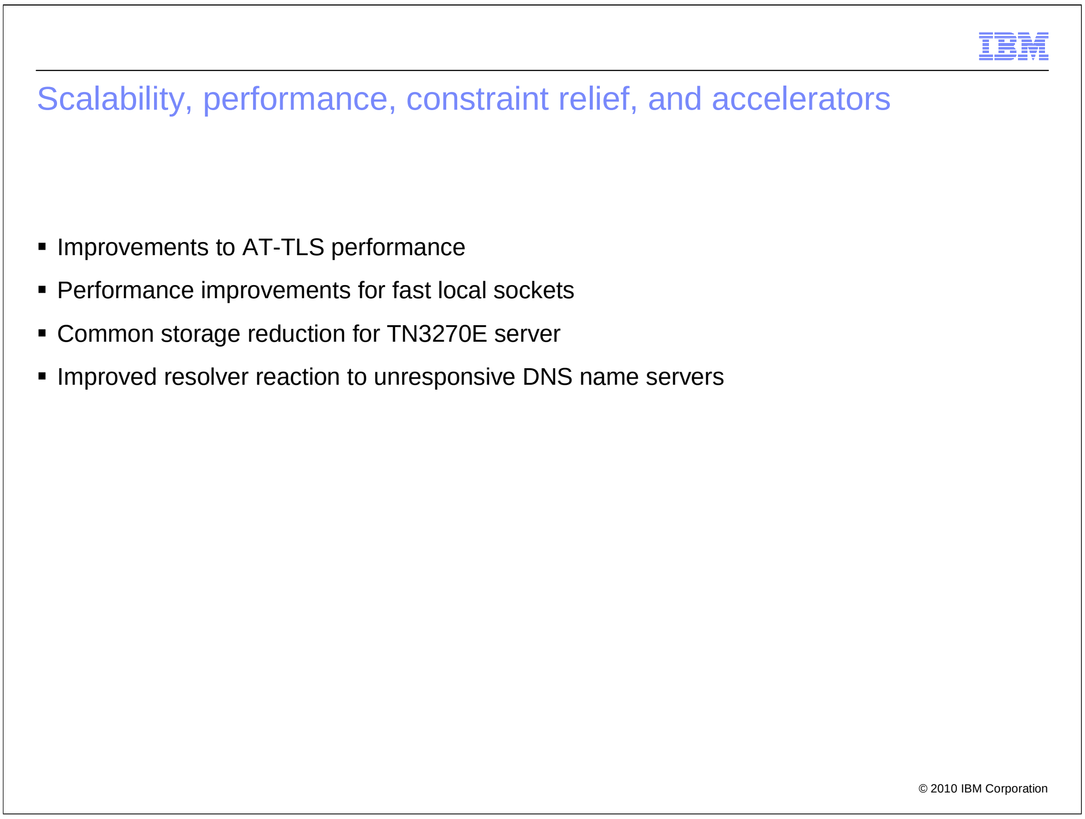
z/OS Communications Server

Performance improvements for AT-TLS, fast local sockets, telnet, and resolver

This presentation describes the performance improvements for AT-TLS, fast local sockets, telnet, and the resolver in z/OS V1R12 Communications Server.



Scalability, performance, constraint relief, and accelerators

- Improvements to AT-TLS performance
- Performance improvements for fast local sockets
- Common storage reduction for TN3270E server
- Improved resolver reaction to unresponsive DNS name servers

AT-TLS performance enhancements have been achieved encrypting and decrypting data.

Performance of existing fast local sockets function was improved by reducing some of the TCP layer processing.

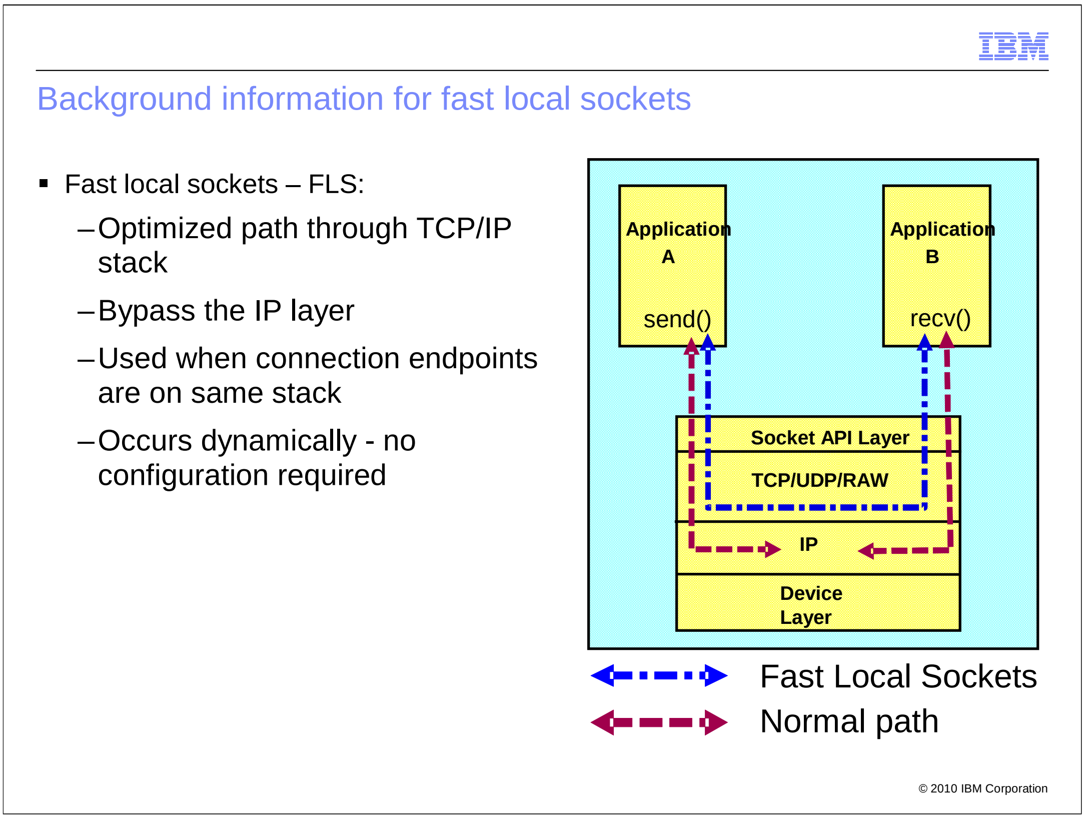
For TN3270E server there is a reduction in the amount of ECSA used by VTAM and the amount of telnet private storage being used. Telnet will now use a single ACB for multiple telnet client sessions.

