z/TPF TCP/IP Communications

Jamie Farmer
z/TPF Development

Agenda

- TCP/IP and z/TPF
  - z/TPF Unique Socket APIs
  - z/TPF Network Services Database (NSD)
- Open Systems Adapter
- Flow of a TCP/IP Message

Agenda

- TCP/IP and z/TPF
  - z/TPF Unique Socket APIs
  - z/TPF Network Services Database (NSD)
- Open Systems Adapter
- Flow of a TCP/IP Message

TCP/IP and z/TPF
Not like every other platform…Kernel Sockets

- On most platforms (UNIX, Windows, Linux) sockets are process scoped
  - Sockets are tied to the process, much like an open file descriptor
  - If the process exits, any open descriptors (sockets or file) are closed

- z/TPF Sockets are Kernel based sockets
  - The sockets are owned by the system
  - Allows for sharing of sockets across processes (ECBs)
  - Allows for Asynchronous I/O (ie. activate_on_receipt)
    - Allowing for ECBs to exit while socket connections remain active
- **Kernel sockets are the reason the system has a socket sweeper.**

TCP/IP and z/TPF
Sockets and z/TPF

- An active socket is a communication path between one application to another
- Socket is assigned a socket descriptor by the z/TPF system along with a socket block
- The z/TPF socket block contains the information about the socket including:
  - Remote and local IP addresses
  - Remote and local port numbers
  - Protocol
  - Receive and send buffer anchors
  - Socket characteristics (buffer sizes, blocking/non-blocking mode, etc.)
- Socket apis are used by applications to send and receive data

TCP/IP and z/TPF
The Socket Block Table

- The z/TPF Socket Block Table is the z/TPF core storage that contains a socket block entry for each active socket on the z/TPF system
  - Socket Blocks are the chief mechanism to control the Socket API.
  - The socket block table is backed by 1 meg frames
    - 1 meg frames are used on demand as sockets are needed
      - Once obtained for sockets, 1 meg frames are never returned

- The number of socket block entries is defined in the SNAKEY macro (MAXSOCK parameter).
  - z/TPF supports up to 1 million active sockets
  - The number of socket blocks defined can be dynamically increased using the ZNKEY MAXSOCK-xx command

- The range of socket descriptors for z/TPF is: x’C00001’ - C’FFFFFF’

TCP/IP and z/TPF
The Socket Block Table Layout

The Socket Block Table is described in the ISOCK macro.

Socket Block Header
- System wide z/TPF TCP/IP information
  - Locking, Chains, Status, etc

8K in size

IP Hash Buckets
- Used to locate sockets by local/remote IP/port and protocol information

128K buckets (1meg)

Socket Descriptor Hash Buckets
- Used to locate sockets socket descriptor

128K buckets (1meg)

Socket Block Entries
- Individual socket information
  - Status, send/recv buffers, etc

Each entry is 4K in size

TCP/IP and z/TPF
The IP Message Table (IPMT)

The **IP Message Table (IPMT)** is the primary storage used to hold input and output TCP/IP data in z/TPF

- The IPMT is a set of shared storage for the system
- The size of the IPMT is defined in the SNAKEY macro (IPMTNUM parameter defined in number of meg)
  - The size can be dynamically increased using ZNKEY IPMTNUM
  - The IPMT table is backed on demand by z/TPF 1-meg frames (once obtained for IPMT they are never released)
  - The IPMT storage is divided into 4-K entries (256 entries per 1 meg of IPMT)

- The IPMT holds:
  - To hold output data while being sent and acknowledged.
  - To hold input data when it is being delivered to an application
  - To hold inbound data received from the network:
    - TCP data received out of order until missing data is received
    - To hold IP fragments until the entire message is received
    - To hold input packets that contain data and need to be processed by the upper layers (IP Opzero, TCP Opzero, UDP Opzero)

TCP/IP and z/TPF
Tuning the IP Message Table Size

- The amount of IPMT used is dependent on many factors:
  - Message rate (number of packets sent and received per second)
  - Average size of a packet
  - Size of the socket’s send and receive buffers
  - Round-trip time for TCP sockets, which means how long it takes, on average, to receive responses from remote partners
  - Messages that are read in from the network to the IPMT during dump processing and during input list shutdown.

- The amount of IPMT entries should be at least 5-10 times the number of sockets.
  - So, if the number of sockets defined is 20,000
    - 100,000 – 200,000 IPMT entries (400-800 1-meg frames)

- Can monitor with the ZTTCP DISP STATS command.

TCP/IP and z/TPF
Display TCP/IP Statistical Information - ZTTCP DISP STATS

```
zttcp disp stats
CSMP0097I 23.22.58 CPU-B SS-BSS  SSU-HPN  IS-01 _
TTCP0182I 23.22.58 BEGIN ZTTCP STATS DISPLAY

                         NUMBER   CURRENT MAXIMUM   MAX IN   MAX IN
                         DEFINED   IN USE  IN USE  USE DATE USE TIME
SOCKET BLOCK ENTRIES         100       11      13    29JUL  13.51.12 _
IP MESSAGE TABLE BLOCKS     1280        0     122    30JUL  13.54.59
IPMT FRAGMENT BLOCKS         320        0       0

          30 IP PACKETS SENT
       17434 IP PACKETS RECEIVED
           0 CHECKSUM ERRORS DETECTED
           8 IP FRAGMENTS RECEIVED
          20 TCP MESSAGES RECEIVED OUT OF ORDER _
          80 TCP MESSAGES RETRANSMITTED DUE TO TIMEOUT
           5 TCP MESSAGES FAST RETRANSMITTED
           2 TCP SOCKETS CLEANED UP BECAUSE OF RETRANSMIT TIMEOUTS
           2 IP FRAGMENTS DISCARDED BECAUSE OF EXCEEDING MAXFRAG

END OF ZTTCP DISPLAY+
```

TCP/IP and z/TPF
Monitoring Sockets Using ZSOCK

The ZSOCK command can perform many different functions, including:

- *Display TCP/IP native stack support control block information.*
- *Display the number of bytes sent and received across an individual TCP/IP socket in a 5-second interval.*
- *Convert TCP/IP native stack support resource information.*
- *Deactivate TCP/IP native stack support sockets.*
- *Display a summary table of socket descriptors and selected socket control block information.*
- *Disable and enable the creation of TCP/IP native stack support sockets in the z/TPF system.*
- *Display the sockets that are using the most IP message table (IPMT) blocks.*
- *Display socket rate information for all applications.*
- *Display socket trace information for a specified socket.*
- *Display the active sockets that have the most total exceptions or the most exceptions of a specific type.*
- *Display socket options.*

TCP/IP and z/TPF
ZSOCK Individual Socket Block Formatted Display

```
zsock disp format socket-C0000E
CSMP0097I 20.03.35 CPU-B SS-BSS  SSU-HPN  IS-01
SOCK0043I 20.03.35 TCP SOCKET CONTENTS FORMATTED
SOCKET BLOCK ADDR 000000020001E000
LOCAL IP -          9.057.013.085  LOCAL PORT  -                   21
REMOTE IP -         9.056.224.021  REMOTE PORT -                50363
PROTOCOL -                    TCP  SOCKET TYPE -               STREAM
SOCKET DESCRIPTOR -      00C0000E  1052 STATE -                     N
FIRST HOP IP -      9.057.013.001  VLAN ID -                        0
SEND BUFF SIZE -            32767  SEND BUFF IN USE -               0
RECV BUFF SIZE  -           32767  RECV BUFF IN USE -               0
BYTES SENT -                  227  BYTES RECEIVED -                37
NEXT SEND SEQ -        1139237436  LAST ACKED SEQ -        1139237436
NEXT RECV SEQ -         664483604  MAX SEGMENT SIZE -            1452
WINDOW SCALE -                  1  SEND WINDOW SIZE -           14600
STATE -               ESTABLISHED  AVG ROUND TRIP -          0.005062
MAX PACKET SIZE -            1492  SEND WINDOW BLOCKED -            N
CONGESTION WIN -             3130  SLOW START THRESH -          65535
ZERO WINDOW SENT -              0  ZERO WINDOW RCVD -               0
CURRENT SOCRATE -               0  MAXIMUM SOCRATE-                 0
SOCRATE LIMIT REACHED -         0  INPUT MESSAGE PRIORITY -         0
RETRANSMITS -                   0  OUT OF ORDER -                   0
FRAGMENTS IN -                  0  FRAGMENTS OUT -                  0
SEND ECBS QUEUE THRESHOLD -    10  SEND ECBS QUEUE LENGTH -         0
CLOSE ISSUED -                  N
DNS NAME - linuxtpf.pok.ibm.com
AOR PENDING -                   N
AOR TOKEN -                        AOR PROGRAM NAME -
SOCKET CREATED - TUE JUL 30 20.02.52 2013
END OF DISPLAY+
```

TCP/IP and z/TPF
ZSOCK Summary Display

```
zsock sum lport-9999
CSMP0097I 22.18.17 CPU-B SS-BSS  SSU-HPN  IS-01
SOCK0021I 22.18.17 SOCKET SUMMARY INFORMATION
SOCKET     LOCAL            LOCAL   REMOTE          REMOTE    PROT   STATE
 DESC       IP               PORT     IP             PORT

00C00016                     9999                             TCP    LISTEN
00C00017   9.057.013.050     9999   9.057.013.051 49166       TCP    ESTABLISHED
00C00018   9.057.013.050     9999   9.057.013.051 49164       TCP    ESTABLISHED
00C00019   9.057.013.050     9999   9.057.013.051 49167       TCP    ESTABLISHED
00C0001A   9.057.013.050     9999   9.057.013.051 49168       TCP    ESTABLISHED
00C0001B   9.057.013.050     9999   9.057.013.051 49165       TCP    ESTABLISHED
00C0001C   9.057.013.050     9999   9.057.013.051 49169       TCP    ESTABLISHED
SUMMARY TOTAL         7
END OF DISPLAY
```

TCP/IP and z/TPF
ZSOCK Display of Sockets With Highest IPMT Usage

```
ZSOCK IPMT TOP-10

SOCK0033I 03.47.34 BEGIN IPMT USAGE DISPLAY

                                            SEND
RANK    FD    BLOCKS INPUT OUTPUT BLOCKED
---- -------- ------ ----- ------ -------
   1  00C00009    123    93     30    NO
   2  00C0013D    101     0    101   YES
   3  00C00056     96     1     95   YES
   4  00C00078     23    11     12    NO
   5  00C0002D     21    10     11    NO
   6  00C00037     21     9     12    NO
   7  00C00025      6     2      4    NO
   8  00C0011C      5     0      5    NO
   9  00C00043      5     1      4    NO
  10  00C00002      4     0      4    NO

END OF IPMT USAGE DISPLAY
```

Agenda

- TCP/IP and z/TPF
  - z/TPF Unique Socket APIs
  - z/TPF Network Services Database (NSD)
- Open Systems Adapter
- Flow of a TCP/IP Message

z/TPF Unique Socket APIs
z/TPF Special TCP/IP APIs

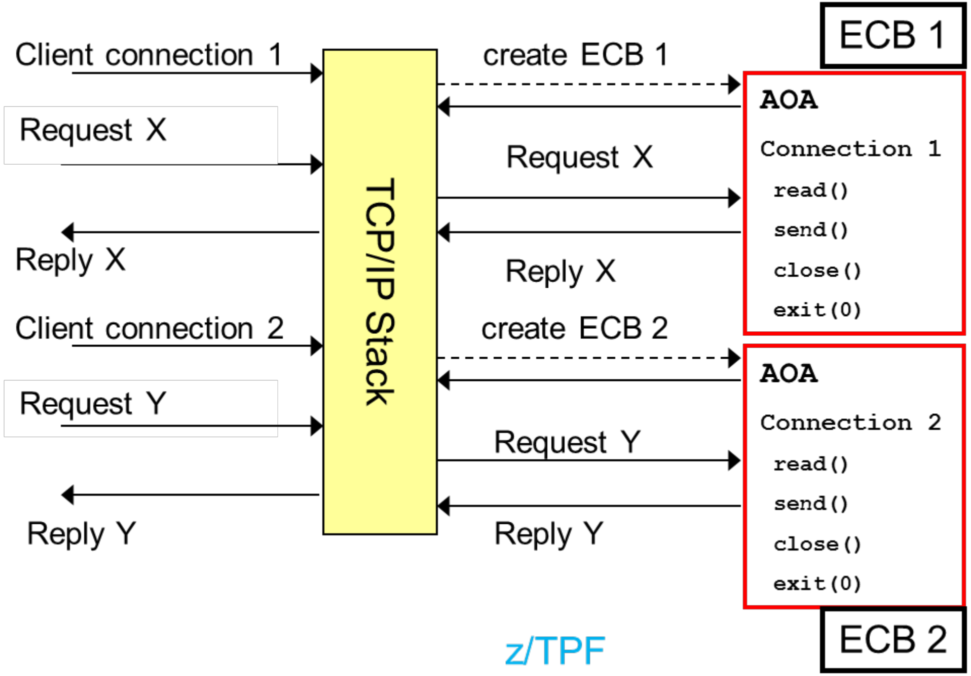
Activate_on_accept (AOA)
- Similar to accept API
- Reduce waiting ECB resources
- Eliminates long-running server ECBs
- Process reactivated once a client connection is established

Activate_on_receipt (AOR)
- Used with read, recv, recvfrom, tcp_read_TCP_message
- Reduce waiting ECB resources
- Eliminates long-running ECBs
- Process reactivated once inbound data arrives

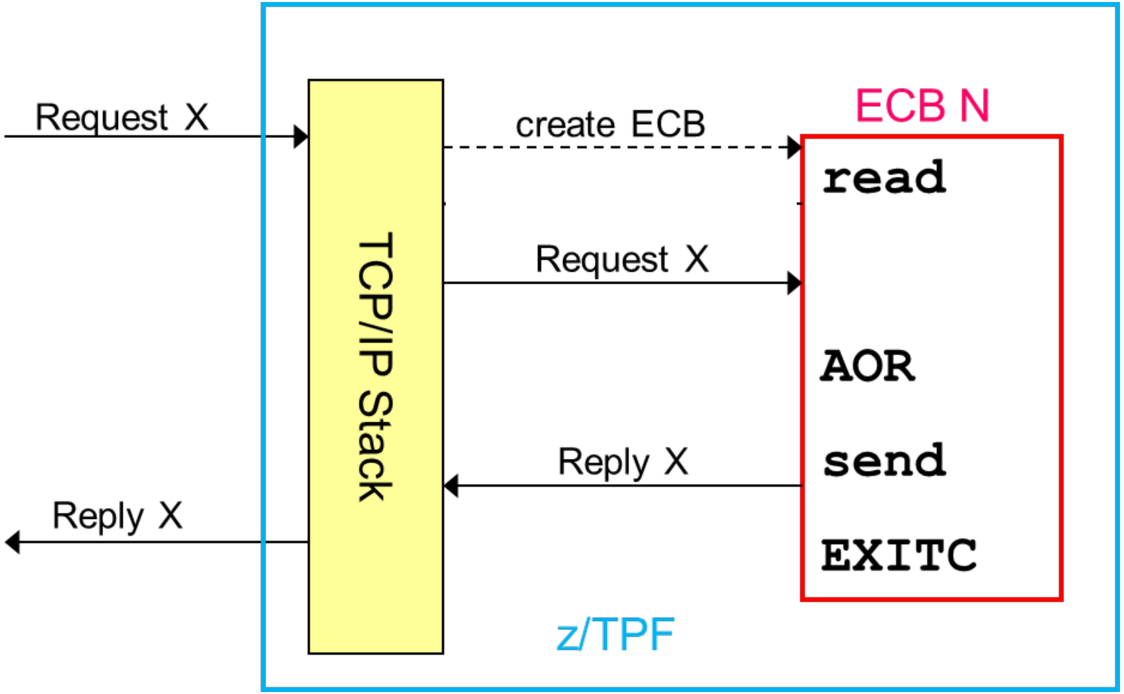
tpf_read_TCP_message and tpf_read_TCP_message2
- Allows application to read complete TCP message with one API call
- Reduces the amount o read APIs that need to be issued

activate_on_receipt_of TCP_message and activate_on_receipt_of_TCP_message2
- Used with tpf_read_TCP_message
- Process reactivated once entire TCP message arrives
- Reduces the amount of read APIs that need to be issued

