z/TPF Communications and Security Enhancements

2022 TPF Users Group Conference
March 27-30, Dallas, TX
Communications Subcommittee

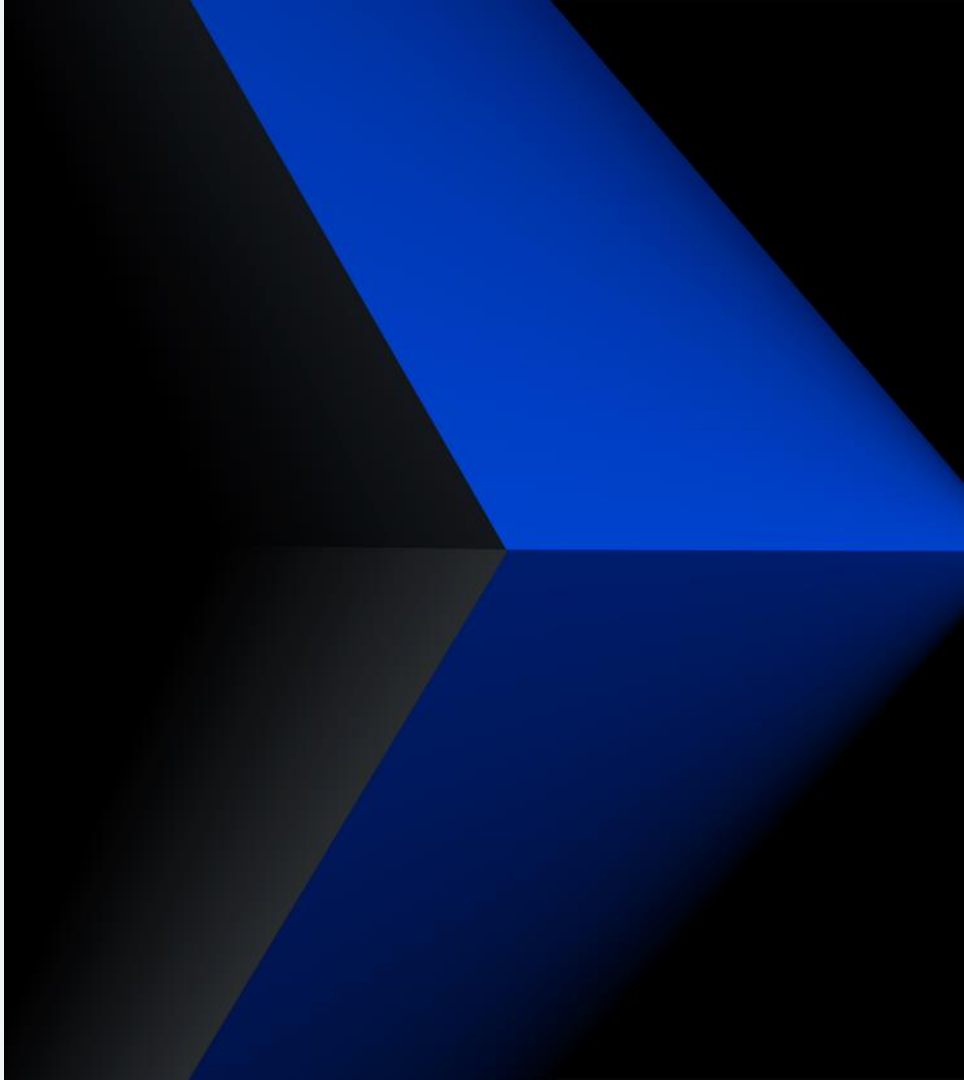
Jamie Farmer

IBM Z

IBM

Disclaimer

Any reference to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.



Agenda

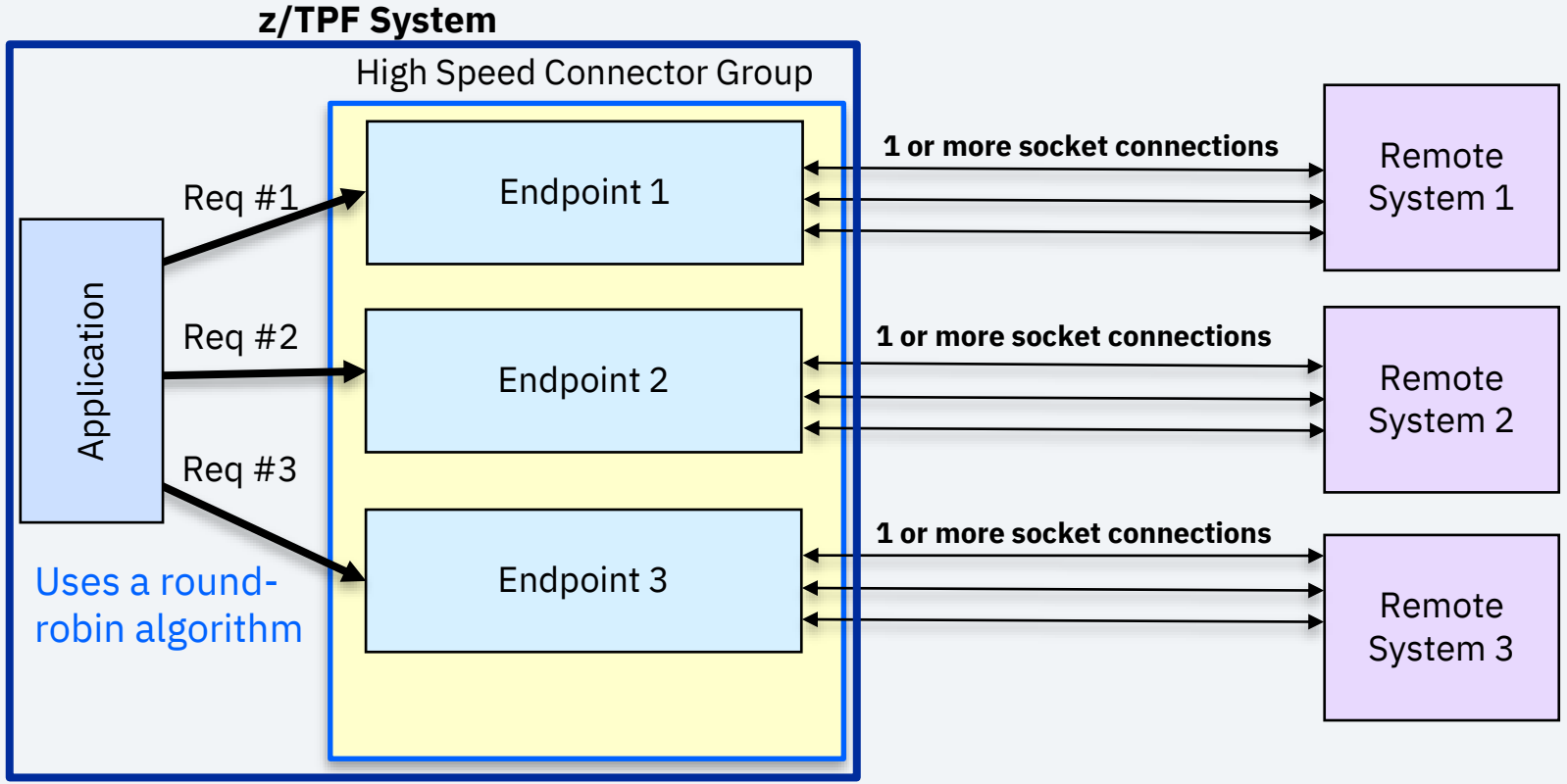
- Improved throughout of high-speed connector when endpoint groups are used as a one-way pipe outbound to remote systems.
- Improvements to allow z/TPF clients to interface with cloud-based systems that are hosting multiple hosts.
- Improve the security of the z/TPF system by having better control over which cipher algorithms are used by z/TPF servers.
- Improve the security of the z/TPF keystore by enhancing the cryptographic algorithms that are used to secure the keys.

