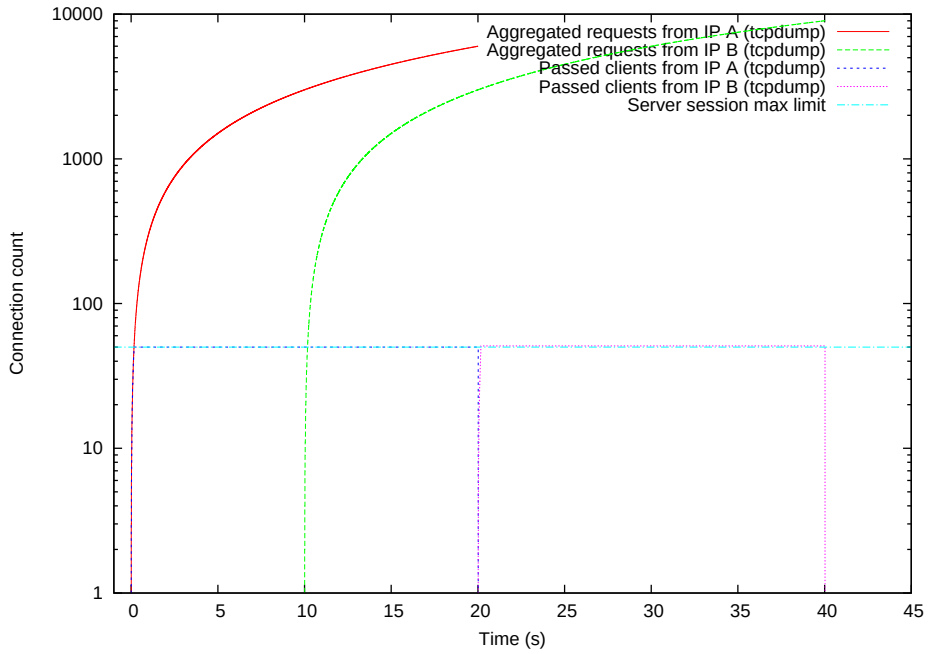

Figure 2: Effects of session max limit, high connection rate

When Client A start initiating connections, it fills the entire available capacity of the limit of 50 sessions within the first second. When Client B starts 10 seconds later, it is unable to open any connections due to no capacity being available. As before, both clients keep initiating new connections to the Dante server, but none of them result in established sessions.

It is only after 20 seconds, when Client A closes all its sessions with Dante that capacity again becomes available. Client B then immediately fills this capacity, having 50 open connections until it exists 20 seconds later.

2.2 Throttle limit

A throttle limit enforces a fixed upper limit on the rate at which new sessions with the Dante server can be initiated. The configuration below is an example of this, setting a limit of a maximum of three new sessions every two seconds:

```
client pass {
        from: 0.0.0.0/0 to: 0.0.0.0/0
        session.throttle: 3/2
}
```

This limit is again placed in the *client pass* rule, meaning that it will not accept new connections at a rate higher than three connections every two seconds. Connections will be blocked and dropped immediately after having been accepted, before SOCKS protocol negotiation. If connections are received at a higher rate, any new connections will be blocked until two seconds has passed.

Again, the operation of this type of limit is illustrated using a setup with two clients. Both clients open three new TCP connections to the Dante server every second, giving a combined rate of six connections per second, four times that permitted by the Dante configuration.

The resulting behavior is shown in Figure 3. Client A starts at around 0 seconds. Unlike when the *session.max* limit was enforced, as shown in Figure 1 and Figure 2, the client is this time able to establish new sessions with Dante continuously, but at a rate lower than the one attempted.

When Client B also starts opening connections after 10 seconds, the rate at which Client A is able to establish sessions is further reduced, but both clients are still able to continuously establish new sessions.

This continues until 20 seconds after start, when Client A terminates, leaving Client B alone to open connections until it terminates 20 seconds later. With a sharp eye one can might even see from the figure that the rate at which Client B establishes new sessions increases after Client A terminates.

As expected, unlike with the *session.max* limit, there is no limitation on how many connections can be opened, and both clients are able to continuously establish new sessions at the server specified rate.

2.3 Per IP address maximum limit

Per IP address limits can reduce the impact a single user/machine can have on other users, assuming the machine has a small number of IP addresses.

The following example shows how this can be configured. Unlike before, the session limit set in the following example, with *session.state.max*, is applied based on IP addresses, with *session.state.key* used to specify which IP address to use.