To assist in diagnosing problems with unresponsive Domain Name System (DNS) servers there will now be a notification to the operator console when a DNS name server does not respond to a query.

In z/OS V1R12 AT-TLS performance enhancements have been achieved encrypting and decrypting data. System SSL has provided a new API which can access data space storage and encrypt/decrypt directly into this storage.

I

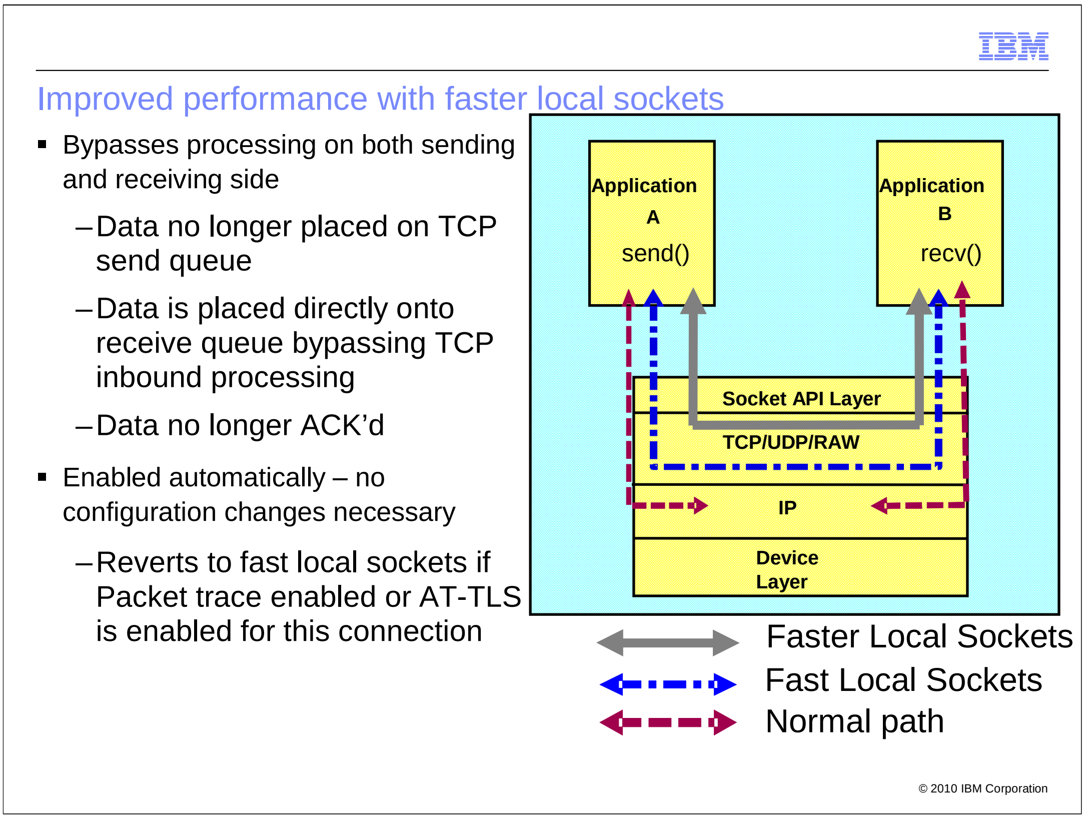
Performance measurements show significant processor reduction for request/response and streaming workloads. Measurements show significant improvement in the number of transactions per second when the processor is fully used (ITR). Request/Response runs improved by about 45% and streaming workloads improved from 25% to 37% (depending on direction of data flow).



Background information for fast local sockets

- Fast local sockets – FLS:
  - –Optimized path through TCP/IP stack
  - –Bypass the IP layer
  - –Used when connection endpoints are on same stack
  - –Occurs dynamically - no configuration required

Fast local sockets (FLS) is an optimized path through the z/OS Communications Server TCP/IP stack when both connection endpoints are on the same stack. It bypasses the IP layer and communicates from sender to receiver by going through the TCP layer only. This occurs automatically and requires no configuration changes.



Improved performance with faster local sockets

- Bypasses processing on both sending and receiving side
  - Data no longer placed on TCP send queue
  - Data is placed directly onto receive queue bypassing TCP inbound processing
  - Data no longer ACK'd
- Enabled automatically – no configuration changes necessary
  - Reverts to fast local sockets if Packet trace enabled or AT-TLS is enabled for this connection

Application A
send()

Application B
recv()