New Socket Selection Algorithm for High-Speed Connector

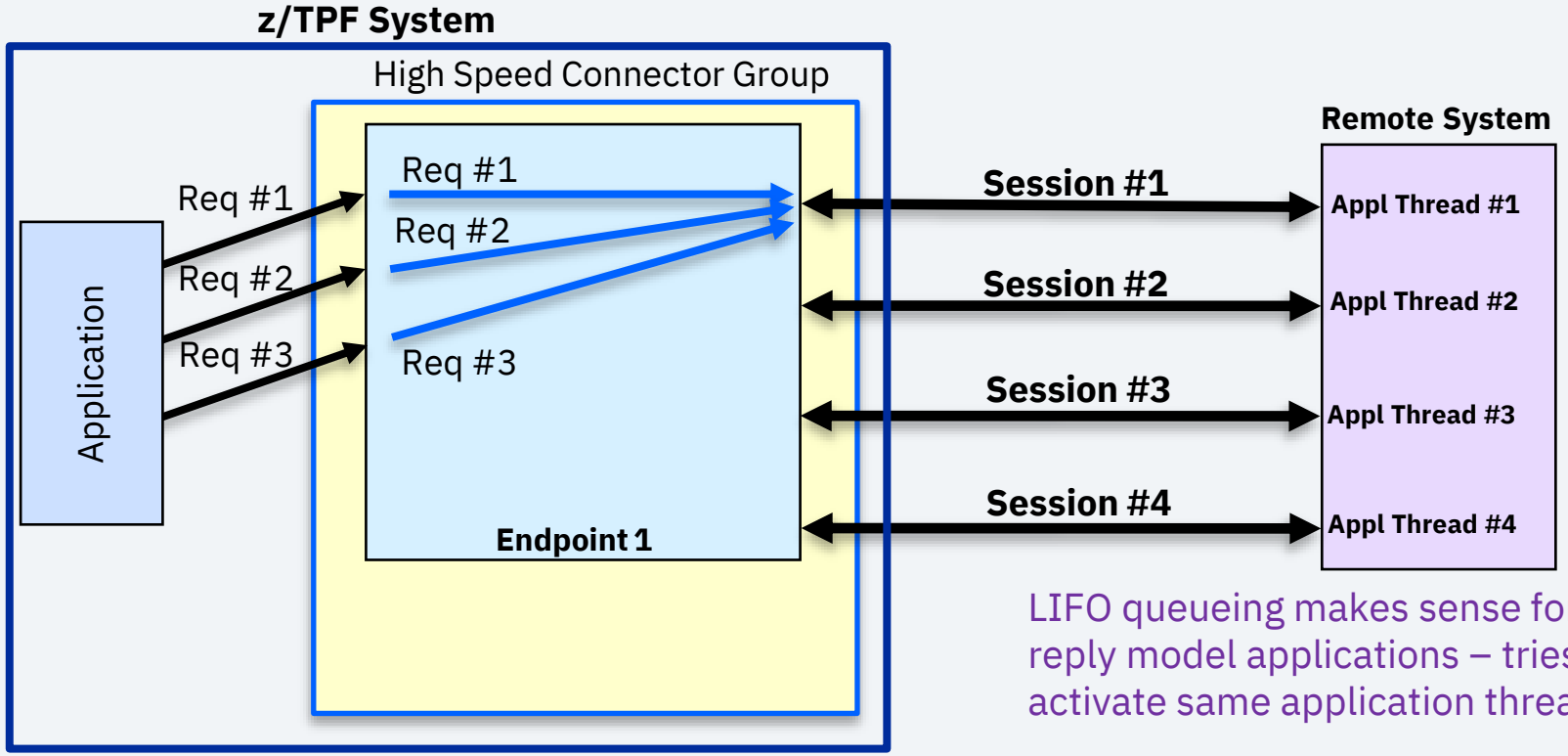
Problem Statement

Using High-Speed Connector as a one-way pipe can overload individual sockets of an endpoint while other sockets for the endpoint remain idle.