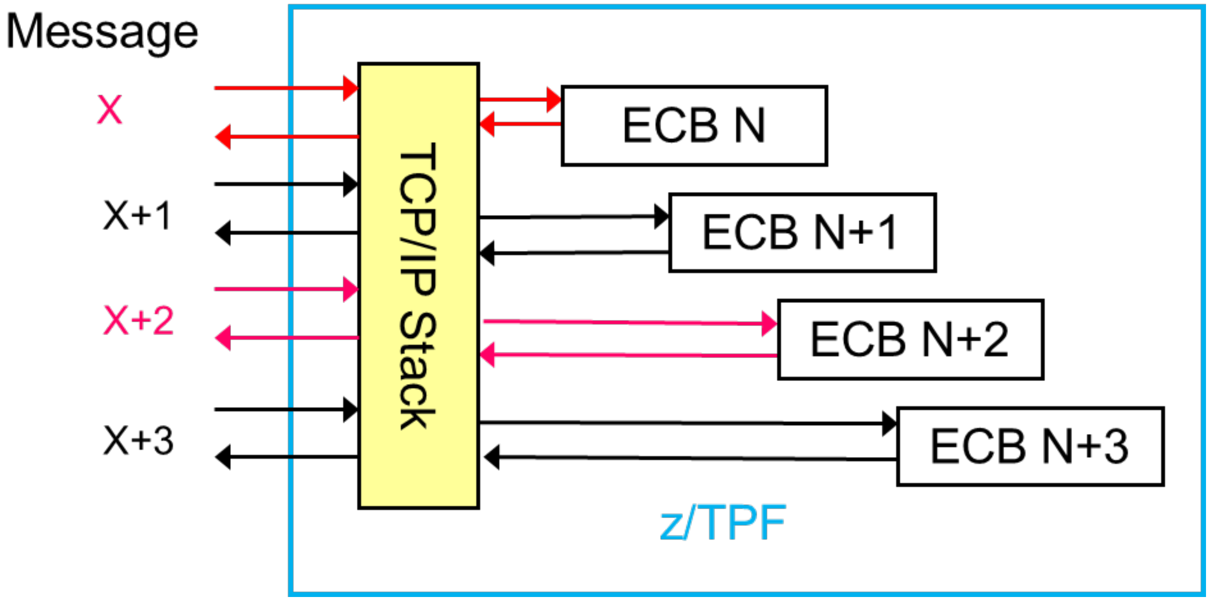
z/TPF Special TCP/IP APIs
TCP Connection Processing using AOA



z/TPF Special TCP/IP APIs
Request/Reply Processing using AOR



z/TPF Special TCP/IP APIs
Request/Reply ECBs using AOR



Agenda

- TCP/IP and z/TPF
  - z/TPF Unique Socket APIs
  - z/TPF Network Services Database (NSD)
- Open Systems Adapter
- Flow of a TCP/IP Message

z/TPF Network Services Database
TCP/IP Messages

- Data is sent and received as a byte stream
  - Nothing is sent to signal message boundaries
  - A “message” is defined by an application

- The system could count messages as every send API or read API
  - Not accurate
  - A single read could contain part of a message or may contain multiple messages

- Most accurate mechanism to count TCP messages is for the application to count
  - **tpf_tcpip_message_cnt** function allows applications to count messages

z/TPF Network Services Database
Counting TCP/IP Messages by Application

- The network services database (NSD) allows you to count messages by port number
  - The port number uniquely defines an application running (ie. Port 21 is FTP)

- Optionally specify a WEIGHT for the application
  - Applications without a WEIGHT
    - » **Message counts incremented on every read / send API**

  - Applications with a WEIGHT
    - » **Message counts incremented on tpf_tcpip_message_cnt function**

z/TPF Network Services Database

Example of Network Services Database With Weight

```
testSrv1   9999/tcp    weight-100    #Test Server1
testSrv2   9998/tcp                  #Test Server2
testSrv3   8888/tcp    weight-200    #Test Server3
```

- testSrv1/testSrv3 messages incremented using tpf_tcpip_message_cnt
- testSrv2 messages incremented by system (every send and read API)
- testSrv3 messages are weighted 2x of testSrv1 message
    - WEIGHT of 100 means a 1:1 ratio of RAW to weighted messages
    - Used in TPF data collection weighted message report

z/TPF Network Services Database

Online Display of the Network Services Database

```
ZIPDB DISP ALL
CSMP0097I 15.11.34 CPU-B SS-BSS  SSU-HPN  IS-01
IPDB0006I 15.11.34 NETWORK SERVICES DATABASE DISPLAY

 APPLICATION   PORT PROTOCOL  WEIGHT  TOS  APPLRATE  SOCRATE  PRIORITY IPTRSIZ
 -----------  ----- --------  ------  ---  --------  -------  -------- -------
    TESTSRV1   9999      TCP     100                                  5  SYSTEM
    TESTSRV2   9998      TCP    NONE                                  5  SYSTEM
    TESTSRV3   8888      TCP     200                                  5  SYSTEM

END OF DISPLAY
```

z/TPF Network Services Database
Example of Network Counting Discrepancy

```
ZIPDB MESSAGES ALL
CSMP0097I 15.15.33 CPU-B SS-BSS  SSU-HPN  IS-01
IPDB0004I 15.15.33 BEGIN PROCESSING MESSAGE RATES+
CSMP0097I 15.15.38 CPU-B SS-BSS  SSU-HPN  IS-01
IPDB0005I 15.15.38 MESSAGE RATES FOR A 5-SECOND INTERVAL

                          INPUT   INPUT      INPUT   OUTPUT  OUTPUT     OUTPUT
   APP NAME   PORT MSG/SEC PKT/SEC  BYTES/SEC MSG/SEC PKT/SEC  BYTES/SEC
  ---------- ----- ------- ------- ---------- ------- ------- ----------
   TESTSRV1   9999     693    4851    6930000     693    4851    6930000
   TESTSRV2   9998    4297    4838    6912000     691    4838    6912000  <-No WEIGHT
   TESTSRV3   8888     692    4849    6927452     692    4848    6926000
      OTHER              0       0          0       0       0          0

  ---------- ----- ------- ------- ---------- ------- ------- ----------
      TOTAL           5682   14538   20769452    2076   14537   20768000

END OF DISPLAY
```

Application just sits in a loop reading 10,000 byte messages from socket, then issues Send of 10,000 bytes to send the response.

z/TPF Network Services Database

Data Collection – Weighted Message Report

TCP/IP WEIGHTED INPUT MESSAGES BY APPLICATION

APPLICATION	PORT	WEIGHT	WEIGHTED MESSAGES	WEIGHTED MSGS/SEC	PERCENT OF TOTAL	CUMULATIVE PERCENT
TESTSRV2	9998	***	340878	3812.74	67.34%	67.34%
TESTSRV3	8888	200	110194	1232.53	21.76%	89.10%
TESTSRV1	9999	100	55126	616.59	10.89%	100.00%
TOTAL			506198	5661.85	100.00%	100.00%

The weighted message counts are used to calculate Mils per Weighted Message in Data Collection!!

INPUT MESSAGES PER SECOND (WORK LOAD)	MIN	MAX	MEAN
TCP/IP WEIGHTED MESSAGES	5584.132	6109.465	5840.164
...			
RESOURCE UTILIZATION PER MESSAGE			
MILLISECONDS PER WEIGHTED MESSAGE	0.053	0.061	0.057

z/TPF Network Services Database
Limiting Inbound Traffic

- The network services database (NSD) allows you to limit inbound messages by port.
  - » Ability to limit by socket or by all sockets associated with a port.
- Defining SOCRATE
  - » Defines the number of messages received per second for **each** connected TCP socket (TCP only)
- Defining APPLRATE
  - » Defines the number of messages received per second for **all** sockets associated with an application (UDP servers are single socket per port)
- Limiting traffic by port only occurs for server sockets (client socket connections are never limited.

z/TPF Network Services Database
Why Limit Traffic in the NSD?

- Control of input messages by port number (or application)
  - » Prevent an individual TCP connection from flooding the system with requests (ie. Looping client?)
  - » Prevent a single application from overwhelming the system with requests - Negatively affecting other message traffic
  - » Prevent denial of service attacks

z/TPF Network Services Database

Applications With A Defined Traffic Limit

The /etc/service File

```
testSrv1       9999/tcp    weight-100 applrate-500               #Test
Server1
testSrv2       9998/tcp    weight-100 applrate-500 socrate-100   #Test
Server2
testSrv3       8888/tcp    weight-200 socrate-100                #Test
Server3
```

Online Display Of NSD

```
ZIPDB DISP ALL
CSMP0097I 10.28.07 CPU-B SS-BSS  SSU-HPN  IS-01
IPDB0006I 10.28.07 NETWORK SERVICES DATABASE DISPLAY

 APPLICATION   PORT PROTOCOL  WEIGHT  TOS  APPLRATE  SOCRATE  PRIORITY IPTRSIZ
 -----------  ----- --------  ------  ---  --------  -------  -------- -------
    TESTSRV1   9999      TCP     100            500                  5  SYSTEM
    TESTSRV2   9998      TCP    NONE            500      100         5  SYSTEM
    TESTSRV3   8888      TCP     200                     100         5  SYSTEM

END OF DISPLAY
```

z/TPF Network Services Database

Reaching Traffic Limits

10 client connections started to port 9999

```
ZIPDB DISPLAY PORT-9999
CSMP0097I 10.35.25 CPU-B SS-BSS  SSU-HPN  IS-01 _
IPDB0003I 10.35.25 NETWORK SERVICES DATABASE DISPLAY

 NAME-  TESTSRV1  PORT- 9999  PROTOCOL-TCP  IPTRSIZE-SYSTEM
 WEIGHT-     100  TOS-     0  SOCRATE-NONE     PRIORITY-     5
 DEFINED APPLRATE LIMIT            500
 CURRENT APPL MESSAGE RATE         500 _
 HIGHEST APPL MESSAGE RATE         500
 NUMBER TIMES LIMIT REACHED          9

 CONNECTIONS                    INBOUND         OUTBOUND
 -----------                 ----------       ----------
 MAXCONN DEF                       NONE             NONE
 CURRENT VALUE                       10                0 _
 HIGHEST VALUE                       10                0
 NUMBER REJECTED                      0                0

 IP -             ANY
 --------------------        ----------
 BACKLOG DEFINED                      5
 CURRENT BACKLOG                      0
 HIGHEST BACKLOG                      0
 CONNECTIONS REJECTED                 0
```

z/TPF Network Services Database
Application Reaching Its Socket Rate Limit

3 client connections started to port 9998

```
ZIPDB DISPLAY PORT-9998
CSMP0097I 10.45.55 CPU-B SS-BSS  SSU-HPN  IS-01 _
IPDB0003I 10.45.55 NETWORK SERVICES DATABASE DISPLAY

 NAME-  TESTSRV2  PORT- 9998  PROTOCOL-TCP  IPTRSIZE-SYSTEM
 WEIGHT-     100  TOS-     0  SOCRATE-   100   PRIORITY-     5
 DEFINED APPLRATE LIMIT            500
 CURRENT APPL MESSAGE RATE         300 _
 HIGHEST APPL MESSAGE RATE         302
 NUMBER TIMES LIMIT REACHED          0

 CONNECTIONS                   INBOUND        OUTBOUND
 -----------                ----------      ----------
 MAXCONN DEF                      NONE            NONE
 CURRENT VALUE                       3               0 _
 HIGHEST VALUE                       3               0
 NUMBER REJECTED                     0               0

 IP -             ANY
 --------------------       ----------
 BACKLOG DEFINED                     5
 CURRENT BACKLOG                     0
 HIGHEST BACKLOG                     1
 CONNECTIONS REJECTED                0
```

z/TPF Network Services Database

Individual Socket Rate Display

```
ZSOCK DISP FORMAT SOCK-C00B03
CSMP0097I 10.48.14 CPU-B SS-BSS  SSU-HPN  IS-01
SOCK0043I 10.48.14 TCP SOCKET CONTENTS FORMATTED
SOCKET BLOCK ADDR 0000000200052000
LOCAL IP -          9.057.013.050  LOCAL PORT  -                9998
REMOTE IP -         9.057.013.051  REMOTE PORT -               49180
PROTOCOL -                    TCP  SOCKET TYPE -             STREAM
SOCKET DESCRIPTOR -      00C00B03  1052 STATE -                    N
FIRST HOP IP -      9.057.013.051  VLAN ID -                       0
SEND BUFF SIZE -           131072  SEND BUFF IN USE -              0
RECV BUFF SIZE  -          131072  RECV BUFF IN USE -            100
BYTES SENT -              1798301  BYTES RECEIVED -          1798317
NEXT SEND SEQ -        1716755359  LAST ACKED SEQ -       1716755359
NEXT RECV SEQ -        1716754521  MAX SEGMENT SIZE -           1452
WINDOW SCALE -                  4  SEND WINDOW SIZE -         131072
STATE -               ESTABLISHED  AVG ROUND TRIP -         0.000208
MAX PACKET SIZE -            1492  SEND WINDOW BLOCKED -           N
CONGESTION WIN -            98027  SLOW START THRESH -         51892
ZERO WINDOW SENT -              0  ZERO WINDOW RCVD -              0
CURRENT SOCRATE -             100  MAXIMUM SOCRATE-              100
SOCRATE LIMIT REACHED -       169  INPUT MESSAGE PRIORITY -        0
```

z/TPF Network Services Database
Application Reaching Its Socket Limits

Starting 10 clients 9998 – Now reach the APPLRATE!

```
ZIPDB DISPLAY PORT-9998
CSMP0097I 10.51.34 CPU-B SS-BSS  SSU-HPN  IS-01 _
IPDB0003I 10.51.34 NETWORK SERVICES DATABASE DISPLAY

 NAME-   TESTSRV2  PORT- 9998  PROTOCOL-TCP  IPTRSIZE-SYSTEM
 WEIGHT-      100  TOS-     0  SOCRATE-   100   PRIORITY-     5
 DEFINED APPLRATE LIMIT                  500
 CURRENT APPL MESSAGE RATE               500 _
 HIGHEST APPL MESSAGE RATE               500
 NUMBER TIMES LIMIT REACHED               12
```

Socket Display Now:

```
CURRENT SOCRATE -                64  MAXIMUM SOCRATE-                100
SOCRATE LIMIT REACHED -         321  INPUT MESSAGE PRIORITY -          0
```

z/TPF Network Services Database
What Happens When Traffic Limits Are Reached?