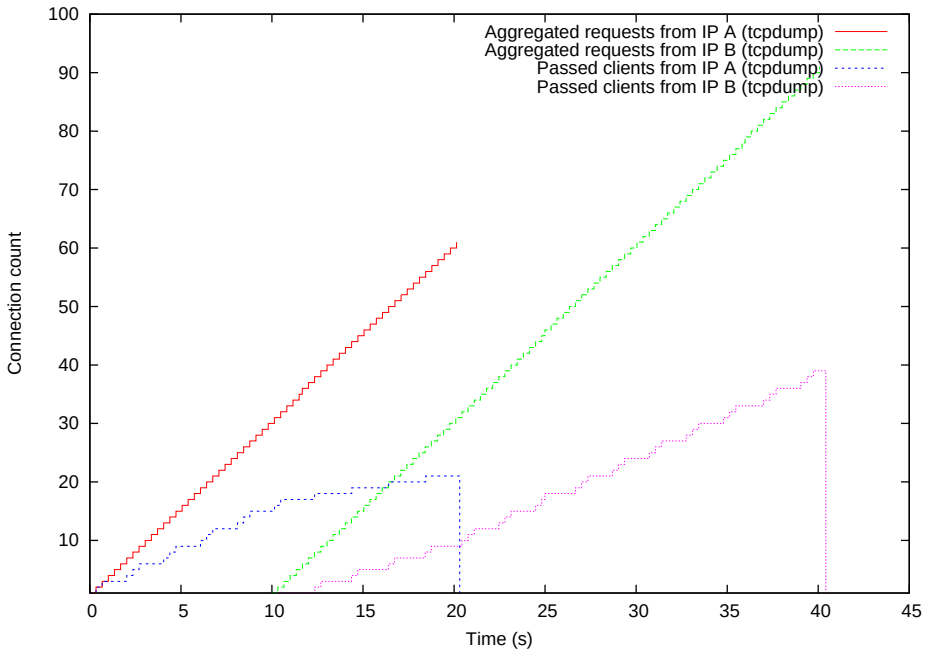

Figure 3: Effects of throttle limit

When the session state key is *from*, the IP address of the client connecting to Dante is used (an alternative key would be *hostid*).

The following rule thus limits the number of active connections from a single IP address to 30, with no combined total limit for all IP addresses.

```
client pass {
        from: 0.0.0.0/0 to: 0.0.0.0/0
        session.state.key: from
        session.state.max: 30
}
```

The resulting behavior is shown in Figure 4. Client A starts opening connections, and as there is no rate limit it is able to establish new sessions via the Dante server without being blocked until the per IP limit for the client is reached.

In the figure, one can see that the lines for “Aggregated requests”, counting the number of connection attempts to Dante by the client, and “Passed clients”, counting the sessions successfully established by the client via Dante, overlap

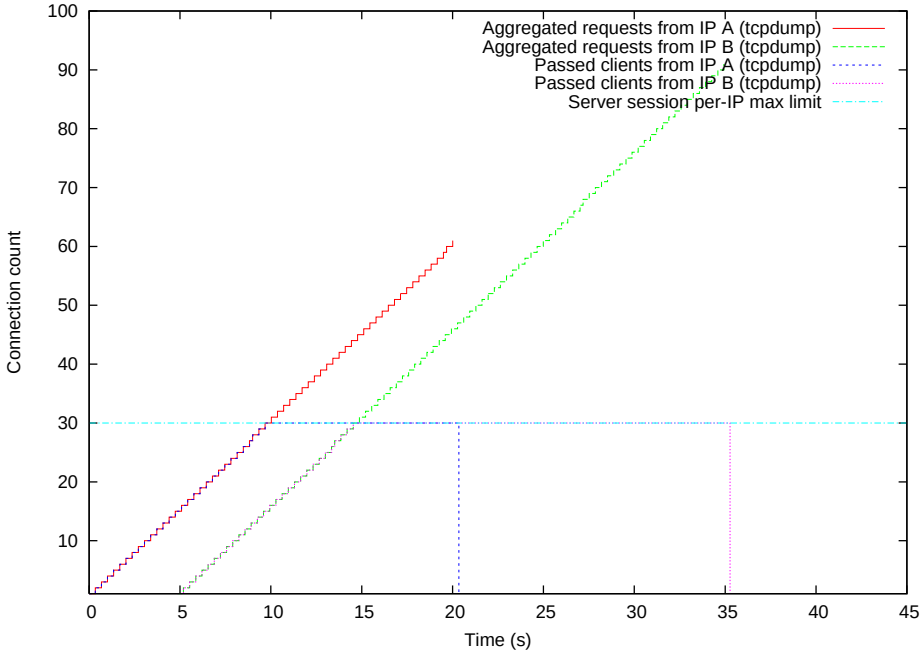

Figure 4: Effects of per IP address limit

until the maximum limit is reached, after which the “Passed clients” line no longer increases.