Socket API Layer

TCP/UDP/RAW

IP

Device Layer

Faster Local Sockets

Fast Local Sockets

Normal path

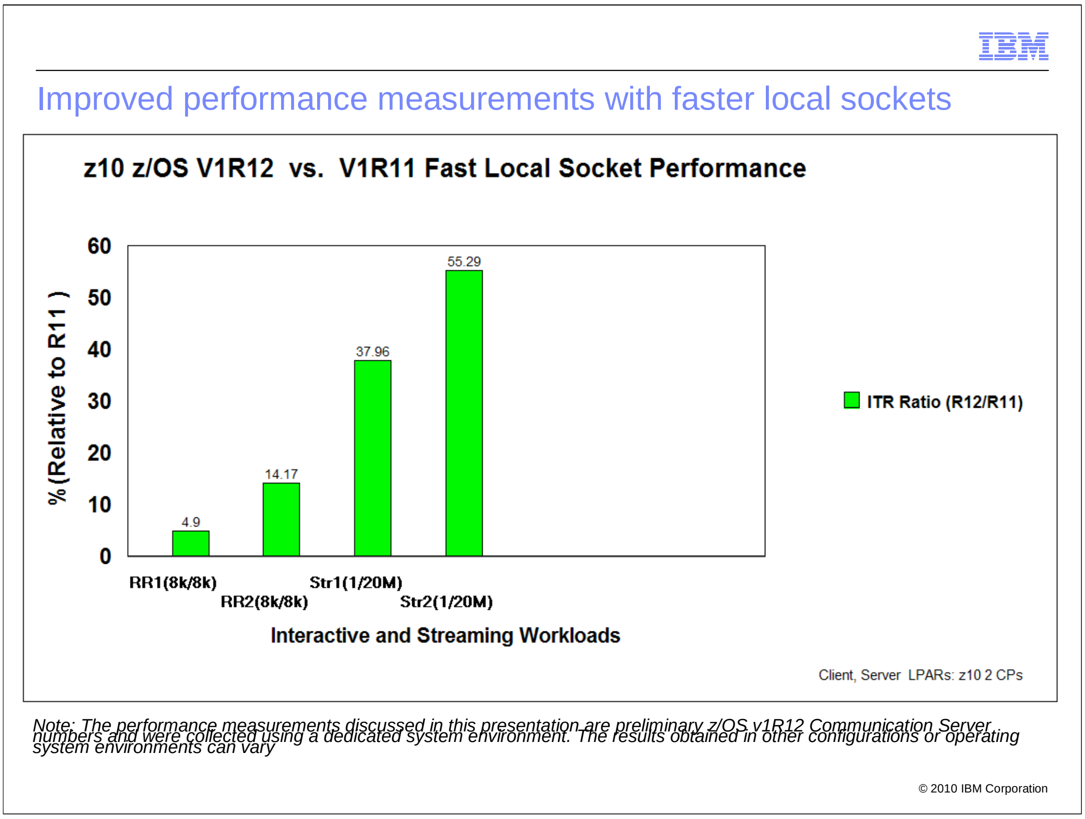
The performance of fast local sockets has improved by reducing some of the TCP layer processing.

Data that is being sent is no longer placed on the sender's queue and then moved to the receive queue.

The data is now moved directly from the sender into the receiver's queue. This bypasses much of the TCP inbound processing.

Acknowledgements of the data are no longer built and sent by the receiver.

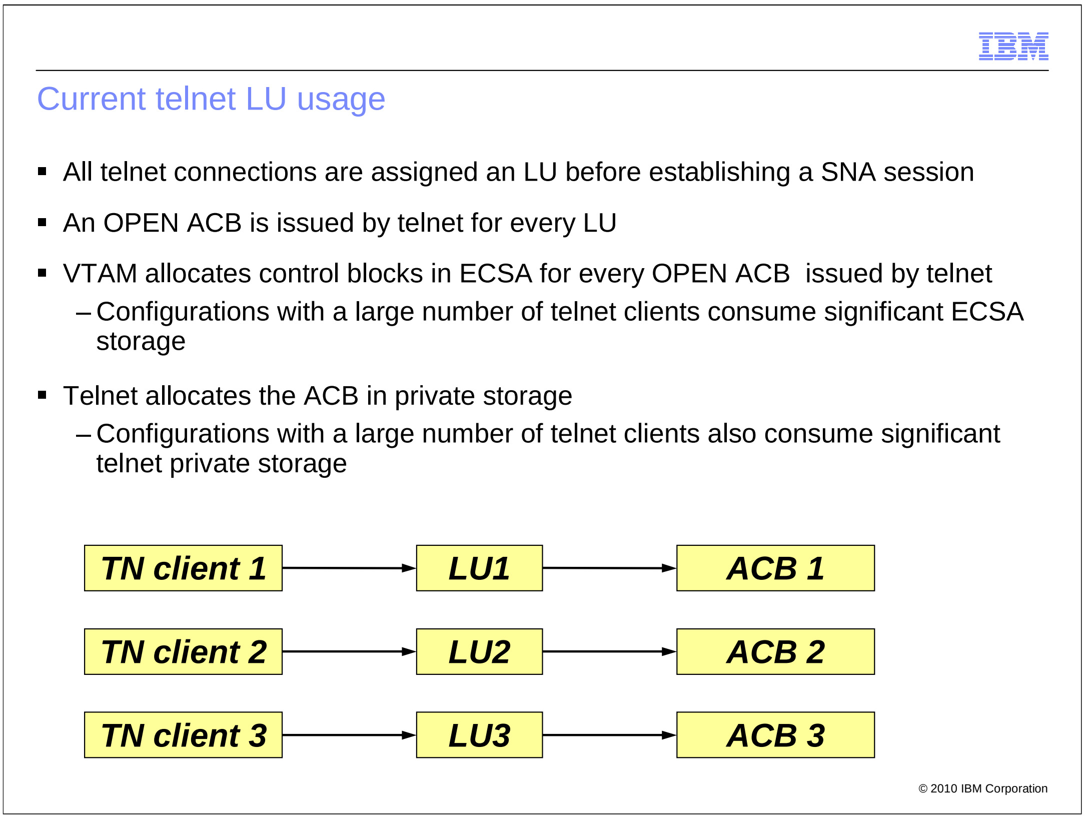
This function is enabled automatically and requires no configuration changes.



Improved performance measurements with faster local sockets

Note: The performance measurements discussed in this presentation are preliminary z/OS v1R12 Communication Server numbers and were collected using a dedicated system environment. The results obtained in other configurations or operating system environments can vary

Early performance measurements show significant improvement in the number of transactions per second when the processor is fully used (ITR). Request/Response runs improved by almost 5% for one connection and by 14% for two connections. For streaming workloads improvements ranged from 38% for one connection to 55% for two connections.