- What happens on the TPF side?
  - Message sits in socket's receive buffer, but system does not tell the application it is there
  - Application read, AOR, etc will indicate that no data is available

- What happens on the remote side
  - For request reply model sockets, the response is delayed and remote application sits on a read API waiting.
  - For one way pipe model sockets, the receive buffer on TPF will eventually fill, shutting down the sending on the remote side
    - » Normal TCP flow control (send window blocked on remote)

z/TPF Network Services Database

Limiting Inbound and Outbound Connections In the NSD

- The network services database (NSD) allows you to limit TCP inbound and outbound connections.
  - Ability to limit TCP connections based on server port number

- Defining MAXCONNIN
  - Defines the maximum number of **inbound** TCP connections that can be active at any given time.

- Defining MAXCONNOUT
  - Defines the maximum number of **outbound** TCP connections that can be active at any given time.

z/TPF Network Services Database

Why Would You Limit the Number of Connections?

- Prevent an individual client from flooding the system with TCP connections
  - Looping client?
- Prevent denial of service attacks

z/TPF Network Services Database
Applications with a Defined Connection Limit

The /etc/service File

```
testSrv1         9999/tcp    weight-100 applrate-500 maxconnin-5 maxconnout-0    #Test Server1
testSrv2         9998/tcp    weight-100 applrate-500 socrate-100                 #Test Server2
testSrv3         8888/tcp    weight-200 socrate-100                              #Test Server3
```

Online Display Of NSD

```
ZIPDB DISP NAME-testSrv1
CSMP0097I 14.40.33 CPU-B SS-BSS  SSU-HPN  IS-01 _
IPDB0003I 14.40.33 NETWORK SERVICES DATABASE DISPLAY

 NAME-  TESTSRV1  PORT- 9999  PROTOCOL-TCP  IPTRSIZE-SYSTEM
 WEIGHT-     100  TOS-     0  SOCRATE-NONE     PRIORITY-     5
 DEFINED APPLRATE LIMIT            500
 CURRENT APPL MESSAGE RATE           0
 HIGHEST APPL MESSAGE RATE           0
 NUMBER TIMES LIMIT REACHED          0

 CONNECTIONS                    INBOUND        OUTBOUND
 -----------                 ----------      ----------
 MAXCONN DEF                          5               0
 CURRENT VALUE                        0               0
 HIGHEST VALUE                        0               0
 NUMBER REJECTED                      0               0
```

z/TPF Network Services Database

Application Reaching Connection Limit Counts

Started connections from remote system into TPF until reaching the limit – subsequent are rejected
Attempted to start TCP client connections outbound – all are rejected!

```
ZIPDB DISP NAME-testSrv1
CSMP0097I 14.55.12 CPU-B SS-BSS  SSU-HPN  IS-01 _
IPDB0003I 14.55.12 NETWORK SERVICES DATABASE DISPLAY

 NAME-  TESTSRV1  PORT- 9999  PROTOCOL-TCP  IPTRSIZE-SYSTEM
 WEIGHT-     100  TOS-     0  SOCRATE-NONE     PRIORITY-     5
 DEFINED APPLRATE LIMIT               500
 CURRENT APPL MESSAGE RATE            500
 HIGHEST APPL MESSAGE RATE            501
 NUMBER TIMES LIMIT REACHED            72

 CONNECTIONS                          INBOUND          OUTBOUND
 -----------                       ----------        ----------
 MAXCONN DEF                                5                 0
 CURRENT VALUE                              5                 0
 HIGHEST VALUE                              5                 0
 NUMBER REJECTED                            8                 7
```

z/TPF Network Services Database
What Happens When Connection Limits Are Reached?

- Outbound Connections
  - Are rejected on the connect() API with a SOCCONNREFUSE error number returned

- Inbound Connections
  - Connection is rejected and abnormally terminated by the TPF stack with a TCP RST
    - » Reason Code “TCP CONNECTION LIMIT EXCEEDED”

```
RWI-01  IPCCW-D1  SOURCE IP-9.57.13.50  DEST IP-9.57.13.51  LEN-40
  TOD-D20F341FBEAB60D1  PROTOCOL-06 (TCP)  SOURCE PORT-9999  DEST PORT-49237
  SEQ-0  ACK-1107028504  WINDOW-0  URGENT OFFSET-0
  TCP FLAG BYTE-14 (ACK, RST)
  REASON CODE - TCP CONNECTION LIMIT EXCEEDED
  IP HEADER   45000028 D1920000 3C068067 09390D32 09390D33
  TCP HEADER  270FC055 00000000 41FBEA18 50140000 6F810000
```

z/TPF Network Services Database

Defining Input Message Priority

- The network services database (NSD) allows you to define an input message priority for inbound IP packets destined for the NSD port.
  - Input message priority of 1-9 (5 is the default)

- **Priority 1** means input packets will bypass input list shutdown checks – inbound packets placed on the ready list.

- **Priority 2-9**, are considered a discard priority
  - Honor input list shutdown – inbound packets added to the input list
  - In the event the system runs out of IPMT blocks, we will discard inbound packets based on the priority
    - » Priority 9 are discarded first.
  - Not widely used – never want to run out of IPMT blocks.

- Can override the NSD input message priority on a socket level using setsockopt() API

z/TPF Network Services Database
Defining Applications With an Input Priority

The /etc/service File

```
testSrv1        9999/tcp    weight-100 applrate-500 priority-1                  #Test Server1
testSrv2        9998/tcp    weight-100 applrate-500 socrate-100                 #Test Server2
testSrv3        8888/tcp    weight-200 socrate-100                              #Test Server3
```

Online Display Of NSD

```
ZIPDB DISP NAME-testSrv1
CSMP0097I 14.40.33 CPU-B SS-BSS  SSU-HPN  IS-01 _
IPDB0003I 14.40.33 NETWORK SERVICES DATABASE DISPLAY

 NAME-   TESTSRV1  PORT- 9999  PROTOCOL-TCP  IPTRSIZE-SYSTEM
 WEIGHT-      100  TOS-     0  SOCRATE-NONE     PRIORITY-     1
 DEFINED APPLRATE LIMIT             500
 CURRENT APPL MESSAGE RATE            0
 HIGHEST APPL MESSAGE RATE            0
 NUMBER TIMES LIMIT REACHED           0

 CONNECTIONS                    INBOUND        OUTBOUND
 -----------                 ----------      ----------
 MAXCONN DEF                       NONE            NONE
 CURRENT VALUE                        0               0
 HIGHEST VALUE                        0               0
 NUMBER REJECTED                      0               0
```

z/TPF Network Services Database

Defining Applications With an Input Priority

ZIPDB MESSAGES ALL

APP NAME	PORT	INPUT MSG/SEC	INPUT PKT/SEC	INPUT BYTES/SEC	OUTPUT MSG/SEC	OUTPUT PKT/SEC	OUTPUT BYTES/SEC
TESTSRV1	9999	514	514	51400	514	514	51420
TESTSRV2	9998	100	100	10000	100	100	10000
TESTSRV3	8888	0	0	0	0	0	0
OTHER		0	0	0	0	0	0
TOTAL		614	614	61400	614	614	61420

LISH0051I 14.44.21 INPUT LIST SHUTDOWN WAS DETECTED

ZIPDB MESSAGES ALL

APP NAME	PORT	INPUT MSG/SEC	INPUT PKT/SEC	INPUT BYTES/SEC	OUTPUT MSG/SEC	OUTPUT PKT/SEC	OUTPUT BYTES/SEC	
TESTSRV1	9999	482	482	48200	482	482	48200	← Messages received in ILS
TESTSRV2	9998	0	0	0	0	0	0	← No traffic in ILS
TESTSRV3	8888	0	0	0	0	0	0	
OTHER		0	0	0	0	0	0	
TOTAL		482	482	48200	482	482	48200	

z/TPF Network Services Database
High Priority Applications Should be Used With Caution

- High priority applications does not mean messages will be processed faster
  - But rather processed when system is resource constrained.

- **Misuse of high priority applications can cause system outages!**

Agenda

- TCP/IP and z/TPF
  - z/TPF Unique Socket APIs
  - z/TPF Network Services Database (NSD)
- Open Systems Adapter
- Flow of a TCP/IP Message

z/TPF Open Systems Adapter (OSA)
OSA-Express Support Overview

- System z network interface card for communication
  - Defined to processor / LPAR using the IOCP

- The z/TPF ZOSAE command creates and updates definitions for z/TPF OSA-Express connections

- Uses a proprietary IBM protocol to communicate with the hosts (ie. TPF)
  - Shared storage between OSA-Express connection and z/TPF system to read and write data.

- VIPA Support
  - Eliminates single points of failure for OSA devices on z/TPF
  - Ability to move away from affected OSA connection or affected processor in a L/C complex.

- OSA to OSA connectivity between LPARS
  - No packets sent on the network.

Open System Adapter (OSA)
VIPA Support

- Static VIPA – defined on a single z/TPF processor.
  - Can be associated with a single OSA-E connection.
  - Can swing to alternate OSA-E connection on the same processor.
    - Transparent to applications
    - Sockets remain active
  - Can be used for processor unique applications.

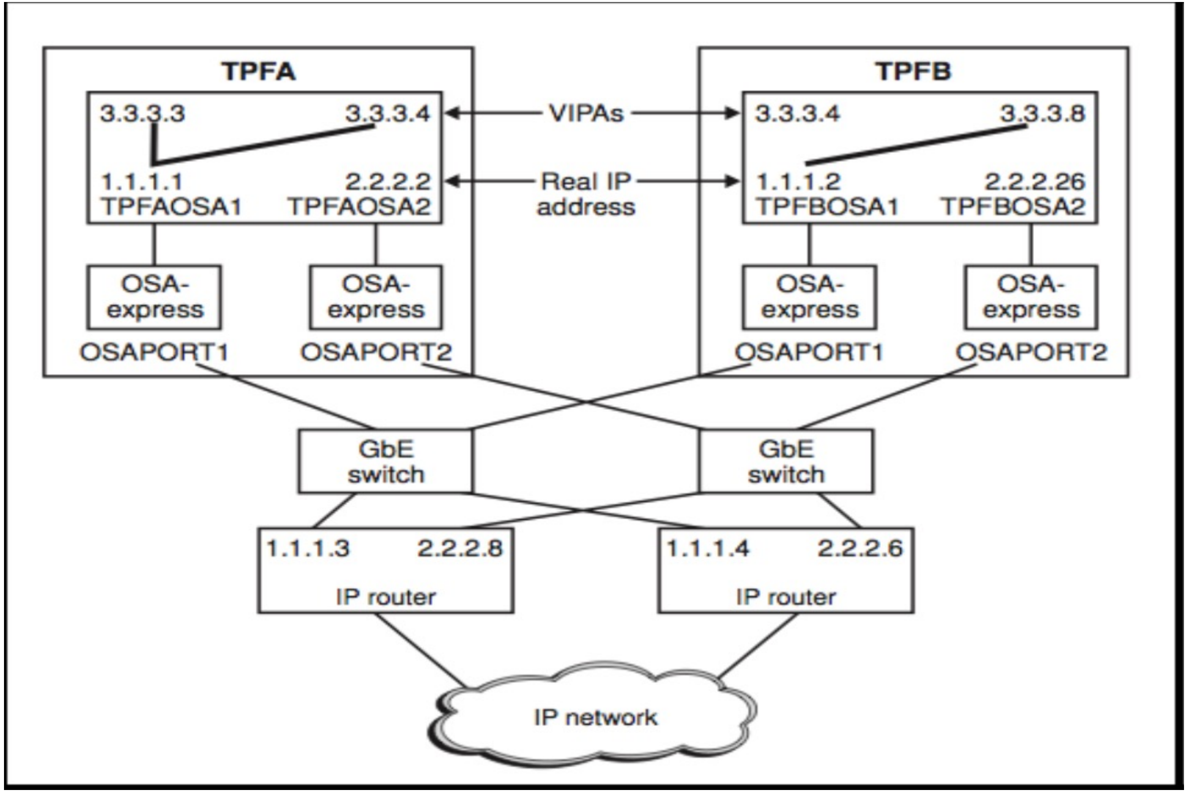
- Movable VIPA – can be defined on all z/TPF processors in loosely coupled complex.
  - Can only be active on a single z/TPF processor at a time.
  - Use for processor shared applications.
  - Use to load balance TCP/IP traffic in the complex.

- Moving VIPA to another processor
  - All sockets associated with the VIPA fail on current processor.
  - When remote clients reconnect, they will connect to VIPA on the new processor.

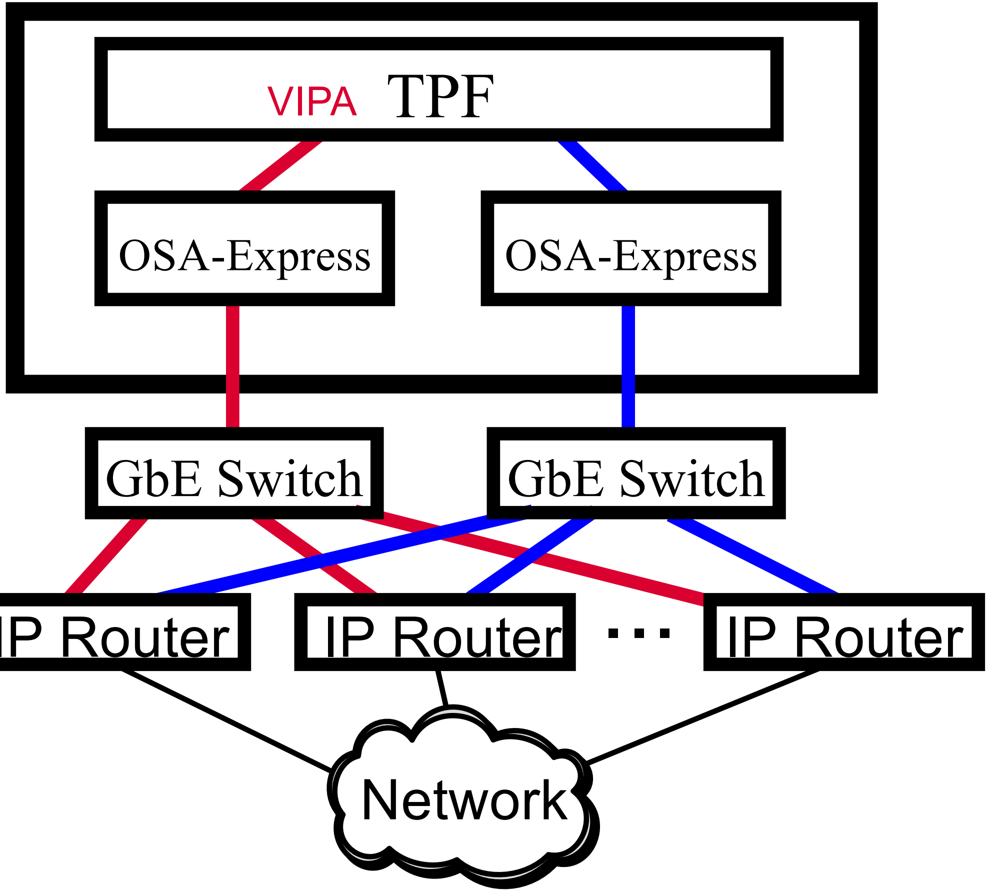
- Moving VIPA can be done
  - Automatically via the UVIP user exit.
  - ZVIPA command.
  - By application program (VIPAC macro or tpf_vipac() C function)

- Display VIPA information and statistics using the ZVIPA command.