As-Is : Endpoint Selection in High-Speed Connector



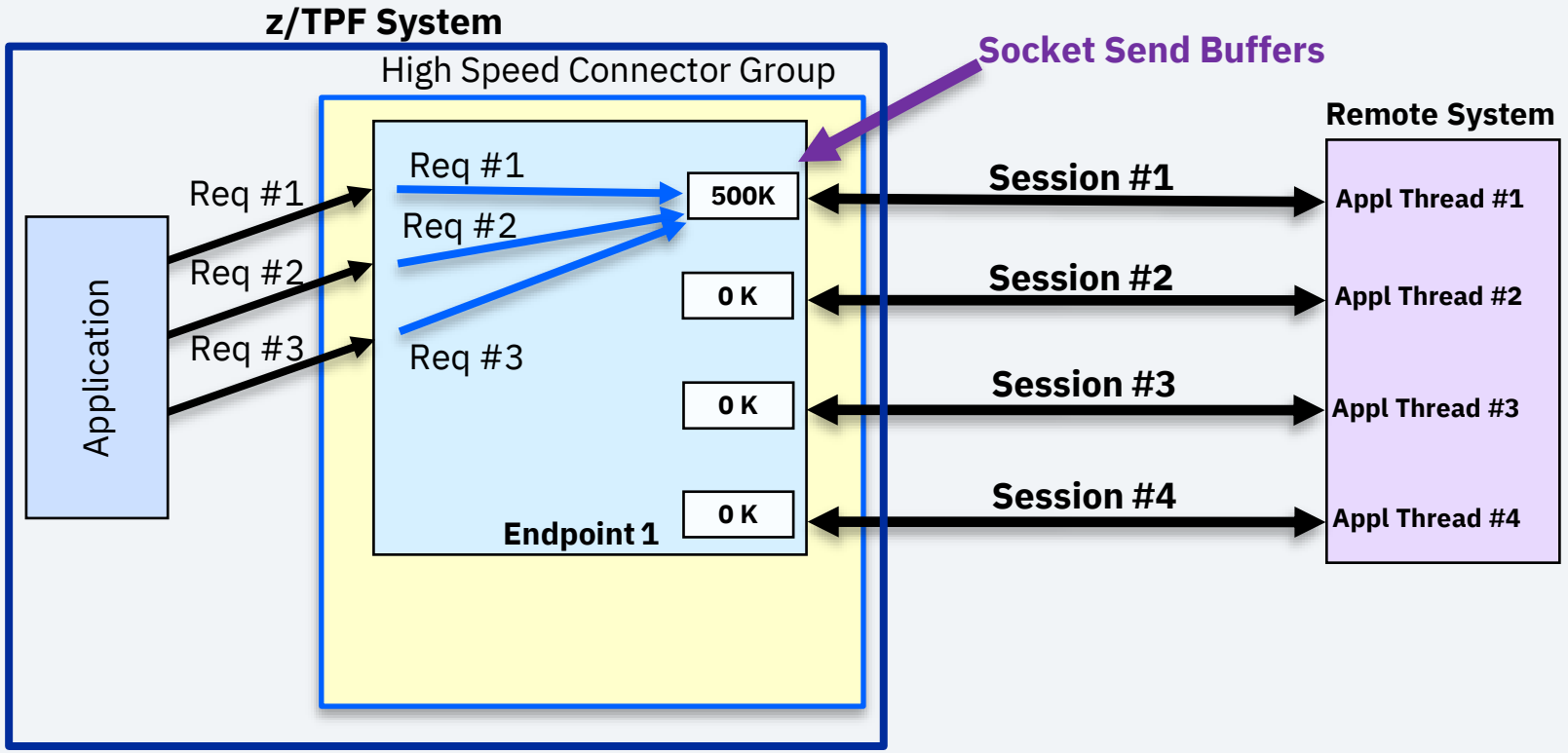
As-Is : Endpoint's Socket Selection in High-Speed Connector



LIFO queueing makes sense for request-reply model applications – tries to always activate same application thread.

Socket Selection of an endpoint uses LIFO queueing. Always try to reuse the same session.

As-Is : Socket Selection Algorithm for LIFO Sockets



For 1-way piped applications, LIFO queueing can cause overload of a single socket inhibiting throughput to the remote system!!

To-Be : Round-Robin Sockets Within an Endpoint

A new endpoint group descriptor (ept.xml) file option has been added called <SocketSelect>

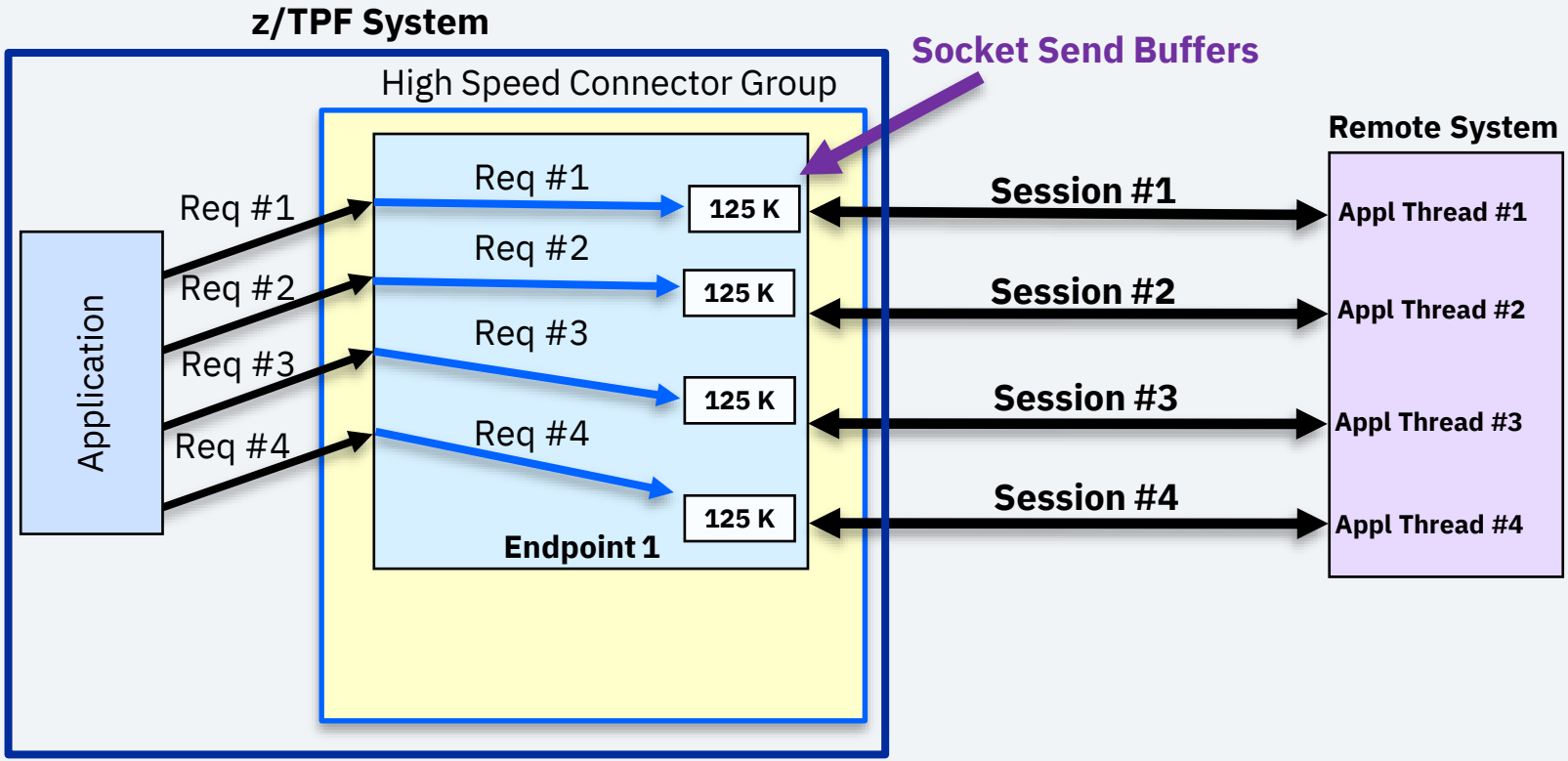
- Default SocketSelect value is LIFO socket selection which is how the high-speed connector behaves today.
- User can indicate FIFO socket selection which will force round-robin of sockets within an endpoint
- The value can be modified on the fly by loading a new endpoint group descriptor file and the change will take effect immediately
- Group definition – will take affect for all endpoints within the group.

