TCP/IP Native Stack Roadmap
for TPF 4.1

Mark Gambino
It Came from the 80's

- TCP/IP has become the most prevalent networking protocol
- The IP and TCP architectures were developed many years ago:
  - Request for Comments (RFC) 791 - Internet Protocol (IP)
  - RFC 793 - Transmission Control Protocol (TCP)
  - Both RFC documents came out in 1981
- The RFCs describe the bits, bytes, and behavior of exchanging data
- "Some Assembly Required"
  - Standards for robust communication features like security, 24x7 availability, load balancing, and flow control have been slow in coming, slow in adoption, or purely roll your own (RYO)
- Networking speeds and principles have changed significantly since 1981
  - Some of the limits, algorithms, and timer values of the original architecture do not work very well at Gigabit Ethernet (GbE) speeds
  - Need larger window and buffer sizes
  - Need more elaborate control mechanisms.
The Evolution of a Very Robust TCP/IP Stack

- Supporting the base requirements of the IP and TCP architectures is a start
- Need much more to run and manage a large server environment with real-time transactions
- TPF TCP/IP native stack support came out on PUT 11
- Over 60 enhancements to TPF in the TCP/IP area since then:
  - Different connectivity options
  - Improved availability
  - Performance improvements
  - Many diagnostic tools to enhance usability and operations
  - Various load balancing options
  - Support for extended socket APIs and socket options
  - Numerous security measures
  - Interface with standard network management platforms
Connectivity
Insert Cable A in Slot B

- Support 374x IP routers (PJ26683, PUT 11)
  - Can use same routers for both SNA and IP traffic
  - Uses IP over channel data link control (CDLC), which is very similar to SNA over CDLC
- Gigabit Ethernet (GbE) OSA-Express support (PJ27333, PUT 13):
  - Extremely high-bandwidth pipe
  - Performance and cost benefits over channel-attached router solutions
  - Can greatly simplify your network configuration
- Fast Ethernet OSA-Express support (PJ27625, PUT 14)
  - Useful for test systems or lower-volume TPF production systems.
Connect the Dots
Availability
Pick a Path, Any Path

- Ability to map one TPF IP address across multiple 374x routers (PJ26683, PUT 11)
  - Eliminates single point of failure in 374x environment
- IP routing table support (PJ26890, PUT 12):
  - Allows you to define which path (local IP address) to use to reach a given remote network for TPF client applications
  - Allows you to define multiple paths for TPF to use to reach a given remote network
- IP routing table support for OSA-Express gateways (PJ27333, PUT 13):
  - Allows you to define which path (first hop IP router) to use to reach a given remote network
  - Allows you to define multiple paths (more than just the two default gateways) for TPF to use to reach a given remote network.
Is It Real, or Is It Virtual?

- Virtual IP Address (VIPA) support (PJ27333, PUT 13):
  - A local IP address (VIPA) can swing from one OSA-Express adapter to another
  - Sockets bound to that VIPA move as well
  - Enables sockets to survive failure of an OSA-Express adapter, GbE switch, or first hop router (for Fast Ethernet)
    - Swing is automatic and transparent to applications

- Ability to automatically swing VIPAs when all default gateways fail (PJ28584, PUT 17):
  - Useful if you have only one default gateway, which is the case if you are using a combination layer 2/3 switch/router
  - Not needed if you have multiple first hop routers (defined via the IP routing table) or are sending traffic over the local network only.
Move It or Lose It

- Ability to move a VIPA from one TPF processor to another in the complex (PJ27333, PUT 13):
  - Enables an IP address (VIPA) to always be active (assuming at least one TPF processor is up and connected to the network)
  - Move is automatic (via user exit) when a processor is taken out of the complex
  - Move can be operator initiated as well
- Ability to move a VIPA with force (PJ28591, PUT 17):
  - Allows a VIPA to be moved to another processor even if the processor that owns the VIPA is IPLing or is not responding.
Sample Configuration to Survive Component Failures

TPFA

2.1.1.2 3.1.1.2

OSA OSA

GbE Switch GbE Switch

IP Router IP Router IP Router

TPFB

4.4.4.4

2.1.1.1 3.1.1.1

OSA OSA

IP Router

374x 374x 374x
Performance
Verify the Data... and Be Quick About It!