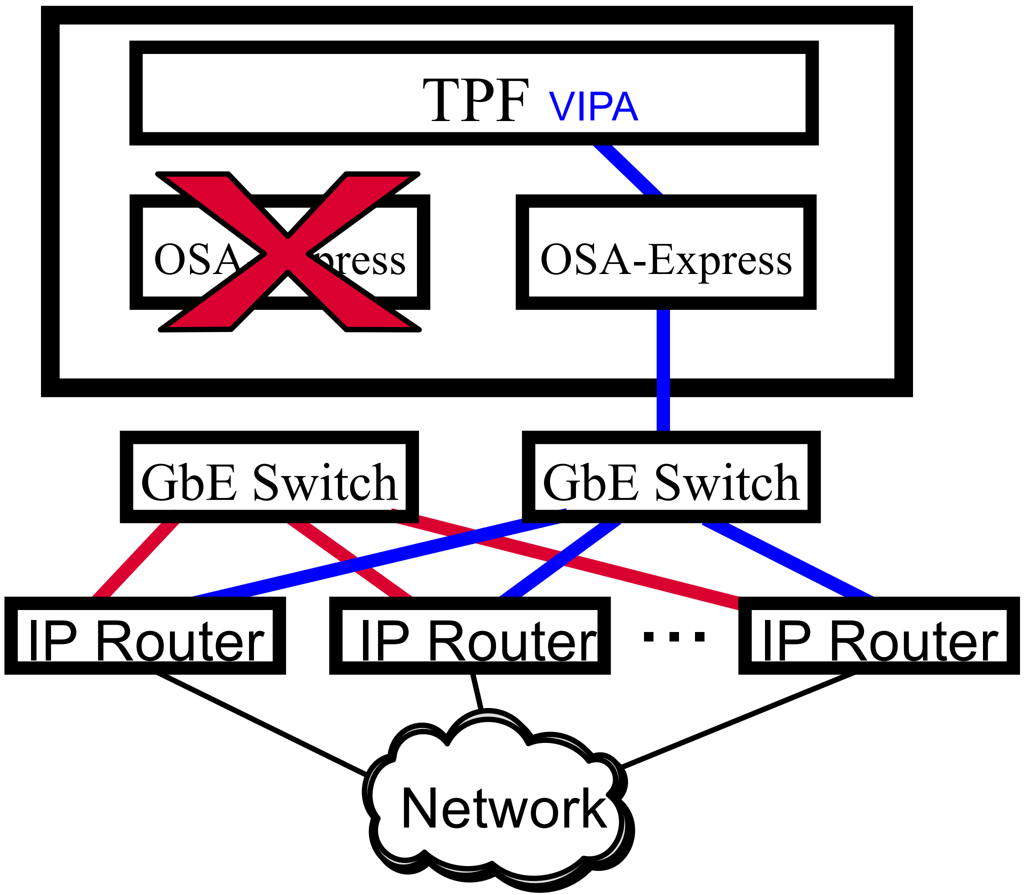
Open System Adapter (OSA)
Typical z/TPF OSA Configuration



Open System Adapter (OSA)
Swinging VIPAs to Alternate OSA



Open System Adapter (OSA)
Swinging VIPAs to Alternate OSA



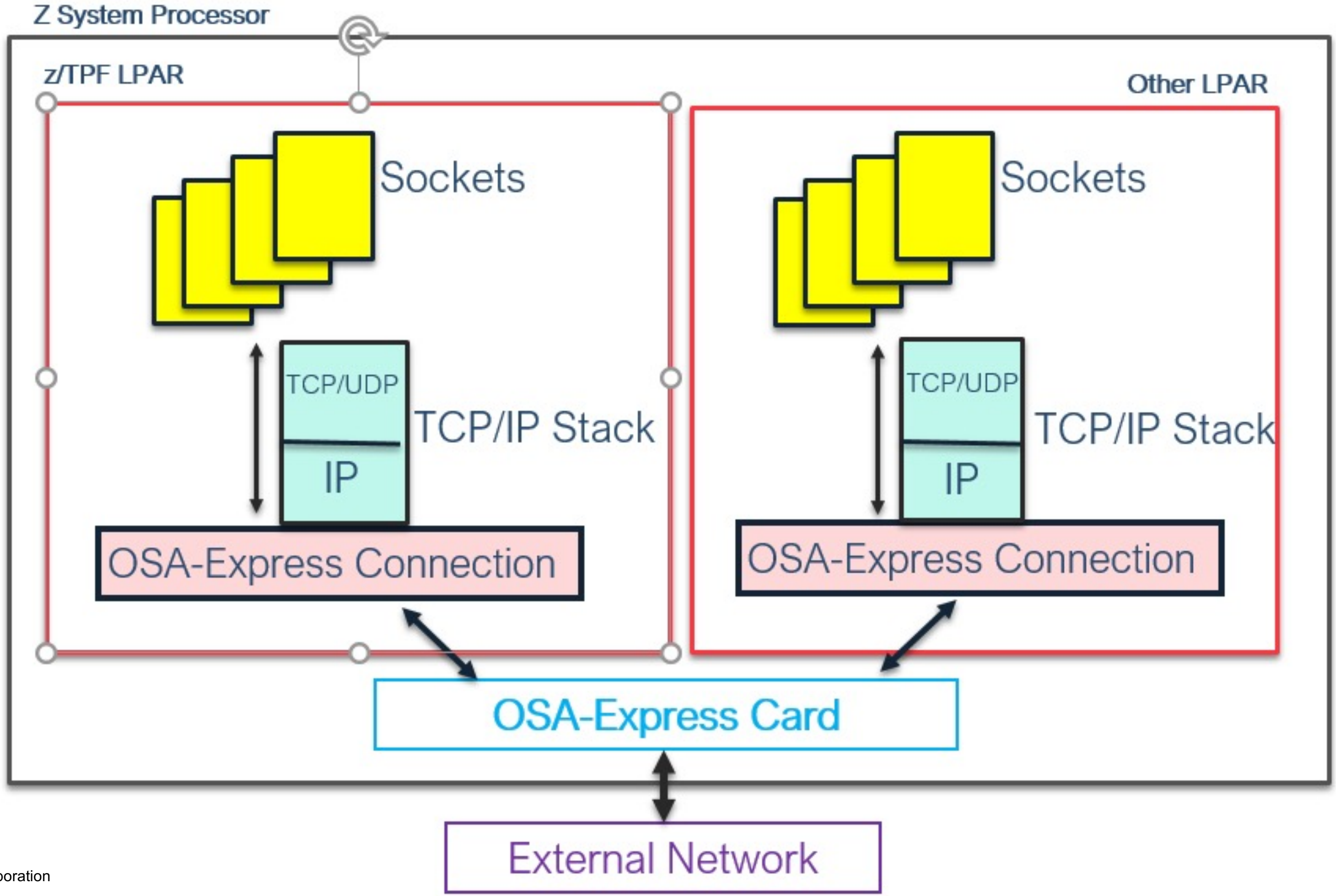
Open System Adapter (OSA)
OSA Related System Definitions

- SNAKEY macro MAXOSA parameter
  - The maximum number of OSA-Express connections that can be defined on the z/TPF system.

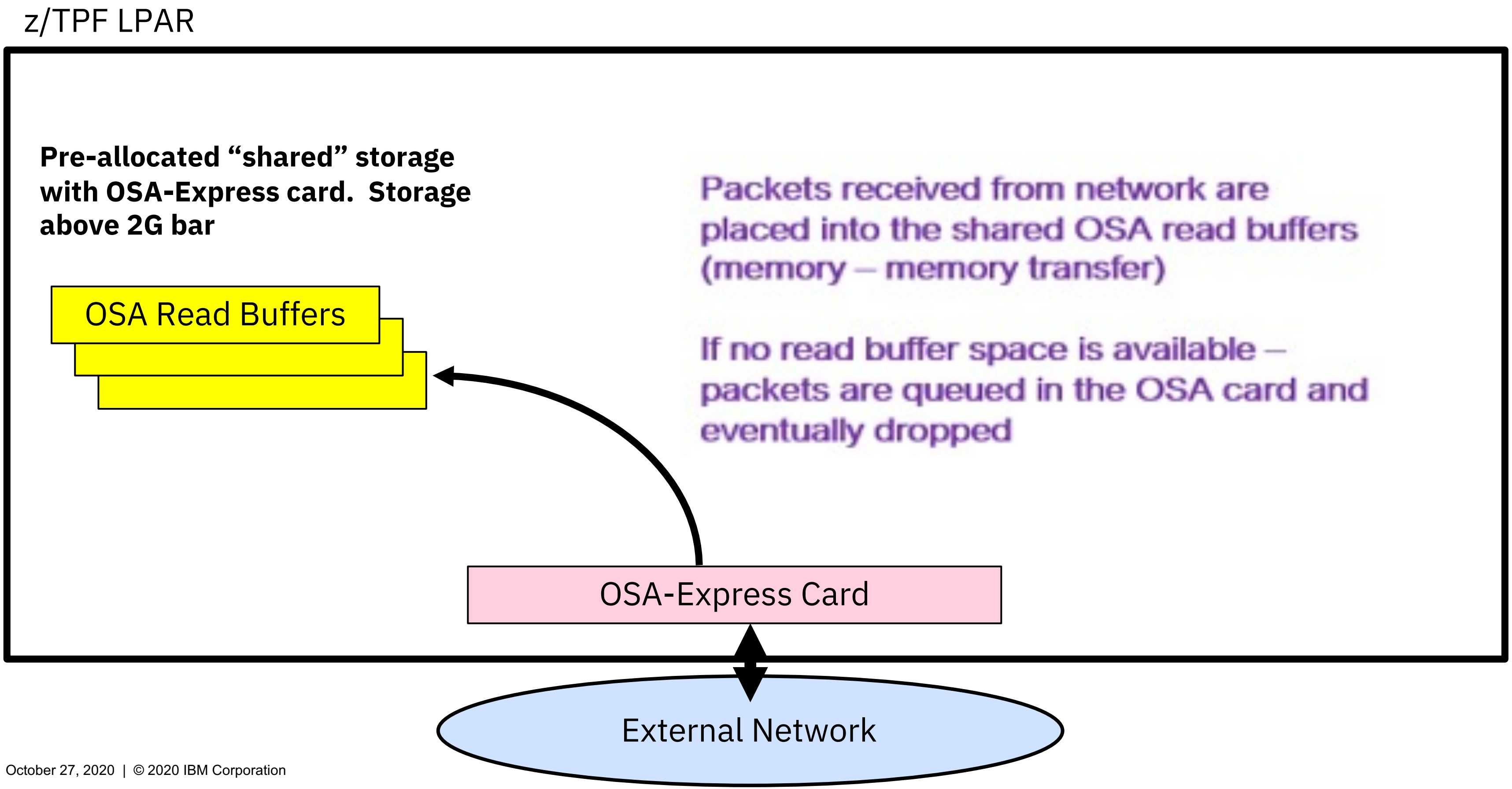
Agenda

- TCP/IP and z/TPF
  - z/TPF Unique Socket APIs
  - z/TPF Network Services Database (NSD)
- Open Systems Adapter
- Flow of a TCP/IP Message

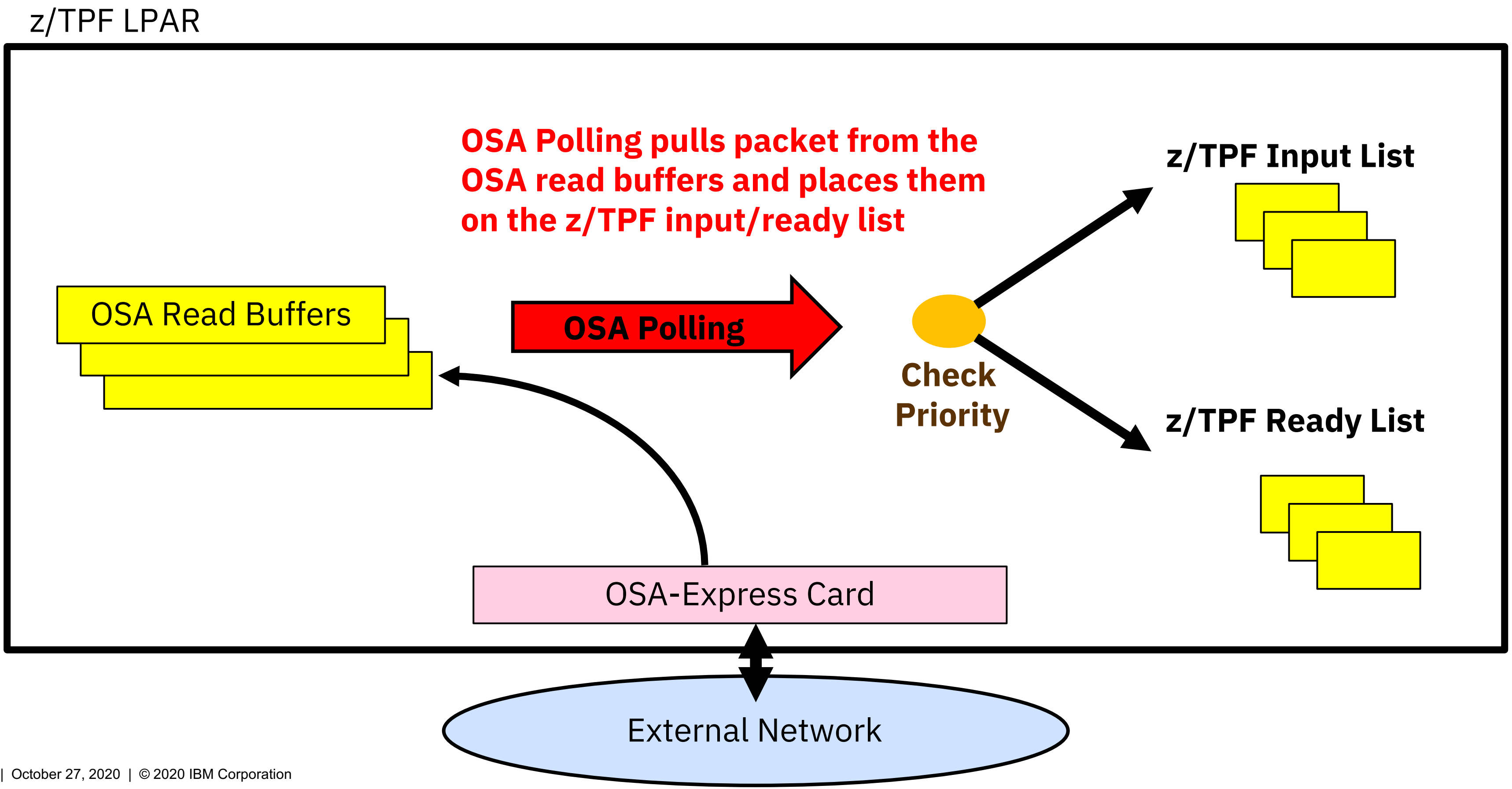
Flow of a TCP/IP Message
Typical OSA Configuration