To-Be : Round-Robin Sockets (FIFO Queueing) Example

```
<tns:groupName>loopnres</tns:groupName>
      <tns:TLS>YES</tns:TLS>
      <tns:qMaxDepth>20</tns:qMaxDepth>
      <tns:qThreshold>16</tns:qThreshold>
      <tns:syncTimeout>3000</tns:syncTimeout>
      <tns:heartbeatInterval>60</tns:heartbeatInterval>
      <tns:SocketSelect>FIFO</tns:SocketSelect>
</tns:EndpointGroup>
```

If SocketSelect is not specified, the LIFO algorithm is used.

To-Be : Socket Selection Algorithm for FIFO Sockets



FIFO SocketSelect option spreads traffic across all available sockets!

To-Be : ZCONN Command Display Updates

The ZCONN CONFIG command display was updated to reflect the Socket Selection algorithm defined for this group.

```
User:   ZCONN CONFIG G-linresp

  DESCRIPTION-Linux endpoint group
  NAME- linresp     TYPE-USER     TLS-N
  QUEUE DEPTH-   20          QUEUE THRESHOLD-   16
  HEARTBEAT INTERVAL-   60  SYNC TIMEOUT- 10000
  MAX ASYNC DATA-  4096     MAX ASYNC MESSAGE- 4096
  ENDPOINT SESSION WARNING INTERVAL-   1
  GROUP SESSION WARNING INTERVAL-   1
  GROUP QUEUE WARNING INTERVAL-   1
  LENOFFSET-NONE   LENFIELDLEN-NONE  HEADERLEN-NONE
  PRIORITY-NORMAL  CONNECT TIMEOUT-DEFAULT
  SOCKETSELECT-FIFO
  ALIAS HOSTNAMES:  NONE
             . . .
```

The ZCONN TOPOLOGY was updated with a new diagnostic to indicate if/when a socket send buffer full condition occurs.

```
User:   ZCONN TOPOLOGY G-linresp

                  . . .

 ENDPOINT-loop8601     ROLE-PRIMARY    STATUS-ACTIVE
   REMOTE IP-127.0.0.1     REMOTE PORT-8601
   REMOTE HOST-loopback.pok.ibm.com
   START SESSIONS-       0  MAX SESSIONS-              4
   IN USE SESSIONS-      0  MAX IN USE SESSIONS-       1
   ACTIVE SESSIONS-      2  SESSIONS STARTING-         0
   NUM CONNECTS-         2  NUM CONNECT FAILS-         0
   LAST EPT TIMEOUT TOD-NONE
      LAST EPT TIMEOUT PORT-NONE
   LAST CONNECT ERROR TOD-NONE
      LAST CONNECT ERROR REASON-NONE
   LAST SEND BLOCKED TOD-D9DCDEAD673EC178 (2021-06-14 14:00:31)
      LAST SEND BLOCKED PORT-49163   SEND BLOCKED MSG SIZE-1032
```

May be an indication of a throughput issue when applications are not expecting a reply (one-way pipe)

Value Statement

Having a round-robin socket selection algorithm option can provide better throughput through high-speed connector as well provide usability and diagnostic improvements.

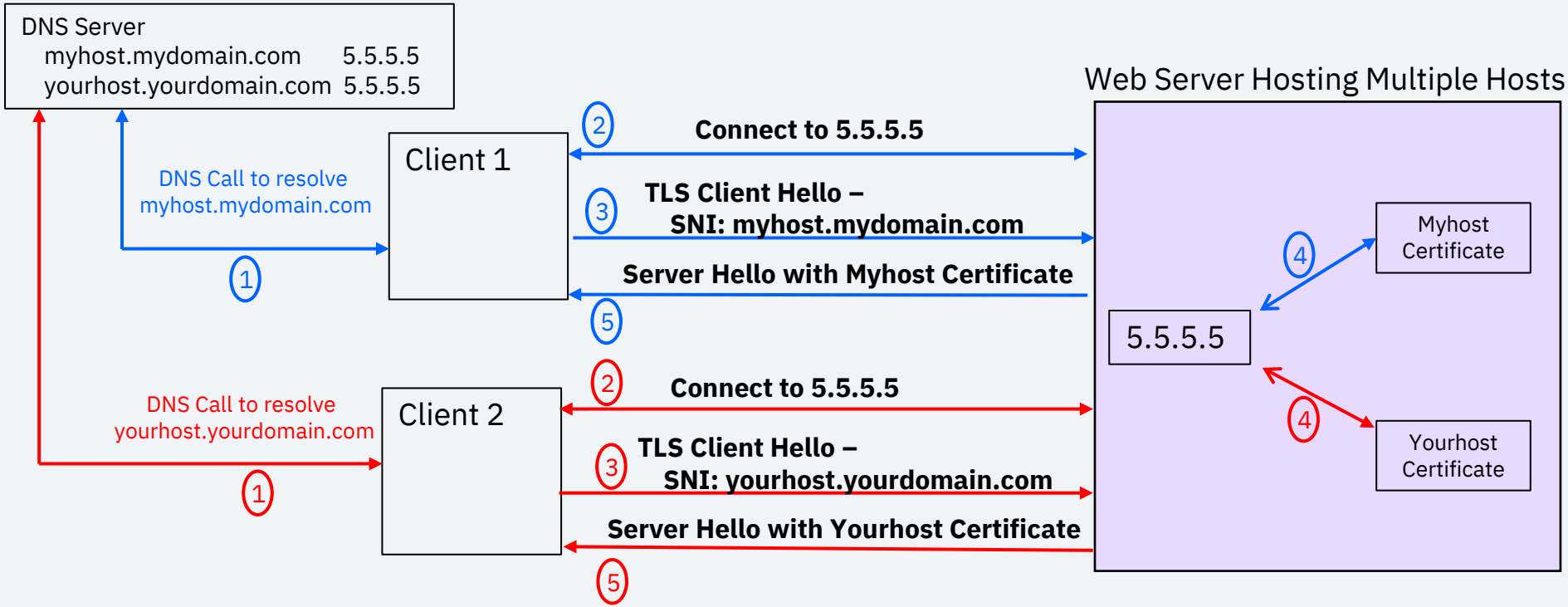
Delivered with PJ46544 (Aug 2021)

Server Name Indicator for z/TPF TLS Clients

Background

- The Server Name Indicator (SNI) Extension allows a TLS server to host multiple server certificates all under a single IP address.
- For the server to determine which TLS certificate to use, the client adds the SNI extension to the TLS Client Hello which indicates the hostname / domain.
- The server can use this information to return the corresponding certificate based on the name passed by the client

Server Name Indicator Extension Example



Problem Statement

- Some cloud-based remote servers requires the SNI extension to be specified in the TLS Client Hello.