Current telnet LU usage

- All telnet connections are assigned an LU before establishing a SNA session
- An OPEN ACB is issued by telnet for every LU
- VTAM allocates control blocks in ECSA for every OPEN ACB issued by telnet
  - Configurations with a large number of telnet clients consume significant ECSA storage
- Telnet allocates the ACB in private storage
  - Configurations with a large number of telnet clients also consume significant telnet private storage

TN client 1 → LU1 → ACB 1

TN client 2 → LU2 → ACB 2

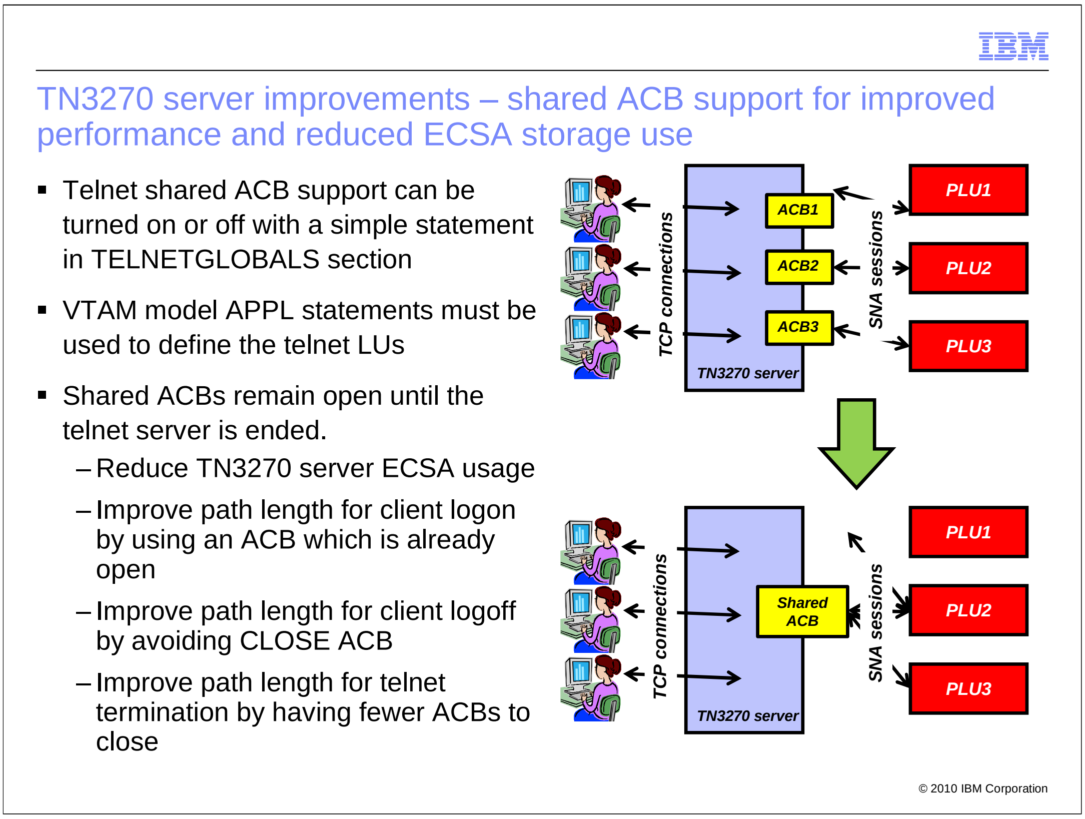
TN client 3 → LU3 → ACB 3

A telnet LU must be represented by a corresponding VTAM definition such as a static APPL statement or a VTAM model APPL statement. To establish a session with a target host application, telnet first issues an OPEN ACB for each assigned LU name, followed by a SETLOGON RPL and REQSESS RPL.

This picture shows you have an OPEN ACB for each telnet LU name that represents a client.

This enhancement for telnet addresses problems related to excessive storage usage by large telnet configurations.

ECSA based control blocks are allocated by VTAM for every OPEN ACB issued by telnet and the ACB resides in telnet private storage.



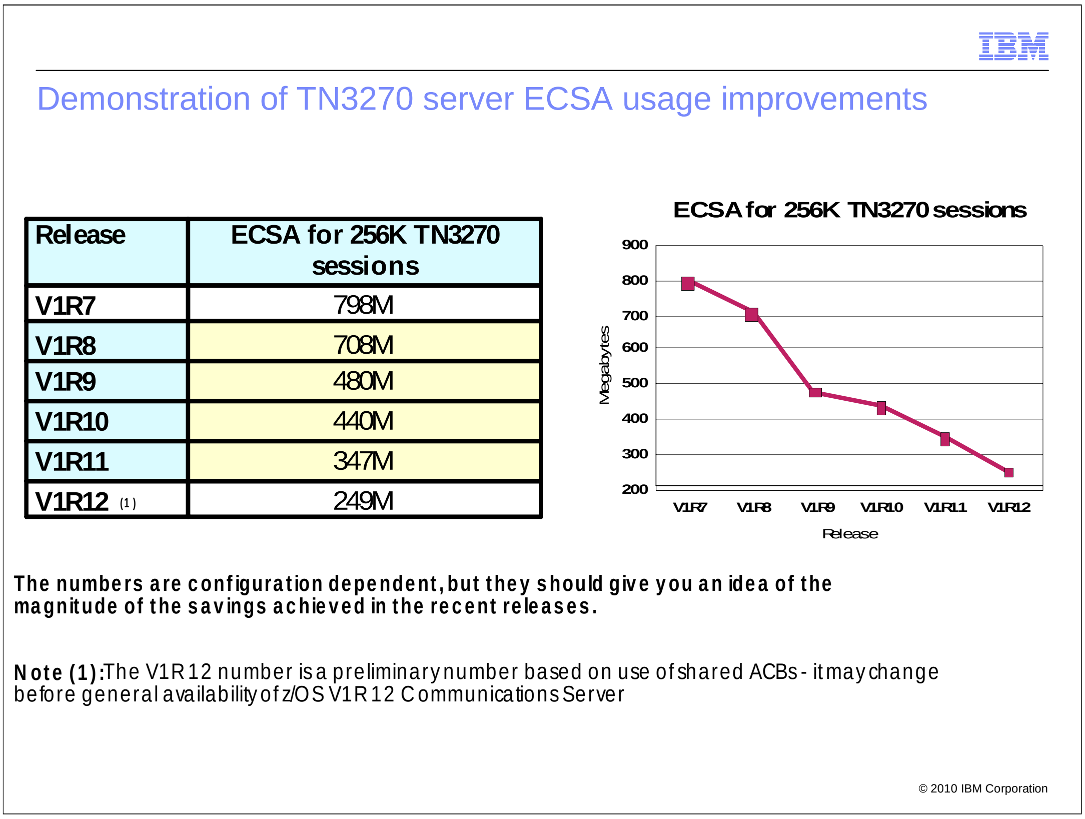
TN3270 server improvements – shared ACB support for improved performance and reduced ECSA storage use