Flow of a TCP/IP Message
TCP/IP Inbound Message Processing – OSA Polling



Flow of a TCP/IP Message
TCP/IP Inbound Message Processing – OSA Polling



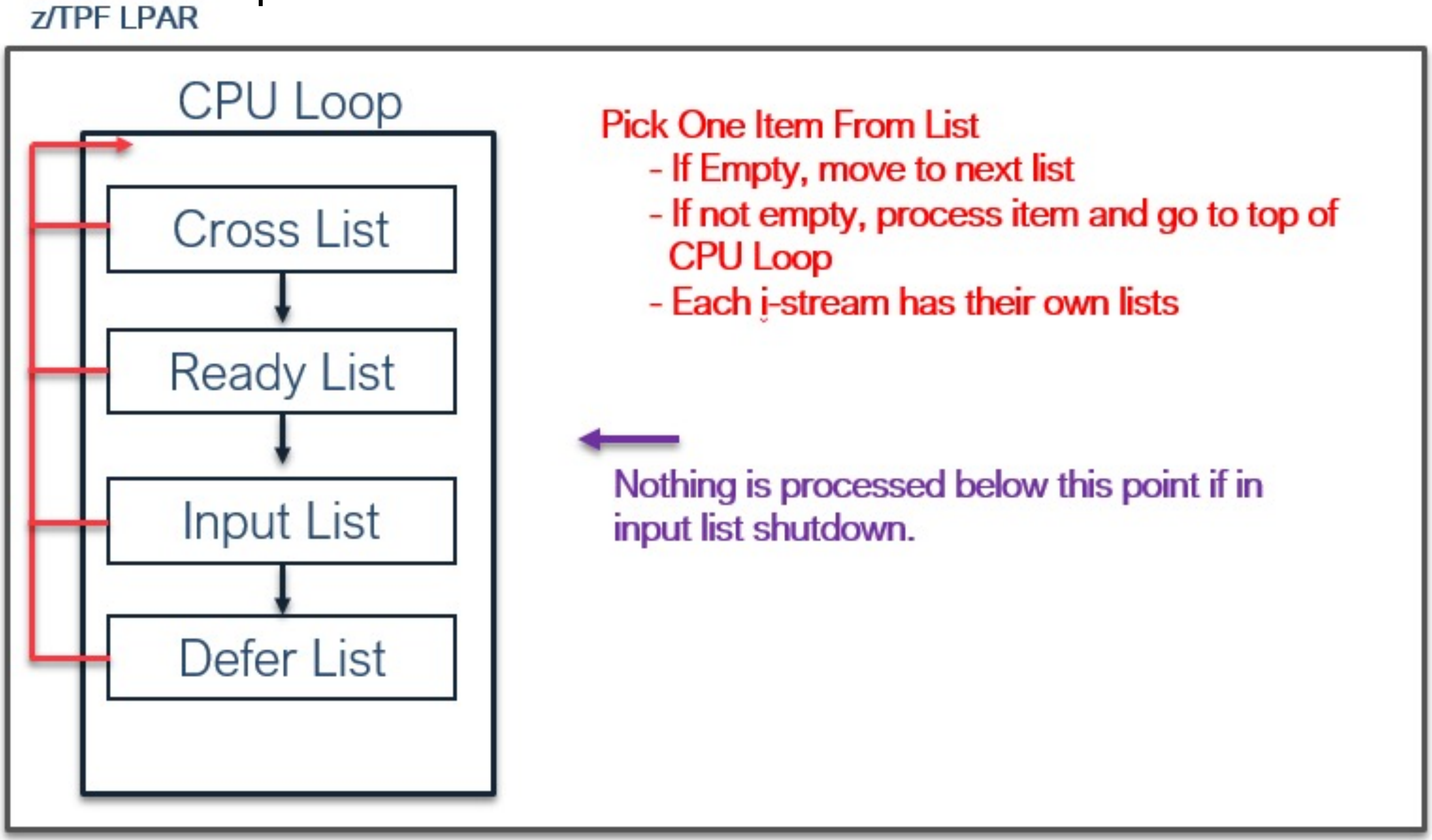
Flow of a TCP/IP Message
OSA Polling Details

1. Packet is pulled from OSA read buffer and copied into an IPMT block
2. z/TPF IP trace is called to trace the inbound packet
3. IPMT block containing inbound packet is placed on the OSA input or OSA ready list
   a. List used depends on the input priority defined in network services database or socket
4. Go get next packet from OSA read buffer – continues until ALL packets available at the time are processed

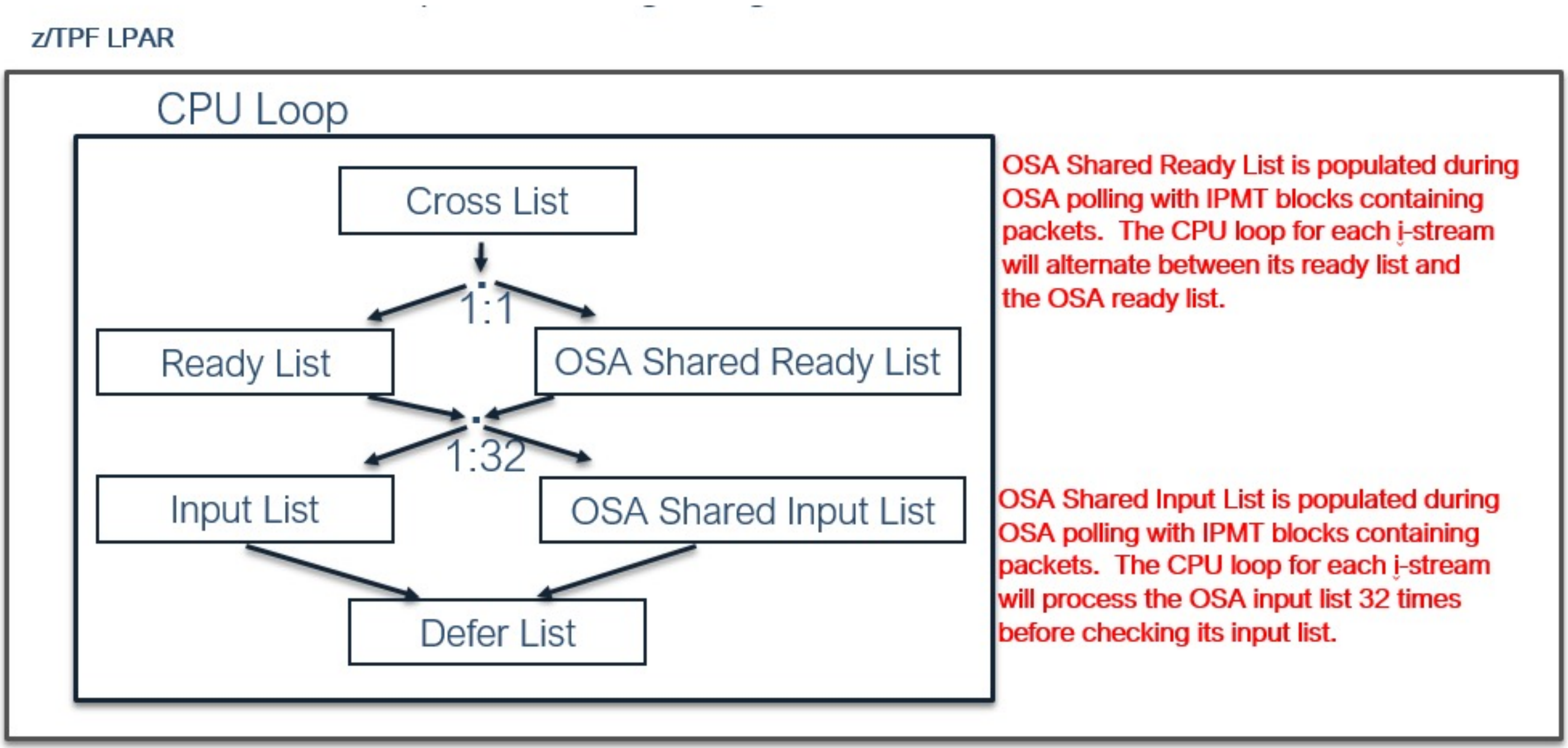
OSA Staging Queue: A staging area for packets to be introduced to the system at a later point in time

- Regular priority and in input list shutdown
- OSA polling called from external interrupt processing or system error processing

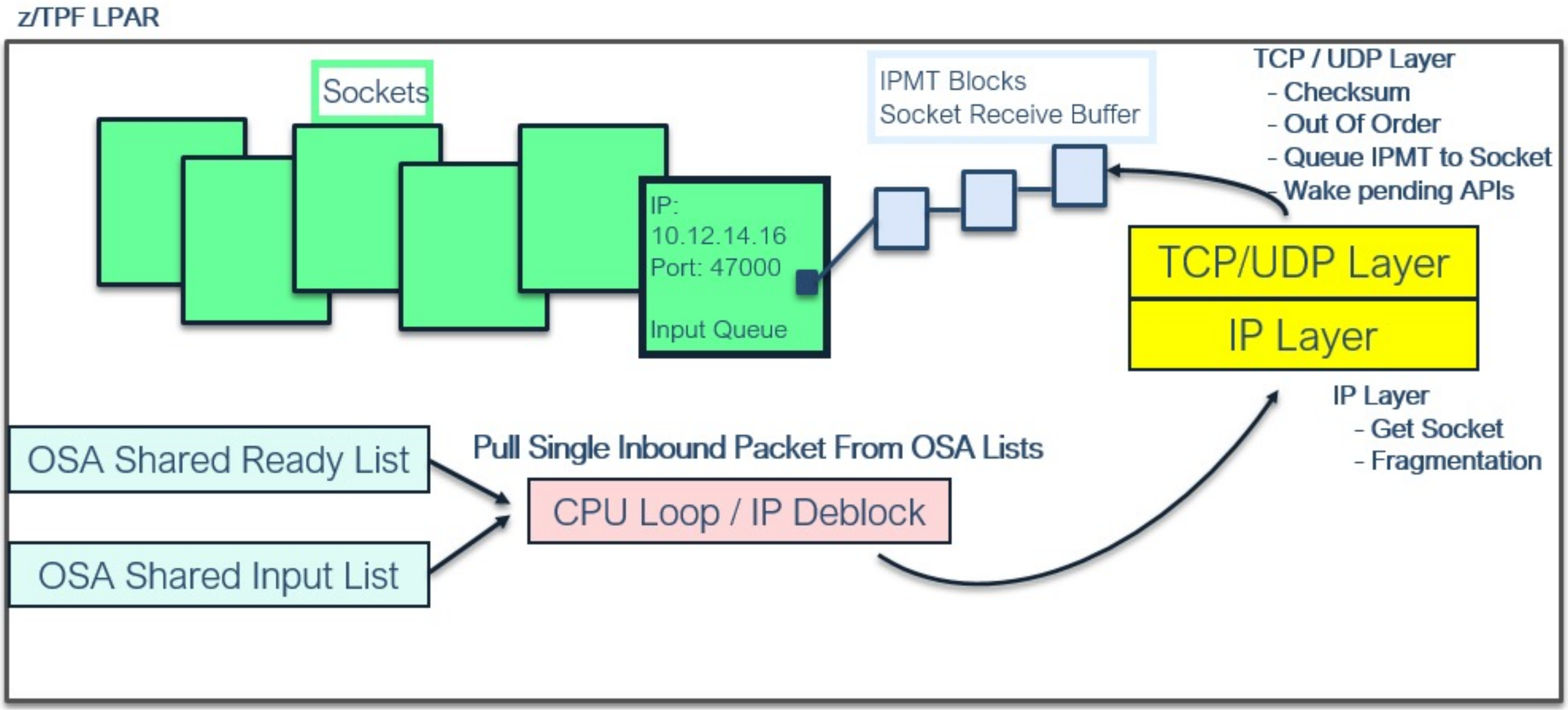
Flow of a TCP/IP Message
High Level CPU Loop