- Hardware Checksum Support (PJ26842, PUT 12):
  - Calculating the TCP or UDP checksum requires math operations on every byte of user data in the packet
  - Requires several hundred instructions for large packets, even if you use a highly optimized software routine
  - IBM G4 processor series introduced a new CKSM instruction that enables checksums to be calculated by hardware rather than by software
  - TPF automatically determines whether the processor supports the CKSM instruction
    - If supported, TPF uses hardware to perform checksum operations
    - If not supported, TPF uses software to perform checksum operations.
Why Ask the Same Question Over and Over?

- Cache information learned from remote Domain Name System (DNS) servers (PJ27268, PUT 13):
  - When TPF client applications want to connect to a remote host, DNS is used to resolve the remote host name (like iluvchocolate.com) to its IP address
  - TPF sends the request to a remote DNS server
  - TPF now caches the response from the DNS server in memory
  - Subsequent requests to resolve this host name use the cache rather than asking an external DNS server the same question again
  - If the response from DNS server contains multiple IP addresses, the order of IP addresses presented to the application (output of the gethostbyname API) is changed (round-robin fashion) each time the cache entry is accessed
  - Information is cached only if the response is marked as allowed to be cached, and cached only for the amount of time specified in the response.
Traveling First Class versus Economy Class

- IP (RFC 791) defined a type of service (TOS) field in the IP header
  - Use of this field was not defined/standardized at that time
  - Result was that all traffic flowed at the same priority
- Differentiated Services (DiffServ)
  - Defined by RFCs 2474 and 2475
  - Redefines the TOS field as the DiffServ code point field
  - Enables packets to flow at different priorities in the network
  - Quality of service (QoS) architecture similar in concept to SNA class of service (COS)
- Enabling Differentiated Services on TPF (PJ28034, PUT 15)
  - Allows you to define a default DiffServ code point value to be used for output packets built by TPF
- Enhanced Differentiated Services support (PJ28195, PUT 16)
  - Allows you to define a different DiffServ code point value for each application.
Queued direct I/O (QDIO) is used to exchange data between TPF and the OSA-Express adapter

- QDIO is a memory sharing model and is not interrupt driven
- If host does not poll quickly enough, read buffers can become full and input packets from the network can be discarded
- OSA polling enhancements (PJ28064, PUT 15):
  - Ensures that TPF polls OSA-Express connections more consistently
    - Particularly important for heavy traffic on TPF test systems running under VM or TPF systems running under shared PR/SM where TPF is not being dispatched on a timely basis
    - Single I-stream systems are prime candidates for this support
  - Can define more OSA read buffers (from 16 to as many as 64)
  - When running native (not under LPAR) or running dedicated PR/SM, TPF when idle now looks for new input message from OSA rather than waiting for an external event (like an interrupt) to wake up TPF and trigger OSA polling
    - Improves overall response time and reduces the "bursting" effect.
I'm Lost - Help Find Me!

- When a message is sent on a TCP socket, the remote node sends an acknowledgment (ACK) to indicate that the data was received by the remote node.
- If data is lost in the network, no ACK will flow and the sending side will retransmit the data.
- How long to wait before retransmitting the lost data is based on the smoothed round-trip time (SRTT) of the socket.
- On high-volume TCP sockets, the loss of a single packet can significantly reduce throughput.
- TCP fast retransmit support (PJ28344, PUT 16):
  - Enables the sending node to detect more quickly that data has been lost (if multiple packets are in flight).
  - Sending node retransmits the lost data immediately upon receipt of duplicate ACKs rather than waiting for timeout to trigger the retransmit.
TCP (RFC 793) defines flow control between the two endpoints
- Fields in the TCP header advertise a window, which is how much data the remote end is allowed to send to you
  - Based on amount of available space in the socket's receive buffer
  - Does not take into account network conditions
  - No equivalent of SNA node-to-node pacing was defined

TCP congestion control and avoidance (PJ29144, PUT 18)
- Congestion control (based on RFC 2001) is reactive
  - Slow start algorithm introduces traffic into the network exponentially when a socket is started
  - When slow start is completed, keep increasing the rate at which traffic is sent to the network, but do so more slowly
  - If packets are lost, reduce the rate at which packets are sent for this socket
- Congestion avoidance is proactive
  - TPF monitors round-trip times (RTTs) of packets sent and reduces the rate at which packets are sent if RTT values increase
  - The goal is to detect congestion and reduce the rate before packets are lost
- Combination of congestion control and avoidance greatly reduces the amount of lost packets, and increases end to end throughput
- Valuable when sending large amounts of data across slow networks.
Load Balancing
What Name Are You Going by Today?