Client B starting at time 5 does not have any visible effect on the rate at which Client A can establish sessions with Dante, and Client B can also establish sessions with Dante at its full rate until the maximum rate is reached.

Likewise, Client A terminating at time 20 does not have any affect on the number of sessions Client B can establish; Client B has established 30 sessions when Client A terminates, and even though Client B continues trying to establish sessions (as seen from the “Aggregated requests from IP B” line), it is unable to establish any more sessions.

This differs from the behaviour seen in Figure 1, where it can be seen that as soon as Client A has terminated, Client B is able to establish new sessions.

The two clients, each using an unique IP address, are in total able to establish 60 sessions with Dante, but neither client is at any time able to establish more than 30 sessions.

Compared to the behavior with the maximum limit described in Section 2.1

above, there is no upper limit on the total number of active sessions, only a per IP address limit due to the lack of a *session.max* keyword

If there are clients from many different IP addresses, the total number of active sessions can still become large, but placing a limit on the number of sessions from a single IP address makes it possible to reduce the room that a machine with a single IP address has for consuming an undue amount of Dante's resources.

2.4 Per IP address throttle limit

A per IP address throttle limit controls the rate at which connections can come from a single machine/IP address. The following example shows a configuration that limits the sessions from any IP address to three per every two seconds:

```
client pass {
        from: 0.0.0.0/0 to: 0.0.0.0/0
        session.state.key: from
        session.state.throttle: 3/2
}
```

The effects of applying this configuration to the usage scenario of our two clients initiating connections at a fixed rate of three connections every second can be seen in Figure 5.

This plot is very similar to the one in Section 2.2, which uses *session.throttle* to set the same rate limit for all connections. The primary difference is that with a per IP throttle limit there is no effect on client B each time a new session is established by Client A, and Client A has no effect on Client B.

Client A starts opening connections at time 0, and while the throttle limit reduces the rate it can establish sessions at, it is able to establish new sessions at a fixed rate until is terminates at time 20. The same is the case for Client B, which starts 10 seconds after Client A and is able to continue establishing sessions at the same fixed rate until it terminates at time 40.

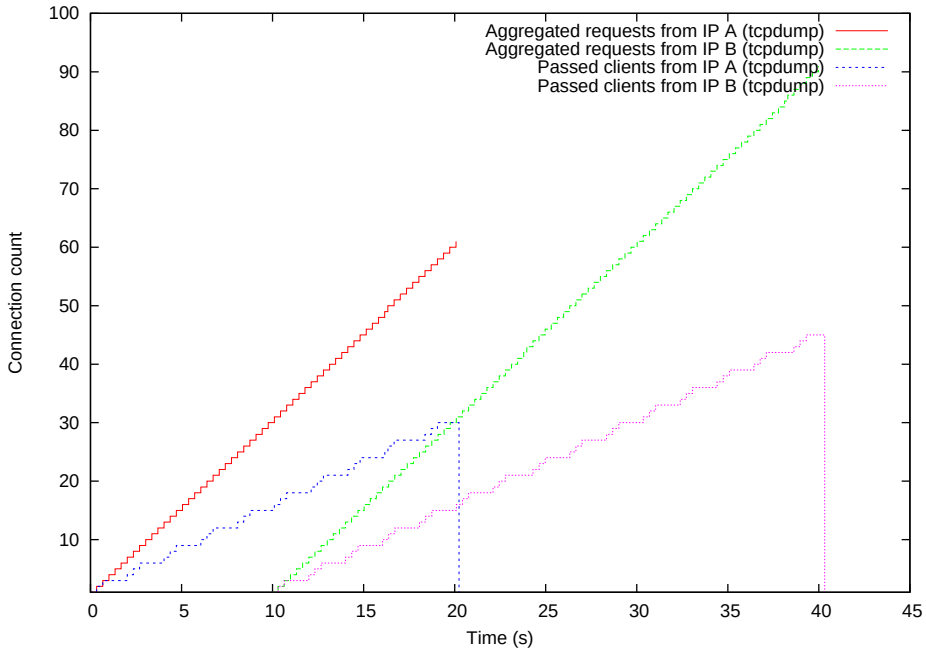

Figure 5: Effects of per IP throttle limit

2.5 Combined limits

Using the different session limits separately can be useful, but they can also be combined for better control. The client-rule below combines all four types of keywords to be able to control both the maximum number of sessions and the establishment rate, both for any given IP address and for all clients in total:

```
client pass {
        from: 0.0.0.0/0 to: 0.0.0.0/0
        session.max: 55
        session.throttle: 4/1
        session.state.key: from
        session.state.throttle: 5/2
        session.state.max: 30
}
```