Flow of a TCP/IP Message
High Level CPU Loop – With OSA



Flow of a TCP/IP Message
Receive Data From a Socket



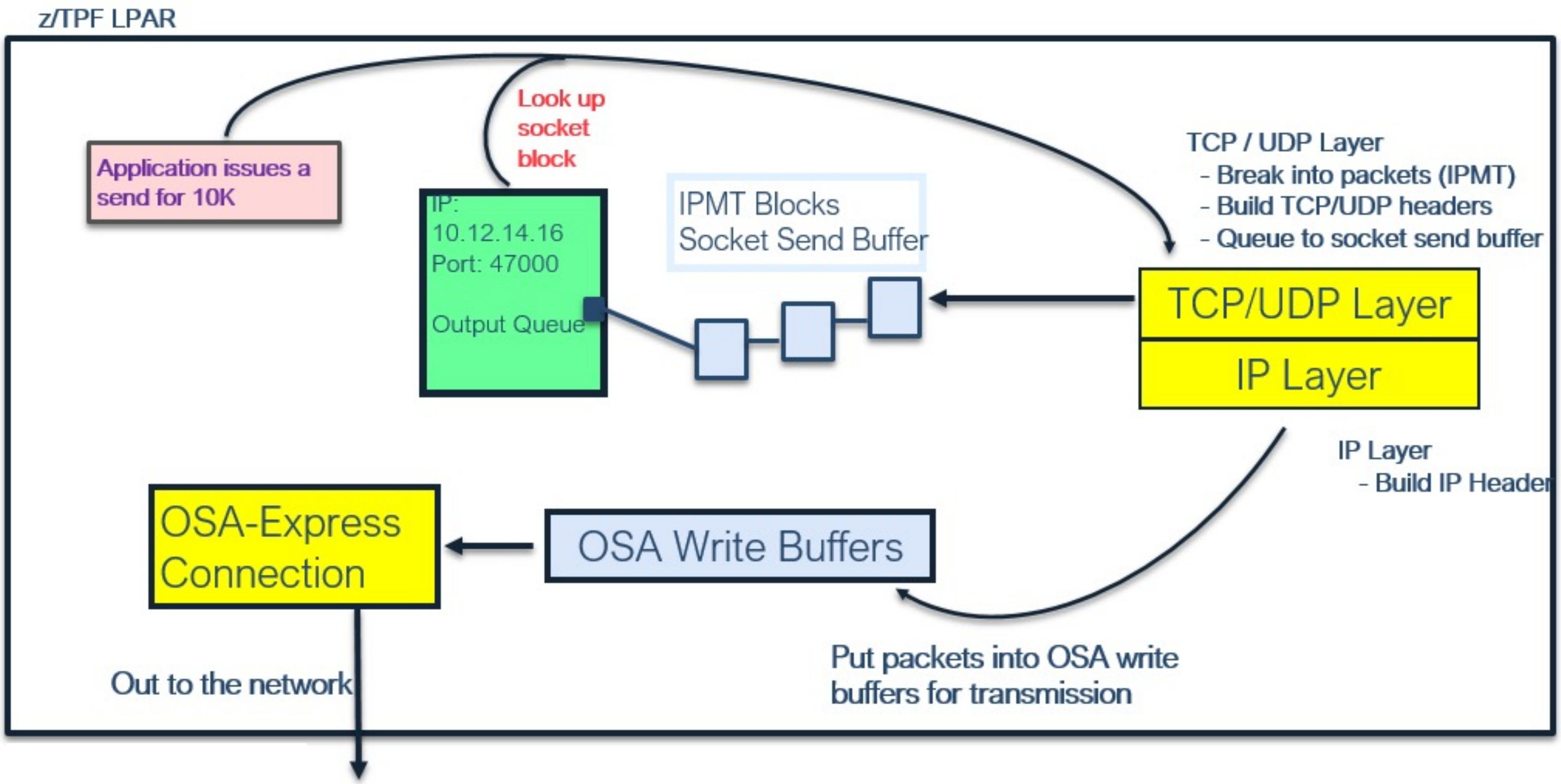
Flow of a TCP/IP Message

Receive Data From a Socket Details

- TCP/IP protocol is a PULL model from application
  - Data is queued to the socket, but is not presented to the application until it is asked for
    - » TCP read-type API (read, recv, recvfrom, activate_on_receipt, etc)

- Application may need to issue more than a single read to read the entire “application message”

Flow of a TCP/IP Message
Sending Data On a Socket



Flow of a TCP/IP Message
Sending Data On a Socket Details

- Once an application has read an inbound message, generally it needs to send a reply (not always)
- Sending data is less complicated as CPU loop processing and input list shutdown checks are not needed.

- Let's say an application sends a 10K reply
  1. Issue send type API to send the TCP/IP message (send, write, etc)
  2. TCP/IP stack breaks the 10K message into IP packets (in IPMTs) based on the maximum packet size
  3. For each packet :
     a. Builds the IP and TCP/UDP headers
     b. Places outbound packet in the OSA write buffer
        (Updates OSA write buffer to point to an IPMT block)
     c. Calls IP trace to trace the outbound packet

Backup Slides

Agenda

- TCP/IP Overview

TCP/IP Overview
General TCP/IP Information

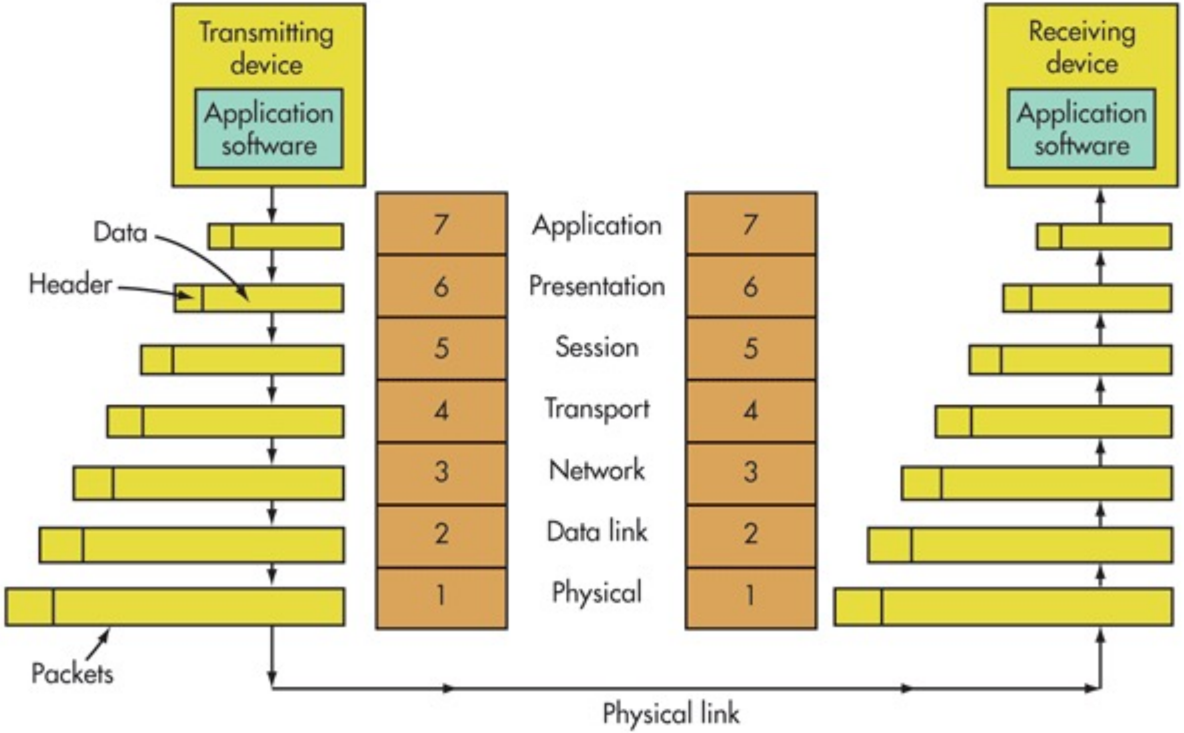
- TCP/IP – Transmission Control Protocol / Internet Protocol
  - A suite of communication protocols used to interconnect network devices on the internet
  - A set of rules and procedures defining how data is exchanged

- TCP/IP consists of:
  - Transport Layer
    - TCP/UDP: Creates channels of communication between hosts connected to the network
  - Network Layer:
    - IP: Defines how to route data to ensure it reaches the correct destination

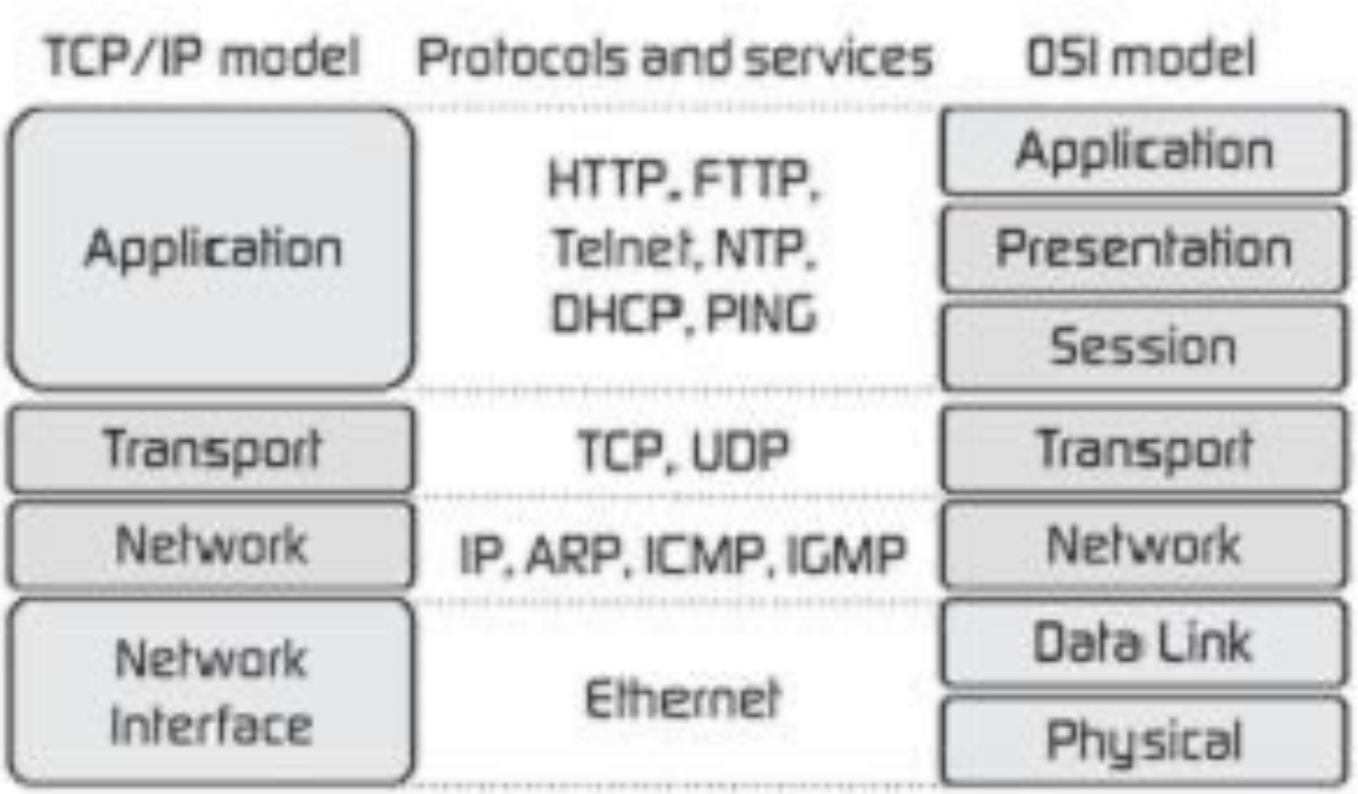
- TCP/IP uses a client/server model of communication where a machine (or client) requests data from another machine (or server)
  - For example, a client machine requests a web page from a server machine

"

TCP/IP Overview
Open Systems Interconnection (OSI) Model

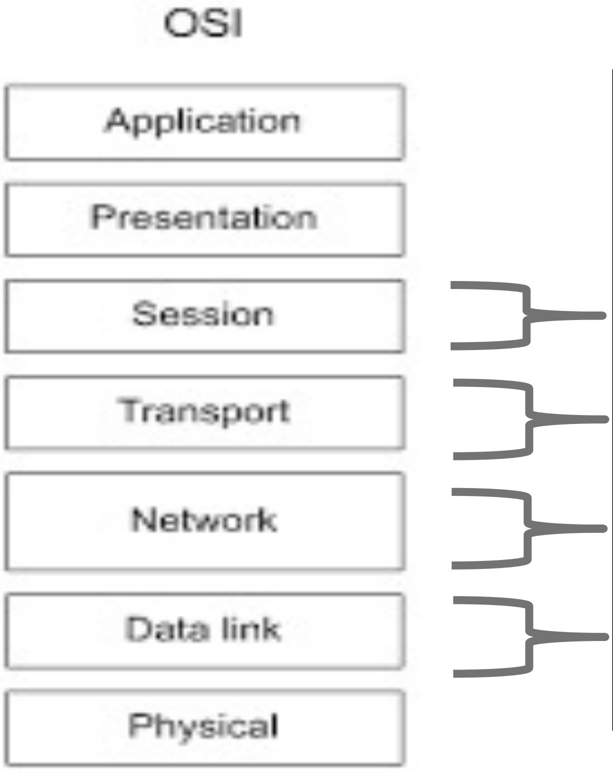


TCP/IP Overview
Where does TCP/IP Fit in the OSI Model?



TCP/IP Overview

What parts of the OSI model will we be looking at?



z/TPF Equivalent

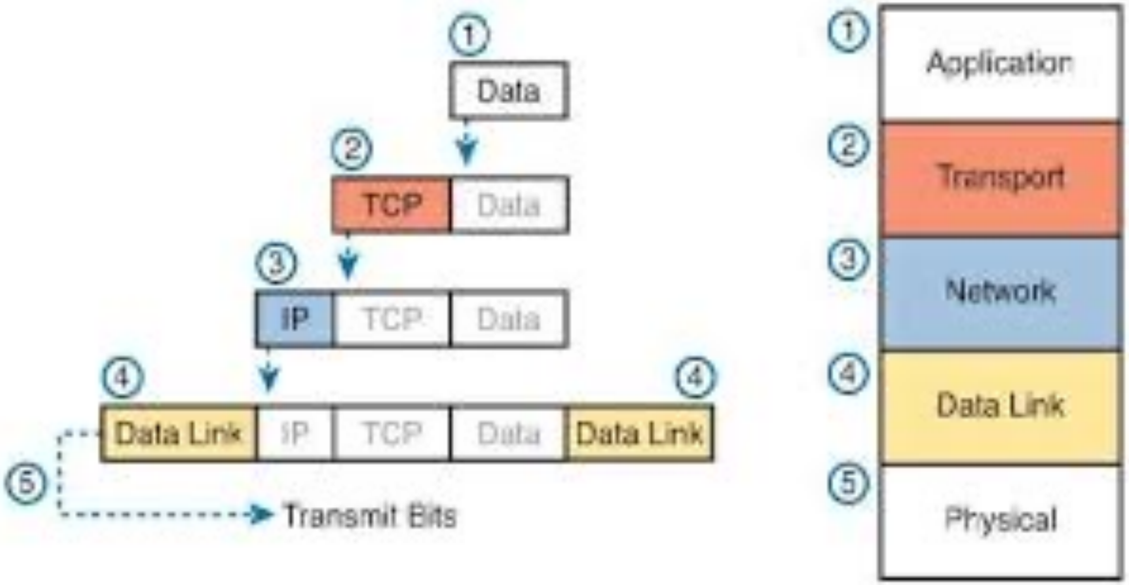
Socket Applications (ie. SSL, FTP, HTTP)