- Telnet shared ACB support can be turned on or off with a simple statement in TELNETGLOBALS section
- VTAM model APPL statements must be used to define the telnet LUs
- Shared ACBs remain open until the telnet server is ended.
  - Reduce TN3270 server ECSA usage
  - Improve path length for client logon by using an ACB which is already open
  - Improve path length for client logoff by avoiding CLOSE ACB
  - Improve path length for telnet termination by having fewer ACBs to close

To reduce the amount of ECSA used by VTAM and the amount of telnet private storage used by telnet, Telnet will now use a single ACB for multiple telnet client sessions. Four shared ACBs are initially opened for each telnet port. Each ACB can be assigned 300 clients. New shared ACBs are opened as the client workload increases for a given port. The client workload per ACB increases from 300 to a maximum value of 1100 clients per ACB.

In addition to saving storage, using shared ACBs has an additional benefit. Path length is improved for logon, logoff, and termination processing because telnet does not have to issue OPEN ACB or CLOSE ACB for every session.

A new statement is provided to enable the function. The default is NOSHAREACB, which disables the new support. SHAREACB/NOSHAREACB statement can be coded only in the TELNETGLOBALS statement block.



Demonstration of TN3270 server ECSA usage improvements

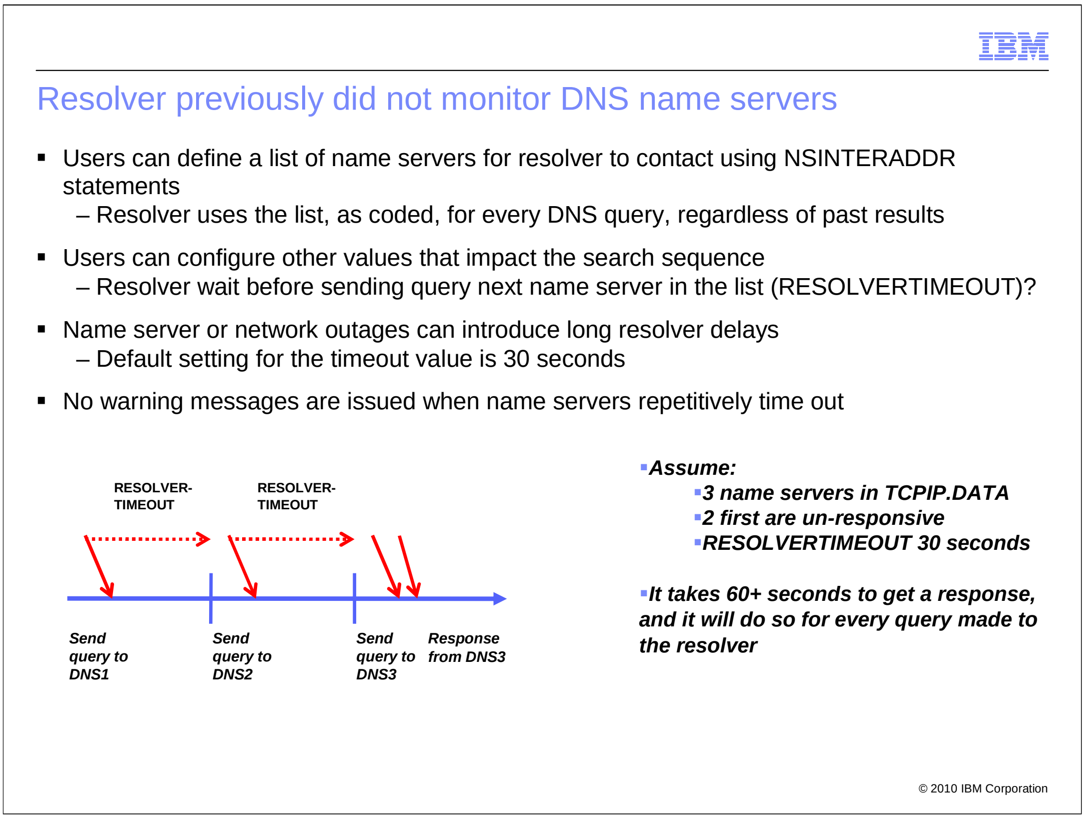
Release	ECSA for 256K TN3270 sessions
V1R7	798M
V1R8	708M
V1R9	480M
V1R10	440M
V1R11	347M
V1R12 (1)	249M

ECSA for 256K TN3270 sessions

The numbers are configuration dependent, but they should give you an idea of the magnitude of the savings achieved in the recent releases.

Note (1): The V1R12 number is a preliminary number based on use of shared ACBs - it may change before general availability of z/OS V1R12 Communications Server

You can see that this line item reduced ECSA usage by almost 100M for a system with 256 thousand telnet sessions. There will also be about 20M in telnet private savings for this configuration as well.