- Support for TLS clients to set the Server Name Indicator does not exist for z/TPF shared SSL.

To-Be : Setting and Getting the Server Name Indicator

int SSL_set_tlsext_host_name(const SSL *s, const char *name)

s is a client SSL object

name must be a valid hostname of a maximum of 255 characters

* The function must be called **before** the SSL_connect() to set SNI properly

const char *SSL_get_servername(const SSL *s, const int type)

s can be either a client or server SSL object

type SSL_CTRL_SET_TLSEXT_HOSTNAME will return the hostname

* The function can be called by a TLS client or server to determine what SNI hostname was set by the client.

To-Be : SNI Middleware Changes

- High-speed connector and the enhanced HTTP client will automatically set the TLS SNI extension when a hostname is specified for connect.

High Speed Connector Endpoint Definition

```
<!-- ============================================================ -->
<!-- Definition of an example primary endpoint                    -->
<!-- ============================================================ -->

<tns:Endpoint>
    <tns:endpointName>lresps11</tns:endpointName>
    <tns:role>PRIMARY</tns:role>
    <tns:destination>examplehost.tst.com:50111</tns:destination>
    <tns:startSocket>1</tns:startSocket>
    <tns:maxSocket>7</tns:maxSocket>
    <tns:bufferSendSize>1048576</tns:bufferSendSize>
    <tns:bufferReceiveSize>1048576</tns:bufferReceiveSize>
</tns:Endpoint>

<!-- ============================================================ -->
<!-- Definition of an example HSC group                           -->
<!-- ============================================================ -->


<tns:groupName>groupexp</tns:groupName>
<tns:TLS>YES</tns:TLS>
<tns:groupDescription>Example host group </tns:groupDescription>
<tns:gMaxDepth>20</tns:gMaxDepth>
<tns:gThreshold>16</tns:gThreshold>
<tns:syncTimeout>10000</tns:syncTimeout>
<tns:heartbeatInterval>60</tns:heartbeatInterval>
```

Wireshark Trace of Client Hello sent by z/TPF

```
Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 124
    Version: TLS 1.2 (0x0303)
    Random: df0b184766573e494c3f59adad153f6f91841b3fa516f25bed37de52769073b3
    Session ID Length: 0
    Cipher Suites Length: 4
    Cipher Suites (2 suites)
    Compression Methods Length: 1
    Compression Methods (1 method)
    Extensions Length: 79
    Extension: server_name (len=25)
      Type: server_name (0)
      Length: 25
      Server Name Indication extension
        Server Name list length: 23
        Server Name Type: host_name (0)
        Server Name length: 20
        Server Name: examplehost.tst.com
    Extension: session_ticket (len=0)
    Extension: encrypt_then_mac (len=0)
    Extension: extended_master_secret (len=0)
      ....
```

Value Statement

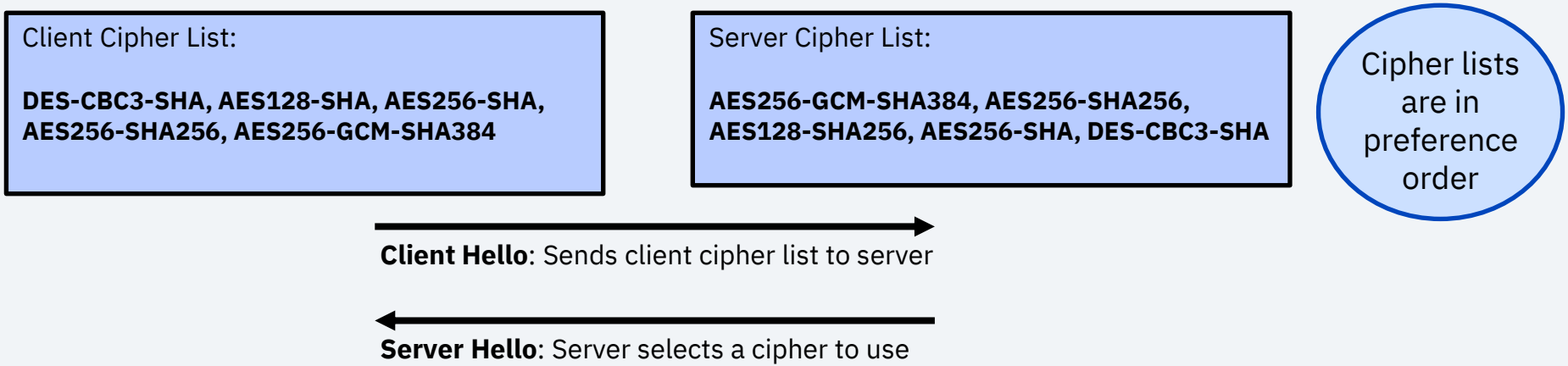
With SNI extension support for z/TPF clients, the z/TPF system can communicate with secure cloud-based servers that are hosting multiple server certificates.

Delivered with PJ46661 (Jan 2022)

z/TPF Server TLS Cipher Preference

Background: TLS Cipher Selection

- The cipher selected for a TLS session is negotiated between the client and the server
  - Ultimately, the server selects the cipher algorithm to use



- By default, in OpenSSL, the server will honor the preference the client specified!
- **In the above example, the DES-CBC3-SHA cipher is used, even though the server prefers AES256-GCM-SHA384 and AES256-GCM-SHA384 is a stronger cipher algorithm!**

Problem Statement

With the cipher preference controlled by the client by default, the server applications on z/TPF do not have total control of the cipher that is selected for use.

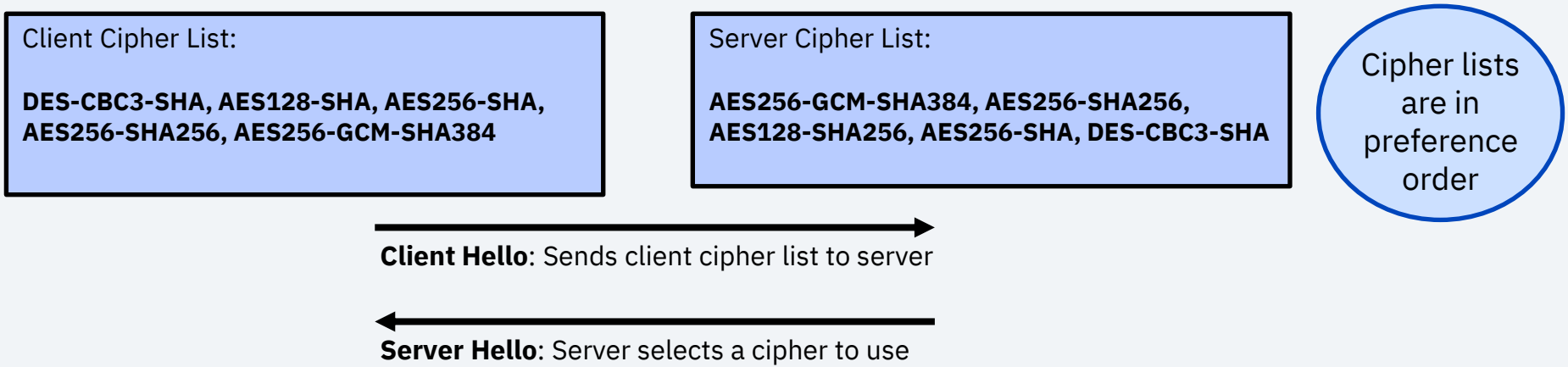
There is an option in OpenSSL to allow for the server preference, but it’s difficult to change all applications and middleware to take advantage of this.