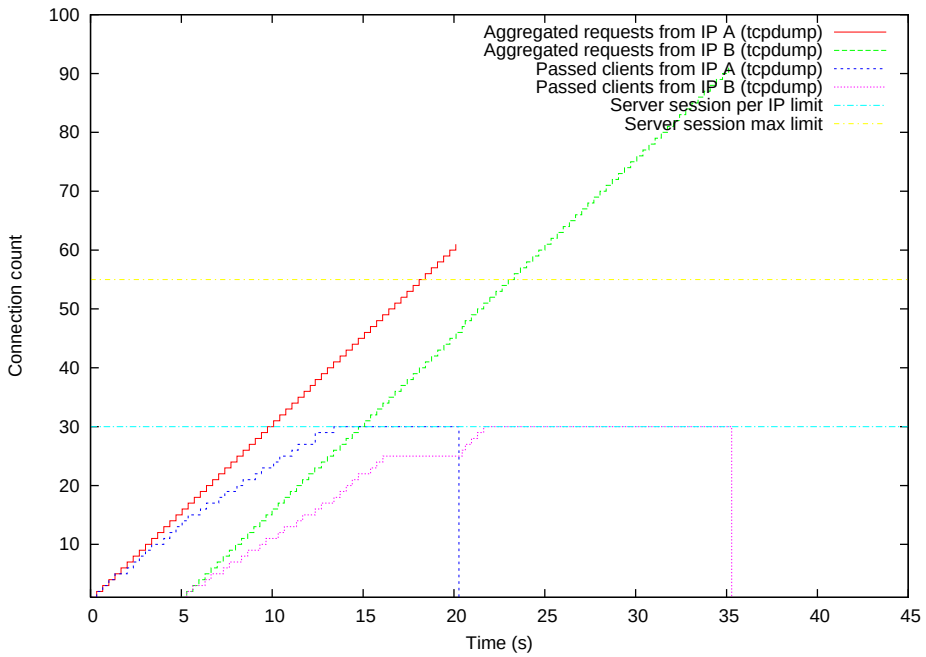

Figure 6: Effects of combined limits

The behavior with all these limits combined is shown in Figure 6. Here both the maximum limits and the per IP throttle limit affect the clients.

Client A starts opening connections at time 0, attempting to establish three sessions per second. During the first five seconds the per IP limit of five sessions per two seconds set in *session.state.throttle* limits the rate of Client A only slightly, while the total rate limit of four sessions per second does not affect Client A since the client only attempts to create sessions at a rate of three per second.

At time 5 Client B also starts up, and it can be seen that the rate Client A is able to establish sessions at is now reduced considerably. This is due the total throttle limit, *session.throttle*, of four sessions per second taking effect because of the added load from Client B, as the combined rate Client A and Client B attempt to establish sessions at is six sessions per seconds, considerably larger then the total limit of four sessions per second.

Client A is able to continue establishing new sessions at the same rate until it at approximately time 14 reaches the per IP limit, *session.state.max*, of 30 sessions.

Client B also starts in the same way, but at around time 16 , the total limit for all clients, *session.max*, 55, has been reached. Client B has at this time only been able to open 25 sessions, as Client A starting five seconds earlier has managed to occupy the other 30 sessions.

It is not until Client A terminates at around time 20 that Client B is able to establish five more sessions and reach the per IP limit of 30 sessions.

3 Summary

The new session functionality in Dante 1.3.2.4 should make it possible to reduce the negative impact against the Dante server from clients that attempt to open too many connections, accidentally or maliciously.

Connection *rate* limits, and limits on both the total *number of sessions* and the number of sessions that can be established from any single IP address can be used to control the resource consumption of the Dante server and should work well in improving the safety in deploying Dante in a environment where it might need to handle connections from a large range of untrusted users.

Feedback to this paper can be sent to *misc-feedback@inet.no*.