Resolver previously did not monitor DNS name servers

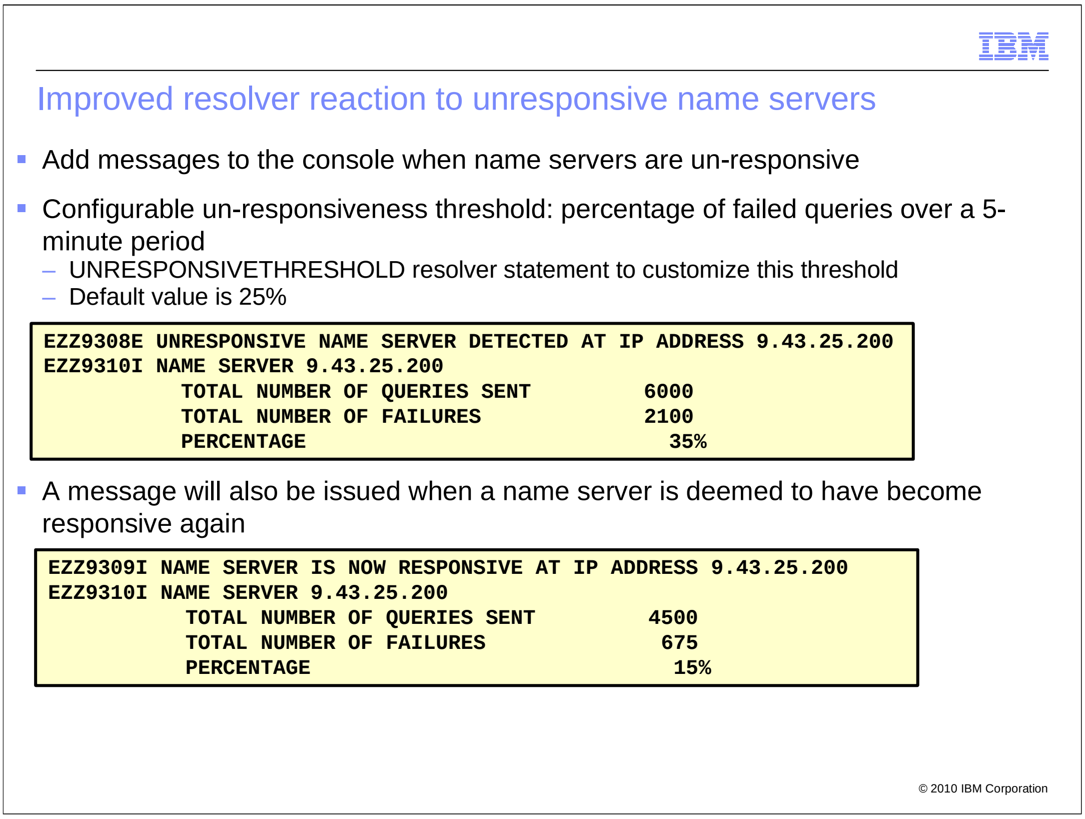
- Users can define a list of name servers for resolver to contact using NSINTERADDR statements
  - – Resolver uses the list, as coded, for every DNS query, regardless of past results
- Users can configure other values that impact the search sequence
  - – Resolver wait before sending query next name server in the list (RESOLVERTIMEOUT)?
- Name server or network outages can introduce long resolver delays
  - – Default setting for the timeout value is 30 seconds
- No warning messages are issued when name servers repetitively time out

- ***Assume:***
  - ***3 name servers in TCPIP.DATA***
  - ***2 first are un-responsive***
  - ***RESOLVERTIMEOUT 30 seconds***
- ***It takes 60+ seconds to get a response, and it will do so for every query made to the resolver***

A major part of resolver processing is sending queries to Domain Name System (DNS) name servers. The resolver learns which name servers to contact from the IP addresses coded on the NSINTERADDR statements in the TCPIP.DATA configuration file. This list of IP addresses is used by resolver on every query that requires communication with name servers, even if previous attempts to contact a specific name server were unsuccessful.

In addition to the list of name servers, users have several other configuration options that can influence resolver interactions with the name servers. For this discussion, two of the more important ones are RESOLVERUDPRETRIES and RESOLVERTIMEOUT. The former option indicates whether resolver should contact the list of name servers multiple times, per query, or just a single time. The latter option specifies the amount of time the resolver should wait for a response from a name server before moving on to the next name server in the list.

The combination of configuration options with resolver's continuing to use name servers even after unsuccessful queries can result in situations where long delays occur during name resolution. In this example, three name servers have been configured on the NSINTERADDR statement. Resolver will send the request for host.raleigh.ibm.com to the first DNS name server. No value for RESOLVERTIMEOUT was coded, so a default setting of 30 seconds is used. That means resolver waits for 30 seconds for the name server at 10.1.1.1 to respond before moving to the next name server in the list. If the first two name servers are not responding, it takes a full minute for resolver to send the request to the third name server in the list. This one-minute delay applies to every request sent by resolver, as long as the first two name servers are not responding. Identifying situations where resolver is the bottleneck for transaction delays, and why it is the bottleneck, has proven to be a challenge for users.



Improved resolver reaction to unresponsive name servers

- Add messages to the console when name servers are un-responsive
- Configurable un-responsiveness threshold: percentage of failed queries over a 5-minute period
  - UNRESPONSIVETHRESHOLD resolver statement to customize this threshold
  - Default value is 25%

```
EZZ9308E UNRESPONSIVE NAME SERVER DETECTED AT IP ADDRESS 9.43.25.200
EZZ9310I NAME SERVER 9.43.25.200
           TOTAL NUMBER OF QUERIES SENT         6000
           TOTAL NUMBER OF FAILURES             2100
           PERCENTAGE                             35%
```

- A message will also be issued when a name server is deemed to have become responsive again