- TPF DNS Server support (PJ27268, PUT 13):
  - When a remote client wants to connect to TPF, it sends a request to DNS to resolve the host name of TPF to an IP address
  - With the DNS server residing in TPF, the request flows into TPF
  - Customizable code in TPF selects the TPF IP address to send back to the remote client
    - The TPF IP address determines the TPF host and network interface on that host that will be used for the new connection with this client
    - DNS response is marked as "do not cache" so that subsequent session requests will ask TPF again and be balanced correctly
  - TPF DNS server responds only with TPF IP addresses that are active
    - Using remote DNS servers instead cannot guarantee this
  - TPF DNS server centralizes the load balancing logic
    - Having each business partner's DNS server balance the load for its sessions does not always result in an overall balanced load.
TPF DNS Server Example

TPF DNS server
- cookies.yum
- chocolate.cake
- boston.creme.pie
- sugar.hi

IP 1

TPF DNS server
- cookies.yum
- chocolate.cake
- boston.creme.pie

IP 4

TPF DNS server
- cookies.yum
- chocolate.cake
- boston.creme.pie

IP 5

TPF DNS server
- cookies.yum
- chocolate.cake
- boston.creme.pie

IP 6

TPF Host Name | IP Addresses
---|---
cookies.yum | 1 2 3 4 5 6
chocolate.cake | 1 4
sugar.hi | 1 4
boston.creme.pie | 5

/etc/host.txt file

remote client
Spread the Wealth

- Wildcard support for TPF DNS server host names (PJ28093, PUT 15)
  - Ability to define an entry in /etc/host.txt file specifying a wildcard
  - One entry (like *.mytpf.com) can represent multiple host names
- Movable VIPA support (PJ27333, PUT 13)
  - Ability to move a VIPA from one TPF processor to another in the complex
    - ZVIPA operator command
  - When a VIPA moves, remote users tied to that VIPA will connect to the VIPA on the new processor
- Program interface to move a VIPA (PJ27491, PUT 14)
  - Allows software programs to do load balancing
- `activate_on_receipt` (AOR) I-stream balancing (PJ27679, PUT 14)
  - AOR creates a new ECB when data arrives on a socket
  - New ECB always used to be created on the same I-stream as the ECB that issued AOR
  - Now have the option to create the new ECB on the least busy I-stream.
Advanced APIs and Socket Options
Share and Share Alike

- Most socket implementations are process-scoped
  - A given socket is tied to a process (ECB)
  - Only that ECB can use the socket
  - If the ECB ends for any reason, the socket fails
- Kernel-based sockets support (PJ26683, PUT 11)
  - Sockets are owned by the TPF system, not a specific ECB
  - Allows a socket to be shared by multiple ECBs
  - Allows a socket to be passed from one ECB to another
  - Necessary for distributed application design that exists in most TPF systems
- Socket sweeper support (PJ26683, PUT 11)
  - Detects sockets that have not been used for an extended period of time and cleans up those sockets
  - Prevents memory leak of TCP/IP resources when application programs forget to clean up sockets themselves.
TPF-Unique Socket APIs

- **activate_on_receipt (AOR)** support (PJ26683, PUT 11):
  - Similar to standard `read` API, but allows data to be read in an asynchronous manner
  - ECB issuing AOR can exit
  - New ECB is created when data arrives on the specified socket
  - No ECBs tied up (suspended) while waiting for data to arrive
  - First introduced with TPF's SNA LU 6.2 support

- **activate_on_accept (AOA)** support (PJ26683, PUT 11):
  - Similar to standard `accept` API, but allows TCP connection requests to be processed in an asynchronous manner
  - No ECBs tied up (suspended) while waiting for remote clients to connect
  - AOA model for Internet Daemon (INETD) support (PJ28496, PUT 17)
    - Allows TCP server applications started, stopped, and monitored by INETD to use AOA.
More TPF-Unique Socket APIs