To Be: System Setting to Use Server Cipher Preference

- A new system-wide control has been created to have all TLS servers in z/TPF use the server's cipher preference.
- New CTK2 variable called SSLSERVP
  - YES indicates to use the z/TPF TLS server's cipher preference
  - NO is the default
- Can be altered online and changes take effect immediately
  - ZNKEY SSLSERVP-YES/NO

To Be: Example with SSLSERVP-YES

- With the global override enabled (SSLSERVP-YES) the server's cipher list defines the preferred cipher to use.



In this case the server's first matching cipher with the client, AES256-GCM-SHA384 is selected!

Considerations for Switching to Server Cipher Preference

- Should not affect connectivity because there is a matching TLS cipher between client and server today
- You should verify the cipher lists you defined for your z/TPF TLS servers are prioritized correctly.
- When changing this value using ZNKEY, ensure the value is also updated in your local version of ctk2.asm
  - Don’t want the setting to be cleared if a new CTK2 is loaded.

Value Statement

With the global server cipher preference users have more control as to which ciphers are selected for TLS sessions with z/TPF TLS servers.

Delivered with PJ46661 (Jan 2022)

Improving the Security of the z/TPF Keystore

Background

- The z/TPF keystore is used to securely create and manage keys used in the z/TPF complex
  - Users can create and manage symmetric keys (such as AES128 or AES256) using the z/TPF symmetric keystore
  - Users can create and manage public and private key pairs using the z/TPF PKI keystore
- These keys are secured in both memory and on disk using a variety of cryptographic techniques (encryption, message digests, etc)

Problem Statement

- The mechanisms used to secure the z/TPF keystore should be updated periodically to ensure we use newer cryptographic techniques, stronger encryption algorithms, and so on.

To Be: New z/TPF Keystore Security Level

- A new z/TPF keystore security level has been introduced that users can transition their z/TPF keystore to use.
  - The new security level will secure the keys within the keystore using newer cryptographic techniques, stronger encryption algorithms, etc.

To Be: Migrating the Keystore to the Latest Security Level

- A new ZKEYS MIGRATE command is available to migrate the persistent version of the z/TPF keystore to the latest security level.
  - Once the persistent keystore is migrated, an IPL of each processor in the loosely-coupled complex is required to complete the migration.
    - The IPL of each processor can be done at a future time. The keystore is still functional after the ZKEYS MIGRATE.
- Migrating the keystore should not have any effect on existing applications.

Value Statement

Migrating the z/TPF keystore to the latest security level provides better security for the z/TPF keystore.

Delivered with PJ46545 (Feb 2022)

Conclusion

- Improved throughout of high-speed connector when endpoint groups are used as a one-way pipe outbound to remote systems.
    - **New Socket Selection Algorithm for High-Speed Connector (PJ46544)**
- Improvements to allow z/TPF clients to interface with cloud-based systems that are hosting multiple hosts.
    - **Server Name Indicator Extension Support for TLS Clients (PJ46661)**
- Improve the security of the z/TPF system by having better control over which cipher algorithms are used in z/TPF servers.
    - **TLS Cipher Preference for z/TPF Servers (PJ46661)**
- Improve the security of the z/TPF keystore by improving the cryptographic algorithms that are used to secure the keys.
    - **Improve the Security of the z/TPF Keystore (PJ46545)**

The Log4J Vulnerability and how it affects the z/TPF Product Family

Background

- In December 2021, numerous vulnerabilities were discovered with the Apache Log4j opensource package.
  - The vulnerabilities exposed flaws in Log4j where a malicious attacker could cause denial of service attacks and even execute code on distributed systems.
  - Log4j is widely used on platforms around the world.
  - The z/TPF system as well as the TPF Operation Server (TOS) use the Apache Log4j package.

z/TPF Vulnerability Remediation

- Vulnerability CVE-2021-44228
  - z/TPF Security Bulletin: https://www.ibm.com/support/pages/node/6528436
  - Apply z/TPF APAR PJ46688 (Dec 2021)
  - Apply TPF Operation Server APAR IT39419 (Dec 2021)
- Vulnerability CVE-2021-45105 and CVE-2021-45046
  - z/TPF Security Bulletin: https://www.ibm.com/support/pages/node/6538936
  - Apply z/TPF APAR PJ46693 (Jan 2022)
  - Apply TPF Operation Server APAR IT39522 (Jan 2022)

What’s Next?

Elliptic Curve Cryptography

Elliptic Curve Cryptograpy – Problem Statement

- The industry is moving away from using RSA for exchanging the secret symmetric key of a TLS session.
  - The industry is moving towards ephemeral public/private keypairs when exchanging the secret symmetric key
    - Ephemeral means public/private key used to exchange the secret symmetric key of an OpenSSL session is uniquely generated for each session
    - Ephemeral public/private keys provides ‘Perfect Forward Secrecy”
      - Limits the exposure if the private key is somehow compromised
  - Added ephemeral Diffie-Hellman (DHE_*) ciphers to z/TPF OpenSSL in Dec 2020 (APAR PJ46292)
    - Operations are only performed in software – expensive!