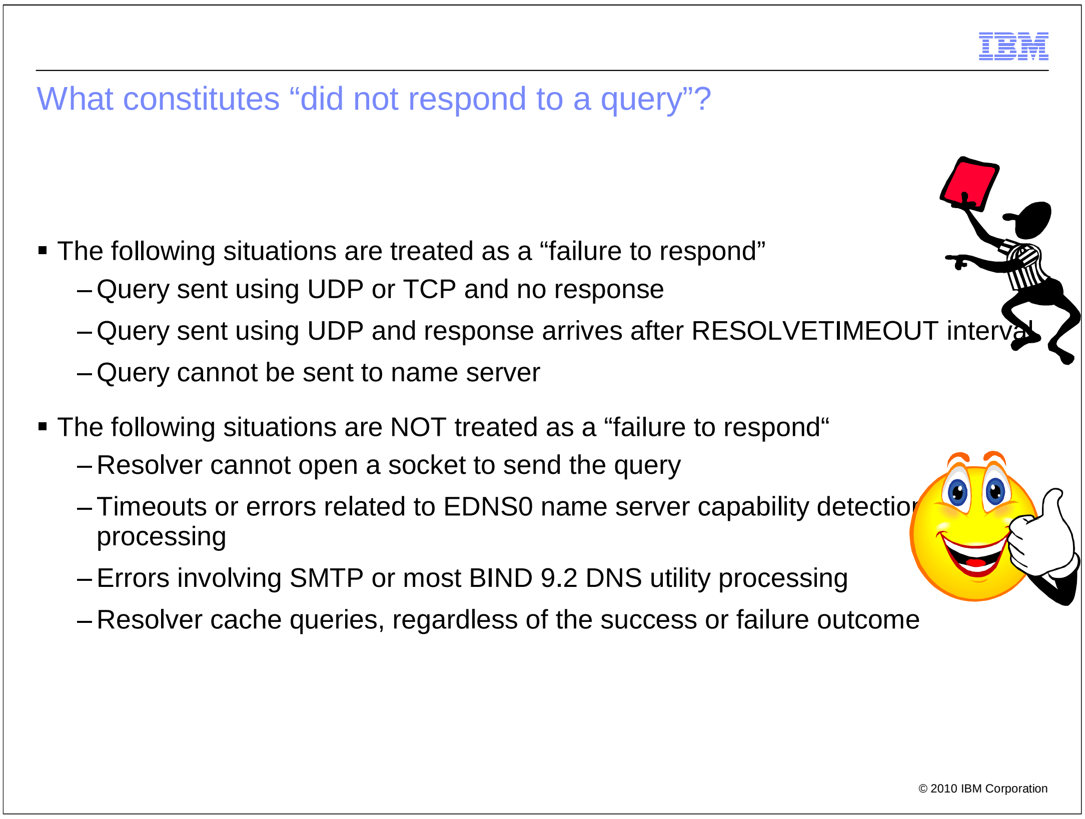
```
EZZ9309I NAME SERVER IS NOW RESPONSIVE AT IP ADDRESS 9.43.25.200
EZZ9310I NAME SERVER 9.43.25.200
           TOTAL NUMBER OF QUERIES SENT         4500
           TOTAL NUMBER OF FAILURES              675
           PERCENTAGE                             15%
```

You can specify the UNRESPONSIVETHRESHOLD resolver setup statement to customize the threshold value when resolver declares a name server to be unresponsive. By default, resolver starts with a threshold setting of 25%. This means that if a name server fails to respond to more than 25% of the queries sent to it over a five-minute period, resolver will declare the name server to be unresponsive. The threshold applies to all name servers in the network, and either all name servers are monitored, or no name servers are monitored.

Because the value represents a percentage, any integer between 0 and 100 can be specified for UNRESPONSIVETHRESHOLD. If you code UNRESPONSIVETHRESHOLD(0), that disables the monitoring function, and resolver maintains no statistics for any name servers in this situation. You can dynamically modify the setting for UNRESPONSIVETHRESHOLD using the MODIFY RESOLVER command. If you specify UNRESPONSIVETHRESHOLD(0), the function is disabled immediately by the MODIFY command. If you specify a non-zero value, the percentage is used by resolver for comparison purposes at the next one-minute checkpoint.

The resolver recalculates the failure rate for each name server every 60 seconds. If a name server had been unresponsive, and the failure rate drops below the threshold setting during this checkpoint, resolver now declares the name server to be responsive. The eventual action message is cleared, and messages EZZ9309I and EZZ9310I are displayed to indicate the name server unresponsiveness condition has ended. These messages, shown on the slide, are intended primarily for automation purposes, and provide an easier method of definitively pointing out when the problem with the name server was alleviated. You can use the information in message EZZ9310I to determine whether the name server is completely responsive again or is hovering at the threshold setting for your environment.

Just as there was no minimum number of requests needed to declare a name server to be unresponsive, there is no minimum number required over five minutes to declare the name server to be responsive. Thus, if absolutely no requests were sent to the name server over five minutes, that causes resolver to view the name server as responsive for that interval.



What constitutes “did not respond to a query”?

- The following situations are treated as a “failure to respond”
  - Query sent using UDP or TCP and no response
  - Query sent using UDP and response arrives after RESOLVETIMEOUT interval
  - Query cannot be sent to name server
- The following situations are NOT treated as a “failure to respond“
  - Resolver cannot open a socket to send the query
  - Timeouts or errors related to EDNS0 name server capability detection processing
  - Errors involving SMTP or most BIND 9.2 DNS utility processing
  - Resolver cache queries, regardless of the success or failure outcome

Resolver does not count every failure encountered when communicating with a name server as a failure to respond on the part of the name server. This slide lists the conditions that are counted as failures. The most obvious error is that the resolver has successfully sent a request to the name server, and no response was ever received. A variation of this error is the situation where the response from the name server arrives late, even if resolver ends up using the response to satisfy the request. In this instance, the name server is not responding within the configured UDP time out value. This results in extra work having to be done by resolver to satisfy the request, so that is considered in the responsiveness formula.