- TCP architecture has no concept of a message
  - Only application programs know where message boundaries are
- Common application programming practice is to place a header at the start of the message that contains the length of the message
- Application programs do at least two read APIs per message
  - One or more read APIs to get the header
  - One or more read APIs to get the message
- Message reassembly is a frequent source of application problems
- tpf_read_TCP_message and activate_on_receipt_of_TCP_message support (PJ29118, PUT 18)
  - Ability to read an entire TCP message with a single API
  - One complete message is returned - never a partial message or more than one message
  - Application program specifies the message format as input to the API, including where the length field resides in the header
  - tpf_read_TCP_message is synchronous like read
  - activate_on_receipt_of_TCP_message is asynchronous like AOR.
Advanced `setsockopt` Options Supported by TPF

- **SO_KEEPALIVE support (PJ26683, PUT 11):**
  - Keep-alive option sends "heartbeat" message on TCP sockets that have been idle to verify that the remote partner is still active
  - Keep-alive timer value
    - RFC 1122 defines default value to 2 hours
      - Poor choice, even for 1989 when this RFC came out
    - For PUTs 11-17, the keep-alive timer value is the same as the socket sweeper value (SOCKSWP value in CTK2)
    - Keep-alive timer can now be configured by the user and specified in seconds (PJ28996, PUT 18)

- **SO_RCVLOWAT support (PJ26683, PUT 11):**
  - Defines the minimum amount of data that must come in before a read-type API is posted
  - Without this, read-type APIs are posted when any data comes in (even 1 byte) - this is standard sockets behavior
  - Useful to reduce the number of read APIs when the application is receiving a large message and knows the length of the message.
Send Me Everything You Have As Fast As You Can

- Standard TCP window size maximum is 64 K
- 64 K can limit throughput on modern high-speed connections
- TCP window scaling support (PJ26683, PUT 11)
  - Enables large TCP window sizes, up to 1 MB for TPF
  - Remote partner must also support window scaling to use this option
- Large SO_SNDBUF support (PJ26683, PUT 11)
  - Can define a socket's send buffer size to be up to 1 MB
  - Large buffer size can help enable high throughput over a socket
- Large SO_RCVBUF support (PJ26683, PUT 11)
  - Can define a socket's receive buffer size to be up to 1 MB
  - Amount of available space in the socket's receive buffer determines the window size advertised to the remote end
    - Defining a large receive buffer size is crucial for high throughput over a given socket.
Big... No, I Mean Really Big!

- TPF socket send-type APIs are atomic operations
  - All of the data is sent or none of the data is sent
  - Prevents interleaving of partial messages when a socket is shared by multiple ECBs
- Limit for a send API was 32 K - same limit as TCP/IP offload support
- Greater than 32 K send support (PJ28087, PUT 15)
  - Allows more than 32 K of data to be passed on a send-type API
  - Enables easier porting of applications that send large data
  - If data on send is larger than the socket send buffer size
    - System accepts as much data as possible
    - Application is responsible for issuing more send APIs to send the remainder of the data
  - If data on send is not larger than the socket send buffer size
    - No change to existing behavior - all or none of the data is sent
    - No code or behavior changes for existing applications.
I'm Not Willing to Wait Forever!

- Many socket APIs can cause ECBs to be suspended while waiting for data to arrive from the network
  - Includes send-type APIs waiting for an ACK to come in and free up space in the socket's send buffer
- Waiting "forever" in a real-time environment is not acceptable
- You can put the socket in nonblocking mode to prevent the ECB from being blocked (suspended)
  - Application must implement retry logic in this case
- Better solution in most cases is to define timeout values
- SO_SNDTIMEO and SO_RCVTIMEO support (PJ26683, PUT 11)
  - Every socket API that could block the ECB has timeout capability
  - Default is 0, which means do not time out
  - SO_SNDTIMEO defines timeout value for:
    - `send`, `sendto`, `write`, `writev`
  - SO_RCVTIMEO defines timeout value for:
    - `accept`, `AOA`, `AOR`, `connect`, `read`, `recv`, `recvfrom`, and all other read-type APIs.
TPF-Unique ioctl Options

- **AOR_BALANCE**
  - If enabled, creates the new ECB on the least busy I-stream when data arrives on the socket during AOR processing
  - You should enable this option if the application is capable of running on multiple I-streams

- **TPF_NOSWEEP**
  - If enabled, this socket will not be cleaned up by the socket sweeper if no APIs have been issued on this socket for a long time
    - Socket sweeper will still clean up the socket for a network failure for an extended period of time
  - You should enable this option if the socket is shared by multiple ECBs and can go for a long period of time without any ECB using the socket