TCP/UDP

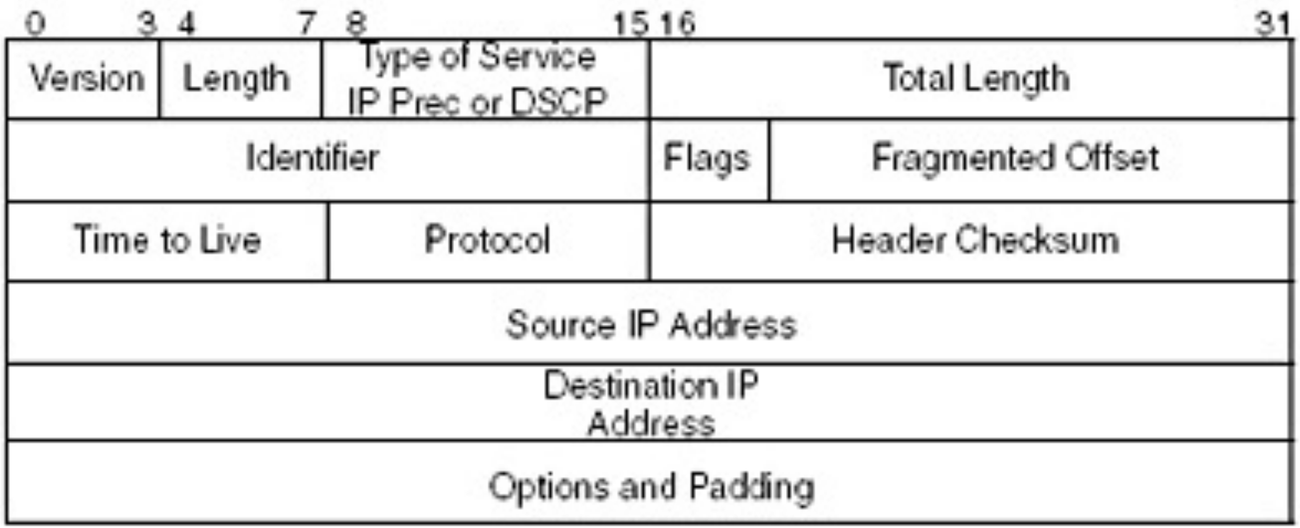
IP

OSA

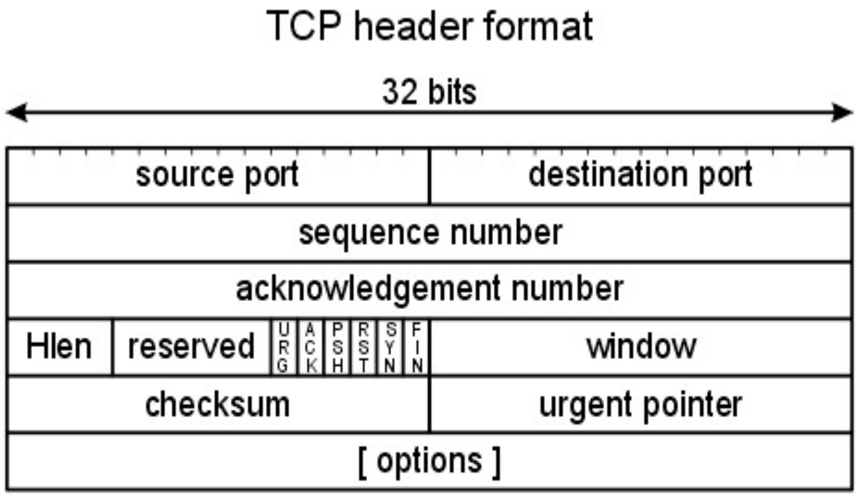
TCP/IP Overview
Each layer in TCP/IP consists of a header

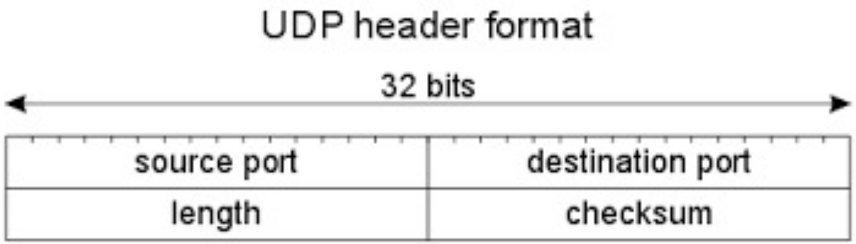


TCP/IP Overview
IP Header



TCP/IP Overview
TCP/UDP Headers





TCP/IP Overview
What is a socket?

- A "socket" is one end-point in a two-way communication link between programs running on the network.
  - It is referenced by applications using socket descriptor which is just a token assigned to the socket.
  - It can also be found and referenced using the following
    - Local IP Address
    - Local Port Number
    - Remote IP Address
    - Remote Port Number
    - Protocol

Two sockets cannot have this same information

- Socket APIs exist to manage the exchange of data between client and server applications.

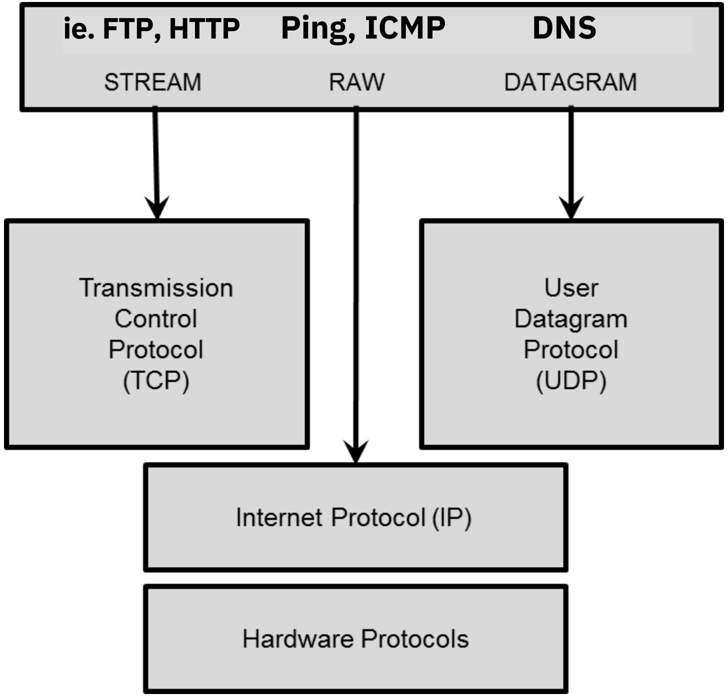
TCP/IP Overview
Types of Sockets

STREAM -- TCP Protocol
- Data is delivered to and retrieved from the buffer as a stream of user data bytes.
- No internal definition of a "message".
- Local socket has a one-to-one association with a remote socket – connection oriented.

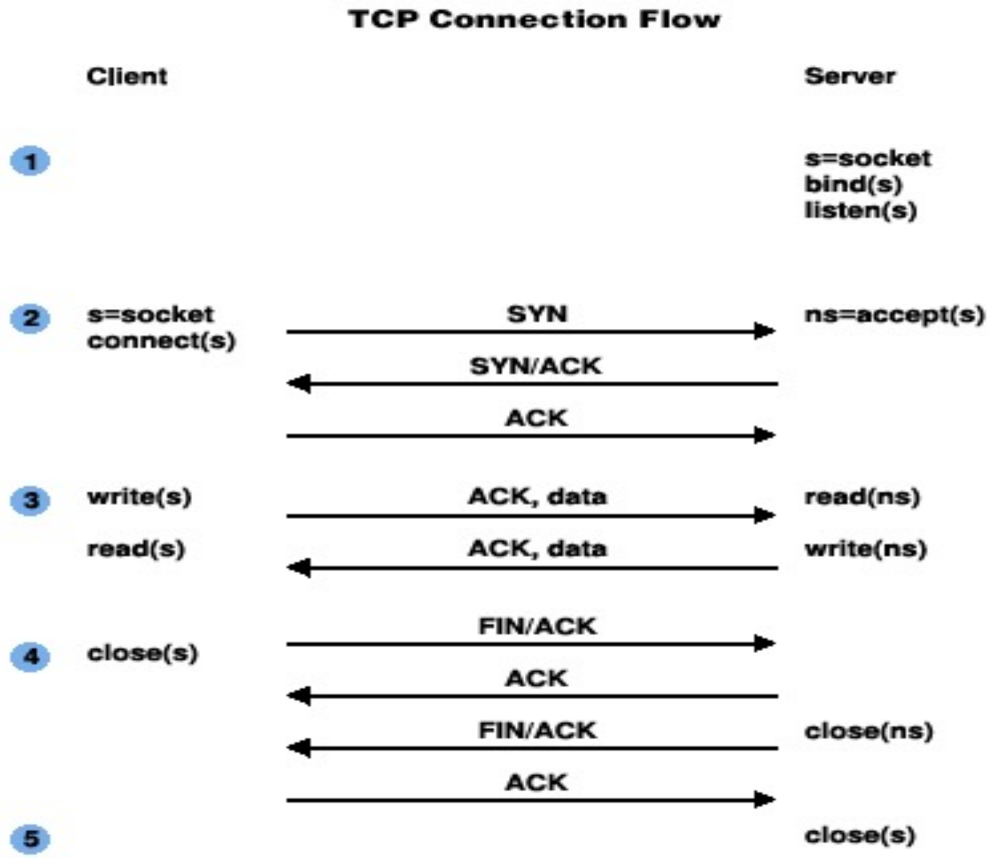
DATAGRAM -- UDP Protocol
- Data is delivered to and retrieved from the buffer as an unfragmented datagram without protocol headers (user data only).
- Local socket may have a one-to-many association with remote sockets.
- No guarantee of delivery, ordering or duplicate protection you get with TCP
- UDP datagram is broken into fragments when sending and are not presented to the receiving application until all the fragments are re-assembled by the remote TCP/IP stack – full datagram

RAW
- Data is delivered to and retrieved from the buffer as an unfragmented datagram with protocol headers
- IP, IGMP, ICMP and RAW may be protocols used
- All data from a RAW-related protocol may be delivered to all RAW Socket Types.
- Intended for utility and system use only.

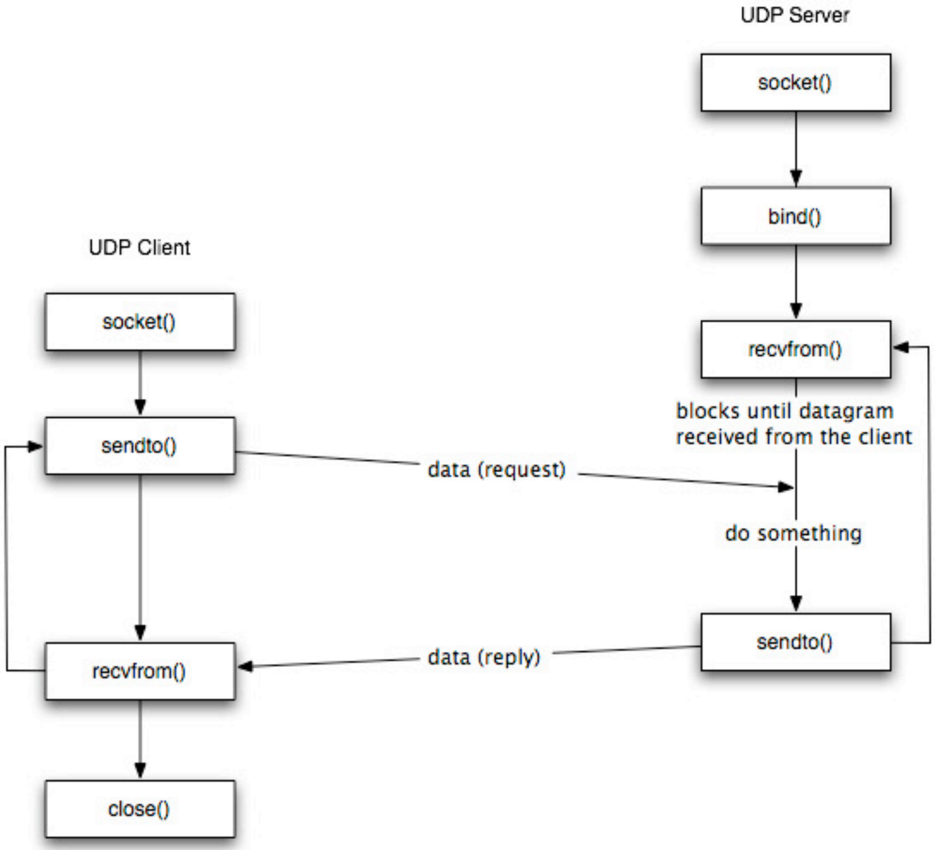
TCP/IP Overview
Socket Types -> TCP/IP Protocols



TCP/IP Overview
TCP Communication Flow



TCP/IP Overview
UDP Communication Flow



Agenda

- TCP/IP Diagnostics and Miscellaneous

TCP/IP Diagnostics and Miscellaneous
TCP/IP System-Wide IP Trace

- IP Datagrams can be traced for the entire network, a particular IP address or a particular OSA.
  - IP Trace data is written to an 80K memory resident table.
    - Optionally write the IP trace data to tape.

- ZTTCP TRACE is used to start/stop trace and specify maximum datagram data traced.

- ZIPTR displays the in memory trace table
  - Compacted or formatted
  - EBCDIC (default) or ASCII translation for the data portion of a formatted trace.

- The offline IP Trace program
  - Reads the IP Trace Data from a tape or set of tapes.
  - EBCDIC (default) or ASCII translation for the data portion of a formatted trace.
  - User specified selection criteria (remote/local IP address, remote/local port number, protocol, and most importantly time)
    - Allows generating reports on subsets of all traced packets.