Elliptic Curve Cryptography – Hardware Acceleration

- The IBM z15 processor provides hardware acceleration to perform scalar multiply operations
  - Scalar multiply is the most expensive computational component of ECC when starting TLS sessions.
  - Scalar multiply is used when …
    - Creating the ephemeral ECC public/private key pair
    - Deriving the secret symmetric key to use for the TLS session

- The hardware acceleration is performed in the CPACF hardware
  - Same on-chip coprocessor used to perform AES, SHA, etc.

```
CPAC0012I 11.27.36 CPACF QUERY DISPLAY
   SHA-1:        ENABLED
   DES/TDES:     ENABLED
   AES-128:      ENABLED
   SHA-256:      ENABLED
   AES-256:      ENABLED
   SHA-512:      ENABLED
   DRNG:         ENABLED
   TRNG:         ENABLED
   AES-128-GCM:  ENABLED
   AES-256-GCM:  ENABLED
   SHA-384:      ENABLED
```

Elliptic Curve Cryptography Coming to z/TPF

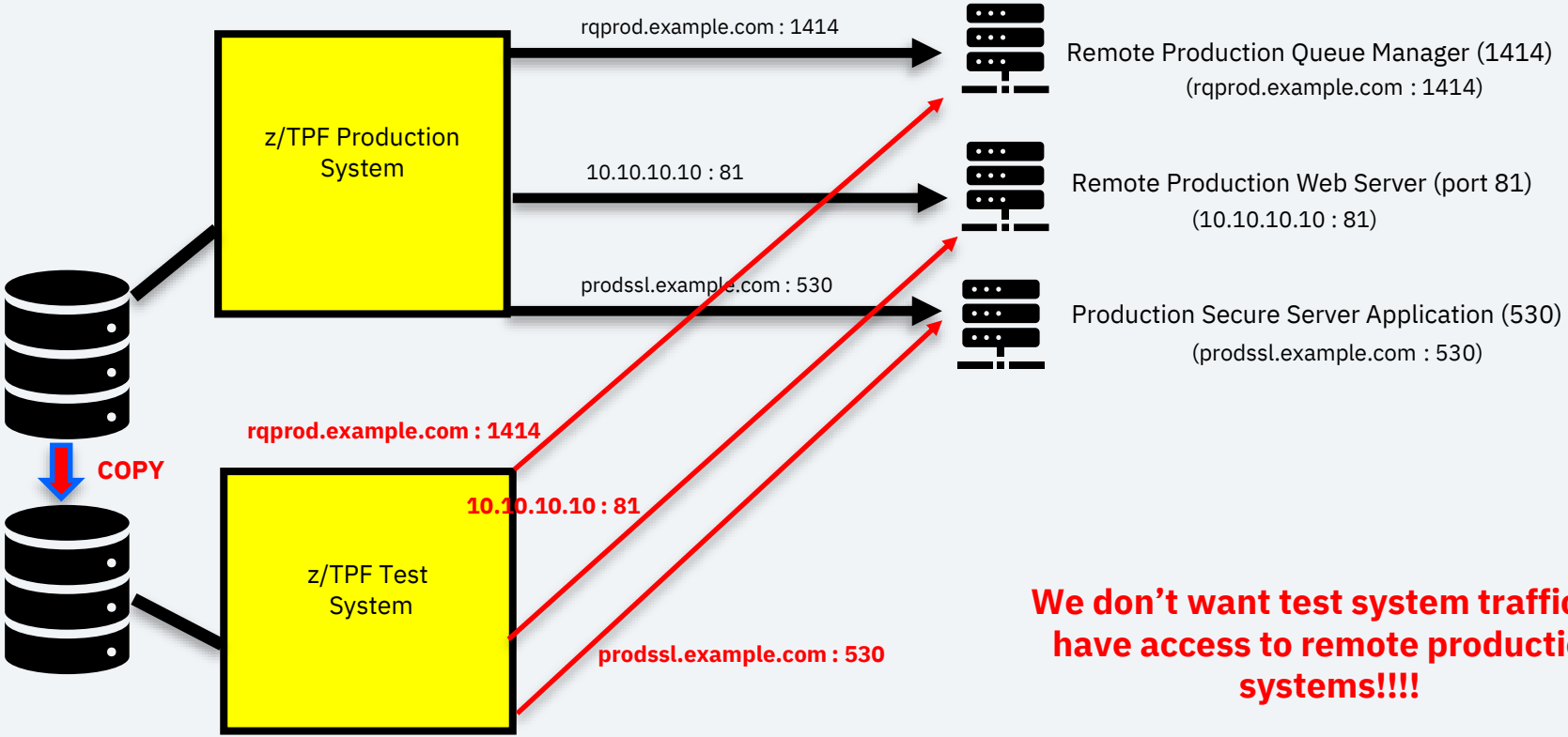
- We are adding the following ephemeral elliptic curve cryptography ciphers to z/TPF's OpenSSL
  - ECDHE-RSA-AES128-SHA256
  - ECDHE-RSA-AES256-SHA384
  - ECDHE-RSA-AES128-GCM-SHA256
  - ECDHE-RSA-AES256-GCM-SHA384
- Use ephemeral ECC for key exchange, but still use RSA for certificate authentication and validation.
- Operations in ECC will be hardware accelerated when on an IBM z15 or higher
  - IBM z14 and below, ECC operations will be done in software making it very expensive
- When z/TPF looks at support of TLS 1.3 in the future:
  - Use of RSA for key exchange is not supported in TLS 1.3
  - Start the move to ECC soon than later in preparation for TLS 1.3
- Target Date is 3Q 2022

Test System Override of Remote IP Connections

Test System Override – Problem Statement

- Many customers make a copy of a production system as a base for test systems.
  - A physical copy of the entire system is made
    - Customer databases, saved configuration information, etc.
- The copied test system is then modified to…
    - Scrub sensitive data from customer databases
    - Redefine IP connectivity for the system
    - Update configuration to prevent access to production
      - For example, delete MQ channels and re-define them for test.
- Updating all application and middleware that accesses remote production systems is time consuming and error prone!