- **TPF_NOSLOWSTART**
  - If enabled, slow start processing is skipped when this TCP socket is started
    - Slow start processing is always done if congestion is encountered
  - Slow start processing can impact response time, especially for short-lived sockets
  - Candidates for disabling initial slow start include:
    - Short-lived sockets where TPF sends more than a packet's worth of data
    - Sockets on a local high-speed network.
The Difference between Working and Working Well

- `getservbyname` and `getservbyport` API support (PJ28195, PUT 16)
  - Maps a server name to its port number and protocol, or vice versa
  - Information is defined in the network services database (NSD)
    - `/etc/services` file
- Extended NSD support
  - Define more characteristics for server applications
  - Message and packet rate information (PJ28195, PUT 16)
  - Different DiffServ codepoint values (PJ28195, PUT 16)
  - TCP connection limiting (PJ28493, PUT 17)
  - Traffic limiting (PJ28901, PUT 18)
- Display NSD information (PJ28901, PUT 18)
  - Ability to display information about applications defined in the NSD, including:
    - Defined parameters
    - Current message and packet rates
    - Connection and traffic limits - current and high-water marks
    - TCP backlog information - current and high-water marks.
Security
I Hear You Knocking, but You Can't Come In

- Firewall support:
  - Prevent several denial of service (DoS) attacks (PJ26683, PUT 11)
  - Detect and prevent more DoS attacks (PJ28587, PUT 17)
  - Packet filtering support (PJ28213, PUT 16)
    - Allows or denies traffic coming into TPF based on the source, destination, or both
      - Violations flagged as such in IP trace
    - Important part of a comprehensive security strategy
      - Need filtering support in edge routers as well; however filtering in the host provides added benefits:
        - Methods exist to bypass packet filtering in some routers
        - Allows you to limit access to applications to only certain users on the local network
    - Packet filtering along with definitions in the network services database (NSD) enables policy-based networking (PBN) on TPF.
I See Your Lips Moving, but I Cannot Understand You

- **Secure Sockets Layer (SSL) support (PJ27863, PUT 15):**
  - Based on OpenSSL support
  - Enables secure communications between TPF and a remote node across an insecure (public) network
  - Supports standard RSA public-key cryptography for session setup
  - Supports standard algorithms for data exchange, including:
    - DES, triple-DES (3DES), RC4, MD5, SHA

- **Shared SSL support (PJ28118, PUT 15):**
  - Ported OpenSSL code is process-scoped
    - SSL session tied to a process (ECB)
    - If ECB ends, the SSL session ends
  - Now SSL sessions can be shared by multiple ECBs
  - Includes AOR support for SSL

- **Secure Apache Web Server support (PJ28118, PUT 15):**
  - Enables secure Web transactions with Apache on TPF using SSL.
You Want It All... Sorry, You Can Only Have So Much

- **TCP connection limiting support (PJ28493, PUT 17)**
  - Limits the number of active sockets for a specified TPF application
  - Prevents one application from using more network or TPF resources than the application is supposed to use

- **Traffic limiting support (PJ28901, PUT 18)**
  - Limits the rate of traffic for a given TPF application, socket, or both
  - Works for both TCP and UDP applications
  - Prevents one user or application with long-running connections from using more network or TPF resources than it is supposed to use
  - No application programming changes are required

- Both traffic limiting and TCP connection limiting act like resource managers and can prevent flooding
  - Intentional - attack of your system
  - Unintentional - unexpected surge in traffic.
Diagnostic Tools
Let's Go to the Videotape

- **IP trace support (PJ26683, PUT 11):**
  - Ability to log IP packet data
  - Use IPTPRT offline facility to find all packets matching a set of input criteria
  - Online display useful for test systems
  - Output can be compact (1 packet per line) or formatted showing many fields in a more readable format

- **Individual IP trace support (PJ27617, PUT 14):**
  - Allows you to trace specific sockets or applications
  - Separate from the systemwide IP trace
  - Useful for debugging problems in real time on production systems
    - No need to switch tapes and process them offline
    - Just set up the trace criteria and display the data online
  - Also useful for debugging applications in a shared test system
  - Option to prevent an individual IP trace from wrapping
    - Useful when first flows are the ones in question.
Tell Me Why