TCP/IP Diagnostics and Miscellaneous
TCP/IP Compact IP Trace Example

```
User:   ZIPTR 5

  System: IPTR0001I 12.41.57 IP TRACE TABLE
          RW IN    SOURCE IP        DEST IP     SPORT DPORT PR FG DATA
          31 02   9.117.249.058   9.117.249.056  9999  1025 06 18 4040F1F0
          32 02   9.117.249.056   9.117.249.058  1025  9999 06 18 4040F1F0F08181
          51 02   9.117.249.058   9.117.249.056  9999  1025 06 18 4040F1F0
          52 02   9.117.249.056   9.117.249.058  1025  9999 06 18 4040F1F0F08181
          31 02   9.117.249.058   9.117.249.056  9999  1025 06 18 4040F1F0
          639 ENTRIES IN IP TRACE TABLE+
```

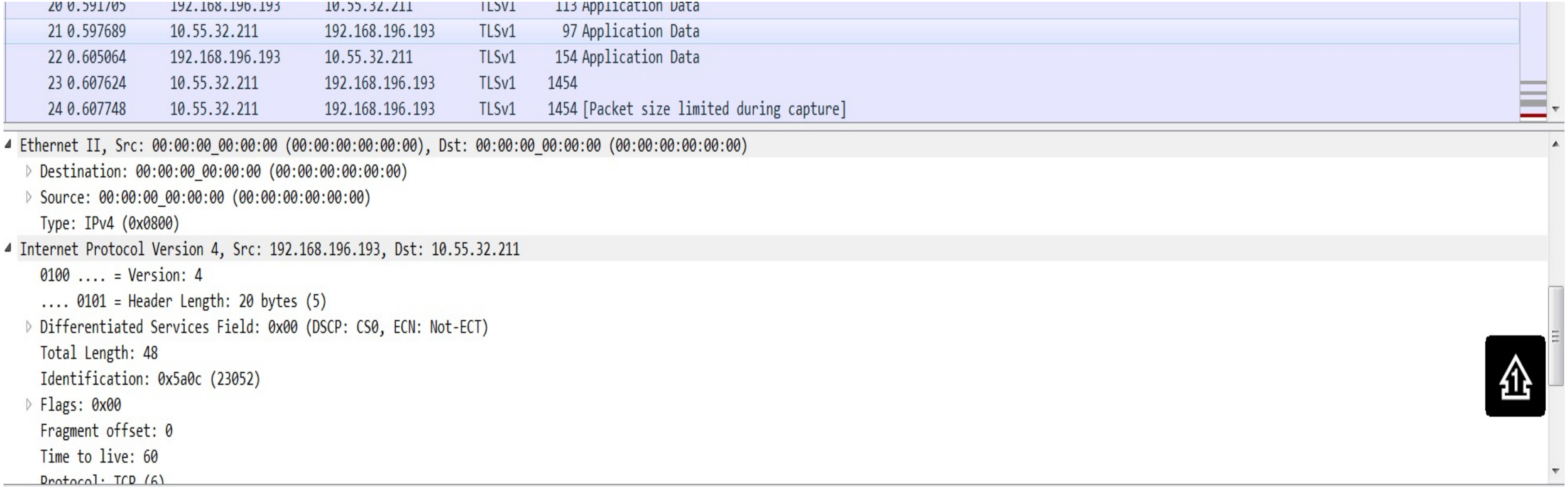
TCP/IP Diagnostics and Miscellaneous
TCP/IP Formatted IP Trace Example

```
User:   ZIPTR 4 FORMAT

IPTR0002I 11.43.28 IP FORMATTED TRACE
RWI-32  IPCCW-02  SOURCE IP-9.117.249.72  DEST IP-9.117.249.73  LEN-48
TOD-B23D05C3CD79FC06  PROTOCOL-06 (TCP)  SOURCE PORT-1025  DEST PORT-9999
SEQ-1547432521  WINDOW-32767  URGENT OFFSET-0
TCP FLAG BYTE-02 (SYN)
IP HEADER   45000030 C5694000 3B0674E2 0975F948 0975F949
TCP HEADER  0401270F 5C3BF249 00000000 70027FFF 7BBE0000 02040F00 01030304
RWI-31  IPCCW-02  SOURCE IP-9.117.249.73  DEST IP-9.117.249.72  LEN-48
TOD-B23D05C3CE4B0406  PROTOCOL-06 (TCP)  SOURCE PORT-9999  DEST PORT-1025
SEQ-1547491231  ACK-1547432522  WINDOW-2047  URGENT OFFSET-0
TCP FLAG BYTE-12 (ACK, SYN)
IP HEADER   45000030 C56A4000 3C0673E1 0975F949 0975F948
TCP HEADER  270F0401 5C3CD79F 5C3BF24A 701207FF BFD10000 02040F00 01030304
RWI-52  IPCCW-02  SOURCE IP-9.117.249.72  DEST IP-9.117.249.73  LEN-40
TOD-B23D05C3D546D004  PROTOCOL-06 (TCP)  SOURCE PORT-1025  DEST PORT-9999
SEQ-1547432522  ACK-1547491232  WINDOW-2047  URGENT OFFSET-0
TCP FLAG BYTE-10 (ACK)
IP HEADER   45000028 C56B4000 3B0674E8 0975F948 0975F949
TCP HEADER  0401270F 5C3BF24A 5C3CD7A0 501007FF 66B50000
120 ENTRIES IN IP TRACE TABLE
```

TCP/IP Diagnostics and Miscellaneous
Viewing System-Wide IP Trace in Standard Format (Wireshark)

- IP Trace reports and displays (offline and online) is a z/TPF unique format.
- An option exists on the offline IP trace to format the IP trace data to allow it to be consumable by open tooling (like wireshark)



TCP/IP Diagnostics and Miscellaneous
TCP/IP Individual IP Trace

- An **in-core only** trace facility that can be used for debugging and analysis
- A user can tailor the in-core trace to a specific remote IP or TCP/UDP port

- Use must configure in CTK2 how many individual IP traces that can be defined and the size of them.
  - IPTRCNUM - maximum number of concurrent individual IP traces
  - IPTRCSIZ - number of pages for each trace

- ZINIP command
  - Define named trace for remote IP address and/or local port number
  - Display trace information
  - Other useful options
    - NOWRAP to see beginning flows of a connection only
    - PAUSE/RESUME to temporarily stop/start tracing

TCP/IP Diagnostics and Miscellaneous

TCP/IP Individual IP Trace Example

```
zinip def name-myTrace port-9999
CSMP0097I 09.13.41 CPU-B SS-BSS  SSU-HPN  IS-01
INIP0001I 09.13.41 INDIVIDUAL IP TRACE MYTRACE DEFINED+

zinip disp format name-mytrace all
CSMP0097I 09.15.44 CPU-B SS-BSS  SSU-HPN  IS-01 _
INIP0007I 09.15.44 INDIVIDUAL IP FORMATTED TRACE MYTRACE DISPLAY
RWI-02  IPCCW-D1  SOURCE IP-9.57.13.51  DEST IP-9.57.13.50  LEN-48
  TOD-D51040B7F7FDF110  PROTOCOL-06 (TCP)  SOURCE PORT-49170  DEST PORT-9999
  SEQ-192862893  WINDOW-65535  URGENT OFFSET-0 _
  TCP FLAG BYTE-02 (SYN)
  IP HEADER   45000030 70600000 3C06E191 09390D33 09390D32
  TCP HEADER  C012270F 0B7EDAAD 00000000 7002FFFF 89FF0000 020405AC 01030304
RWI-01  IPCCW-D1  SOURCE IP-9.57.13.50  DEST IP-9.57.13.51  LEN-48
  TOD-D51040B7F7FDF110  PROTOCOL-06 (TCP)  SOURCE PORT-9999  DEST PORT-49170
  SEQ-192905183  ACK-192862894  WINDOW-65535  URGENT OFFSET-0
  TCP FLAG BYTE-12 (ACK, SYN)
  IP HEADER   45000030 C6310000 3C068BC0 09390D32 09390D33
  TCP HEADER  270FC012 0B7F7FDF 0B7EDAAE 7012FFFF FE8F0000 020405AC 01030304
```

TCP/IP Diagnostics and Miscellaneous
Socket API Trace

Trace at a per-socket level
- Useful because multiple ECBs can share a socket
- Display online via ZSOCK API command
- Trace table resides in each socket block entry
  - Last 3K of the socket block entry

Trace at a per-ECB level
- Useful for debugging socket application programs
- Included and formatted in ECB dumps
- Trace table resides in area pointed to by the ECB

ZSTRC ALTER SOCTRACE / NOSOCTRACE to Enable / Disable the Socket API Trace

TCP/IP Diagnostics and Miscellaneous
Socket API Trace Entry Details

Each entry includes:

- API Input parameters
  - Includes implied parameters like time out values
  - Parameter data displayed in human readable format. Examples:
    protocol=TCP, port=5004, IP=10.2.56.8
- Output, including the API return code
  - If error return code, error value is displayed (like SOCTIMEDOUT)
- How long it took the API to be completed

If ECB becomes blocked (event-waited) during API processing, two trace entries are created:

- Right before the ECB is suspended (blocked)
  - Entry does not contain return code or completion time
- Right before returning to the application program
  - Entry does not contain API input data (would be same as above)
  - Completion time includes the time that the ECB was blocked

TCP/IP Diagnostics and Miscellaneous
Compact Socket API Trace Example

```
ZSOCK TRACE SOCK-C000002
SOCK0035I 15.26.53 BEGIN SOCKET TRACE FOR 00C00002
                                                   COMPLETION
ECB       API        RC                PROGRAM  TIME (SEC)  TIME STAMP
07650000  socket     00C00002          QZZQ          0.003  May 15 15:25:39
07650000  bind       0                 QZZQ          0.011  May 15 15:25:39
07650000  listen     0                 QZZQ          0.002  May 15 15:25:39

07650000  accept                       QZZQ                 May 15 15:25:39
07650000  accept     00C00008          QZZQ         13.634  May 15 15:25:52

07650000  accept                       QZZQ                 May 15 15:25:52
07650000  accept     00C00013          QZZQ          5.213  May 15 15:25:57

07650000  accept     00C00014          QZZQ          0.016  May 15 14:25:57

07650000  accept                       QZZQ                 May 15 15:25:57
END OF DISPLAY
```

TCP/IP Diagnostics and Miscellaneous
Formatted Socket API Trace Example

```
zsock trace format sock-c00002
CSMP0097I 15.26.48 CPU-B SS-BSS  SSU-HPN  IS-01
SOCK0038I 15.26.48 BEGIN FORMATTED SOCKET TRACE FOR SOCKET 00C00002
ECB-07650000  API-socket       PROG-QZZQ  OFFSET-000058  IS-02  May 15 15:25:39
  type-SOCK_STREAM  prot-IPPROTO_TCP
  RC-00C00002         COMPLTIME-0.003sec

ECB-07650000  API-bind         PROG-QZZQ  OFFSET-000148  IS-02  May 15 15:25:39
  port-5004  ip-9.117.241.1  addrlen-16
  RC-0                COMPLTIME-0.011sec

ECB-07650000  API-listen       PROG-QZZQ  OFFSET-000334  IS-02  May 15 15:25:39
  backlog-15
  RC-0                COMPLTIME-0.002sec

ECB-07650000  API-accept       PROG-QZZQ  OFFSET-000520  IS-02  May 15 15:25:39
  addrlen-16  timeout-0
  BLOCKED

ECB-07650000  API-accept       PROG-QZZQ  OFFSET-000520  IS-02  May 15 15:25:52
  port-1027  ip-9.117.232.167
  RC-00C00008         COMPLTIME-13.634sec
```

TCP/IP Diagnostics and Miscellaneous
Socket Monitor User Exit

When special condition occurs for a socket, USMO user exit is invoked with socket descriptor, condition type, and pertinent socket information passed as input.

Conditions include:
- Output messages for a socket waiting to be sent for more than 30 seconds because the TCP window has been closed for that period of time.
- Input messages for a socket queued for 10 seconds without any application reading them.
- A TCP connection request is received but the server is at its backlog limit.
- z/TPF set a TCP window size of 0 because the receive buffer on the socket is full.
- The number of ECBs queued while issuing a **send** on the same socket has crossed one of the four threshold levels (10, 25, 50, or 100).

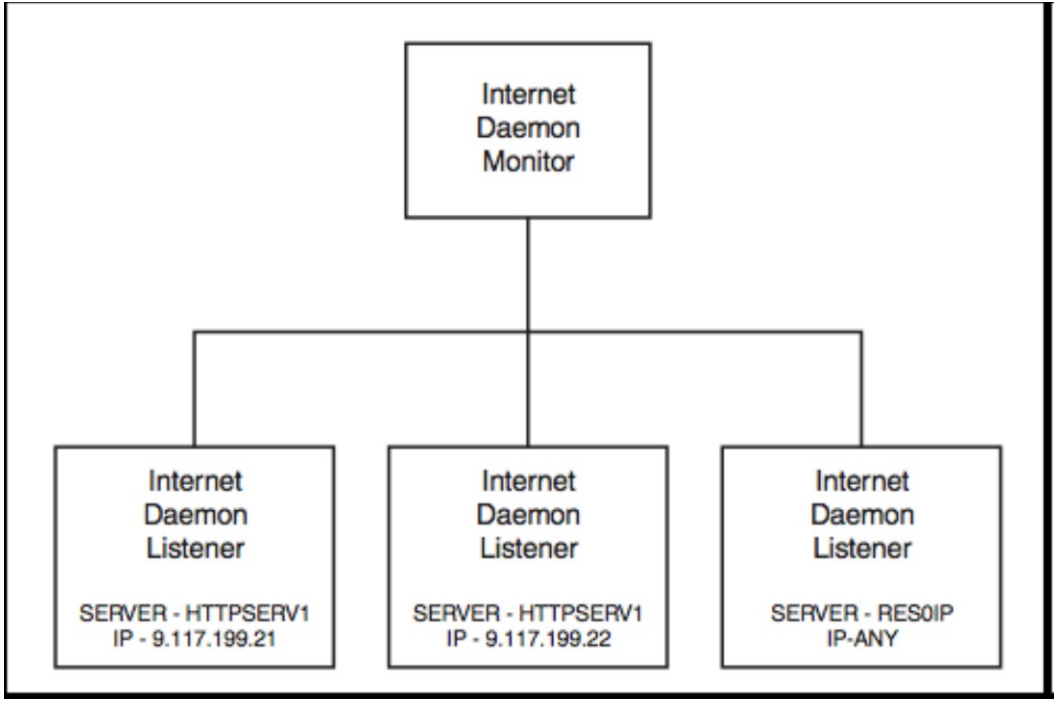
Monitoring is enabled/disabled with ZNKEY SOCKMON-YES|NO.

Default system setting disables monitoring.

TCP/IP Diagnostics and Miscellaneous
The z/TPF Internet Daemon

- To define TCP/IP listeners for a Internet Server applications the recommendation is to use the z/TPF Internet Daemon (INETD)
- The Internet Daemon is responsible for:
  - Starting/Stopping Internet Daemon
    - Including cycle-up and cycle-down
  - Handles errors and automatically recovers when an Internet Daemon listener fails.
- The Internet Daemon provides different models
  - Which model to use depends on the application.

TCP/IP Diagnostics and Miscellaneous
The z/TPF Internet Daemon Model



TCP/IP Diagnostics and Miscellaneous
The Standard Internet Daemon Models

Models similar to other platforms like Unix and Linux.

- WAIT model
  - INETD creates and manages the listener socket.
  - INETD creates new child process to handle new connection (TCP) or data (UDP).
  - INETD will wait for a child to complete processing before handling next connection or inbound data message

- NOWAIT model
  - INETD creates and manages the listener socket.
  - INETD creates multiple child processes to handle connections or data.

- DAEMON model
  - Listener creates single child process to start server application.
  - Child process creates and manages the listener socket.
  - Child process also handles new connections (TCP) or data (UDP).

TCP/IP Diagnostics and Miscellaneous
The z/TPF Unique Internet Daemon Models

Models that are unique to z/TPF

- AOR model
  - INETD creates and manages the listener socket.
  - When a client connects, INETD issues an activate_on_receipt to read the message in a new ECB
    - The application is required to issue subsequent AOR

- AOA/AOA2 model
  - INETD creates and manages the listener socket.
  - INETD uses activate_on_accept to get new connections in a new ECB.
    - Upon receipt of the connection, the INETD enters the application program
  - Difference between AOA/AOA2 is
    - AOA: Application required to issue subsequent AOA
    - AOA2: INETD handles issuing subsequent AOA on the application behalf.

- SSL model
  - INETD created and manages the listener socket along with the SSL CTX
  - When an SSL application connects, the SSL session is passed to the application in a new ECB
    - Uses AOA under the covers

Trademarks

IBM, the IBM logo, ibm.com and Rational are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Notes

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.