Test System Override – Problem Statement Example

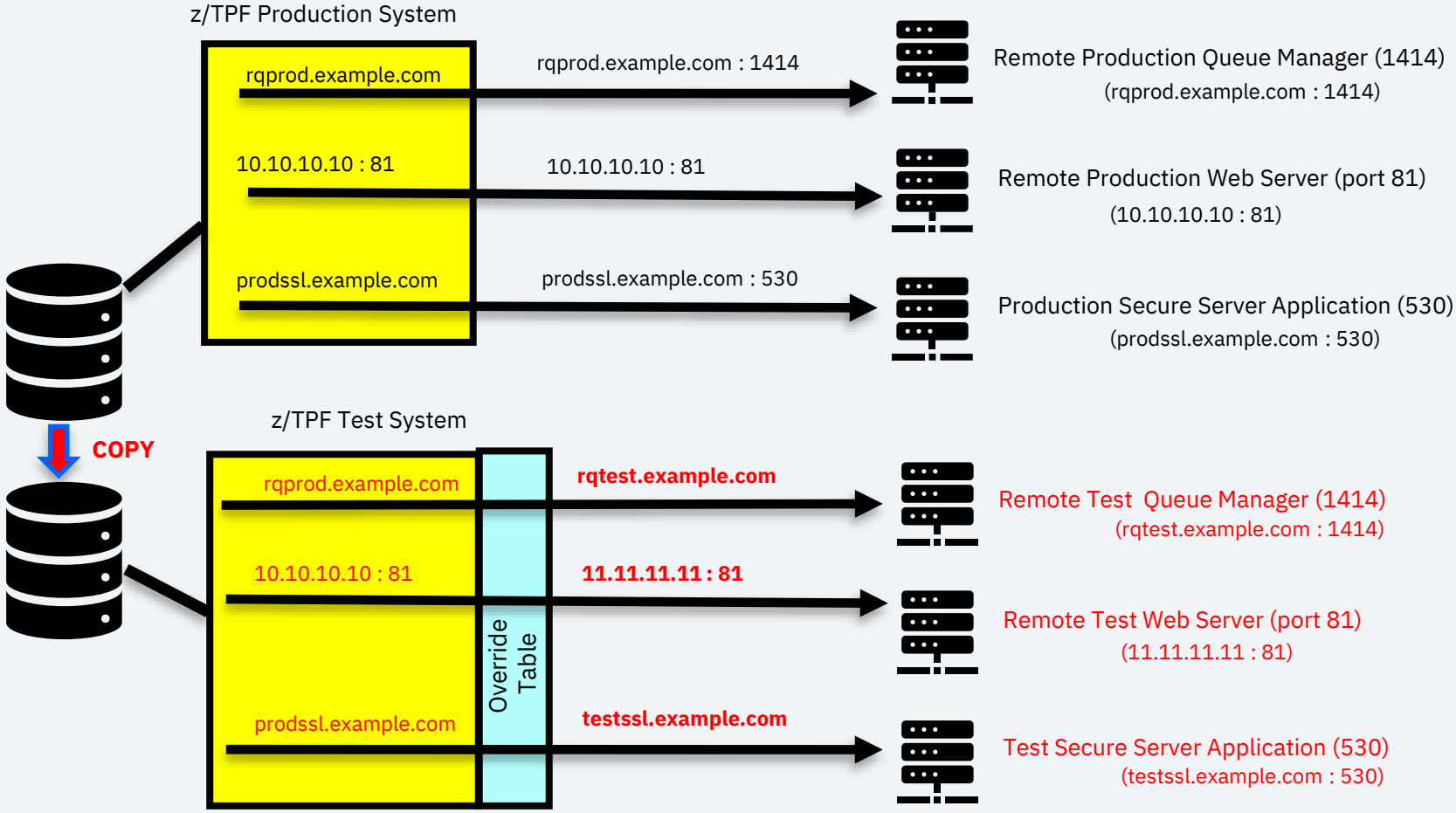


We don’t want test system traffic to have access to remote production systems!!!!

Test System Override – Solution

- Looking at a z/TPF command driven test system override utility.
  - For example:
    - ZTTCP OVERRIDE HOST-myprod.com NEWHOST-mytest.com
    - ZTTCP OVERRIDE IP-3.3.3.3 NEWIP-4.4.4.4
- The override commands can be used as part of turning over the copied production system to test system.
- Allows you to leave the production remote connectivity configuration in place
  - MQ Definitions, High Speed Connector files, home grown applications, etc.

Test System Override – Solution Example



We want sponsor users!

Our development cycle is driven by your feedback.

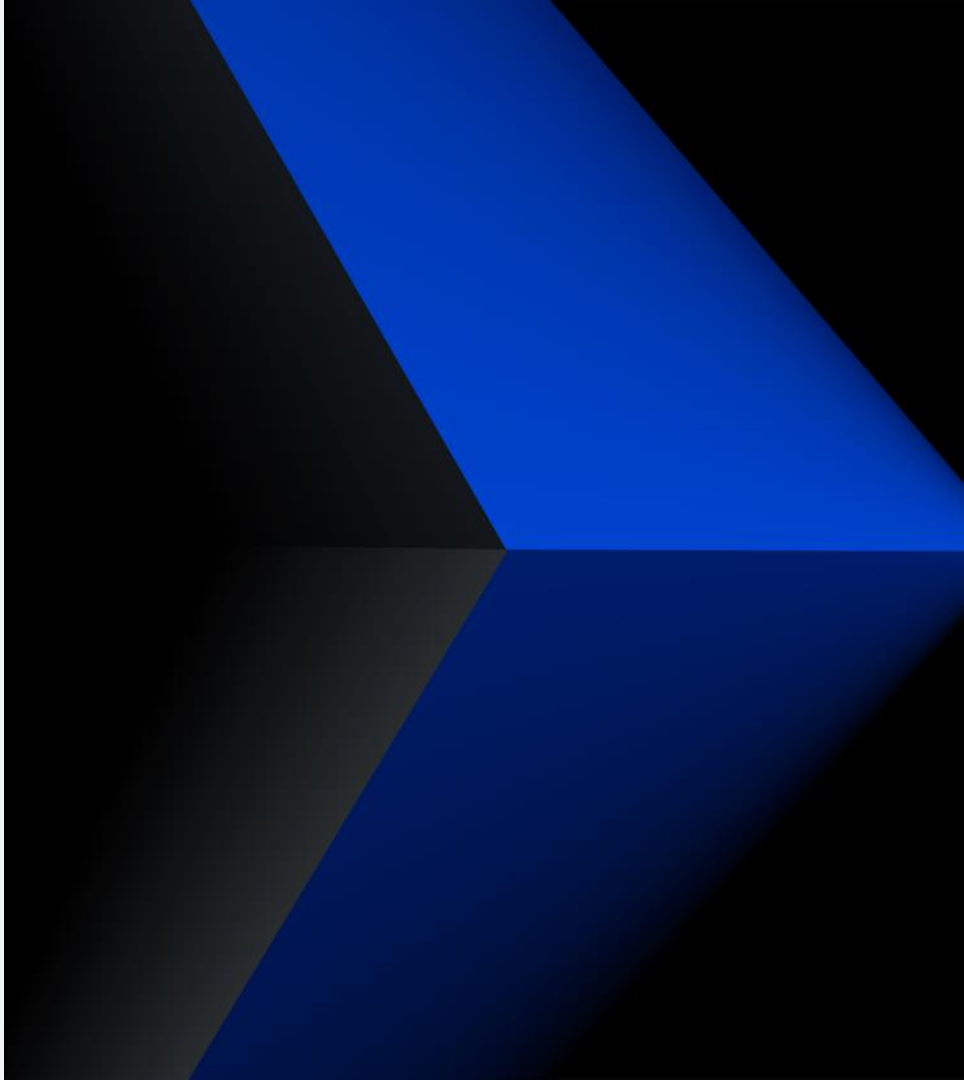
We are looking for sponsor users to assist in design and implementation, targeting the following personas:

- System Administrator

We have already begun engaging with the sponsor users in January 2022.

If you are interested in participating as a sponsor user, please contact:

Jamie Farmer (jvfarmer@us.ibm.com)



Thank you