- Many different conditions can cause a TCP connection to fail, including:
  - Server not active, server backlog limit exceeded, operator command deactivated the socket, socket sweeper cleaned up the socket
- TCP RST (reset) message flows
- Why the reset is sent is not included in the TCP RST message
  - TCP has no equivalent of SNA sense codes to explain errors
- Exception codes in IP trace support (PJ28213, PUT 16)
  - Reason code added to TPF IP trace entries that are exception conditions, including:
    - Any time TPF sends a TCP RST
    - Packet violates a packet filtering rule
    - Packet is potentially a denial of service (DoS) attack
  - Included in both the systemwide IP trace and individual IP trace
  - Makes debugging problems easier.
Show and Tell

- Displaying sockets:
  - ZDTCP NETSTAT support (PJ26683, PUT 11)
    - Shows TPF servers that are active along with how many remote clients are connected to each server
  - ZSOCK DISPLAY support (PJ26904, PUT 12)
    - Display one or more sockets based on a set of input criteria
      - For example, all sockets connected to a specific port
  - ZSOCK IPMT support (PJ28997, PUT 18)
    - Display sockets that are using the most IP message table (IPMT) blocks
    - Can use ZSOCK DATAFLOW (PJ27650, PUT 14) to determine if data is flowing on the sockets
    - Can deactivate sockets that are hung
  - ZSOCK CONVERT support (PJ26904, PUT 12)
    - Convert IP address and ports to its file descriptor (FD), or vice versa.
You are the Weakest Link, Goodbye!

- **ZTTCP INACT SOCKETS** support (PJ26683, PUT 11)
  - Deactivate all sockets
- **ZSOCK INACT** support (PJ26904, PUT 12)
  - Deactivate one or more sockets. For example:
    - Deactivate one specific socket
    - Deactivate all sockets with a specific remote client
    - Deactivate all sockets with a specific TPF application
- **ZSOCK DISABLE** supports (PJ28526, PUT 17)
  - Prevents new sockets from starting while allowing existing sockets to end normally
  - Sample procedure for orderly network shutdown:
    1. ZSOCK DISABLE (prevent any new sockets)
    2. ZTTCP INACT SOCKETS (deactivate all active sockets)
    3. ZTTCP INACT ALL (deactivate all network connections).
Counting Sheep

- Message and packet rate information (PJ28195, PUT 16):
  - Count message and packet rate information per TCP/IP application
    - Separate sent and received counters
  - Results displayed in TPF data collection reports
  - Messages can be weighted to represent resources consumed by that application
  - What is a message?
    - User applications:
      - Can treat each send- or read-type API as one message
        - No change to application programs needed in this case
      - Can update application program to issue `tpf_message_count` API when a message is sent or received
        - Only the application program really knows what a message is
        - Can define pseudo-ports to count messages for non-TCP/IP applications or split one application's traffic into different categories
    - System applications and middleware:
      - All have been updated to update counters based on the definition of a message for the specific application
- Display message rate information online (PJ28901, PUT 18):
  - Ability to display message and packet rate information online.
Managers Can Be Useful Every Now and Then

- Simple Network Management Protocol (SNMP) agent support (PJ27932, PUT 15):
  - Allows TPF to be monitored by SNMP managers, just like any other TCP/IP node
    - For security reasons, SNMP managers can only monitor TPF, not update TPF
  - Includes standard Management Information Base (MIB) 2 variables
    - Ability to add user MIB variables
  - Trap support
    - TPF sends standard traps (like link up and link down)
    - Program interface to send user traps
      - Can be used to send alerts of any kind, not necessarily network-related events
- Display SNMP MIB contents (PJ28168, PUT 16):
  - Ability to display SNMP MIB variables online
  - Includes individual variables or all of a specific group (system, TCP, UDP, IP, and so on).
Summary

- TCP/IP continues to evolve in the industry
- TPF TCP/IP support continues to evolve as well
- Like 2002, there have been a significant amount of TCP/IP migrations this year as well at TPF accounts
- More and more "bells and whistles" are being added to meet your needs and requirements
- We look forward to working with you as your TCP/IP migrations continue
  - Helping you figure out to best exploit the many existing features in TPF TCP/IP support
  - Helping to identify and solidify future requirements.
Legal Information

Any references 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.

IBM is a registered copyright of the IBM Corporation.

Other company, product, and service names may be trademarks or service marks of others.