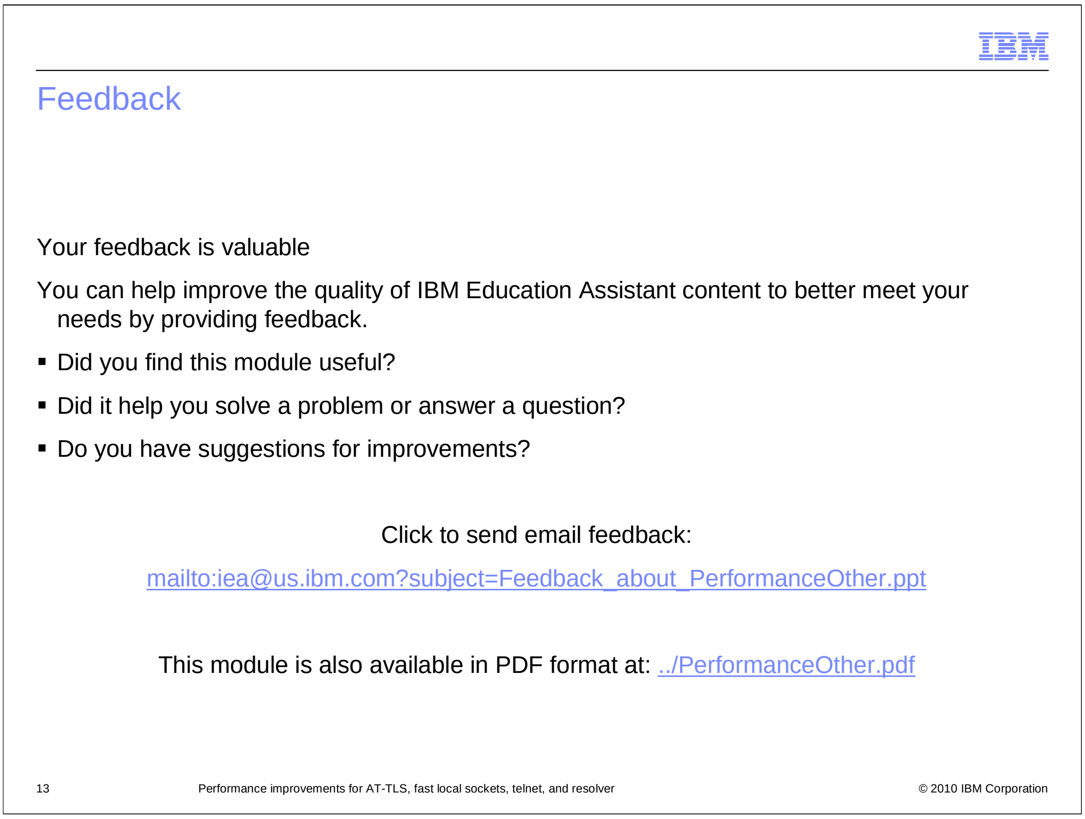
Two other errors that are counted involve the inability of resolver to deliver the request to the name server at all. One involves the inability to send UDP requests to the target name server, and the second involves the failure to establish a TCP connection to the target name server. Both situations suggest either network problems or the specification of an incorrect name server IP address, so both were considered important for identifying name server responsiveness.

Some conditions are not included in calculations for name server responsiveness. The first condition is a situation where resolver cannot open a socket to the TCP/IP stack when trying to send the request to a name server. This typically represents a condition where the stack is not active, which gives no input on the condition of the name server.

As part of determining a name server’s EDNS0 capabilities, resolver might periodically send additional requests as part of name resolution. These requests are always triggered by a truncated UDP response from the name server, which suggests that the name server had to be responsive. If the name server fails to respond to this EDNS0 probe, that most likely means that the name server cannot support EDNS0 as opposed to being unable to respond successfully. Since this traffic is tailored just for resolver processing, it is not included in the calculations of name server responsiveness.

SMTP processing does not use the system resolver, so any failures during SMTP processing are not counted. Likewise, the BIND 9.2 DNS utilities primarily uses its own resolver, so requests using this separate resolver are not included in the calculations either.

Finally, no resolver cache queries are counted in the statistics. This includes both failures and successes when accessing the cache, although failures to get information from the cache typically result in queries being sent to name servers, which can result in monitoring failures.



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

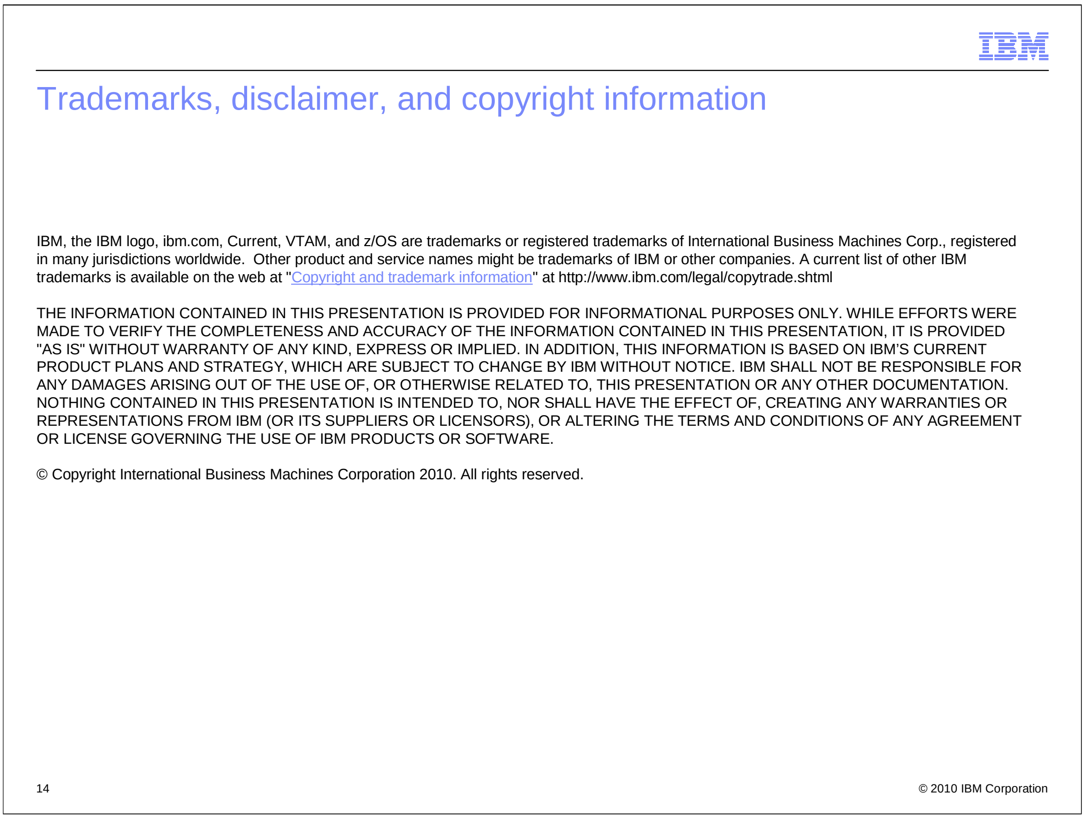
- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_PerformanceOther.ppt

This module is also available in PDF format at: ../PerformanceOther.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.



Trademarks, disclaimer, and copyright information

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.