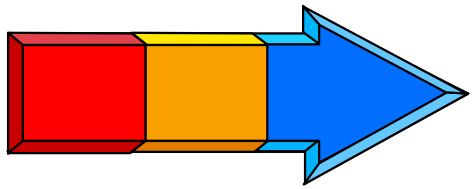
# TCP/IP Native Stack Support Test Results

Superior Performance for IP Access to TPF Hosts

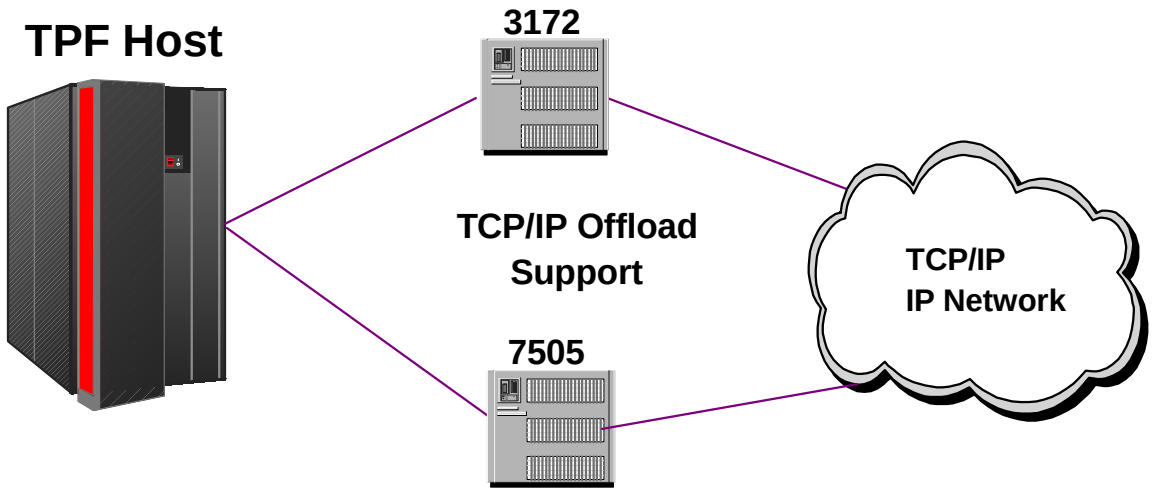
John Foley
World Wide Communication Controller Brand Marketing
foleyjoh@us.ibm.com
February 2000

(c) Copyright IBM Corporation 1999



# Old Support for IP Access to TPF Host

- **Prior to October 1999:**
  - **The only IP access to a TPF Host was TCP Offload**
    - **Only available via 3172 or Cisco 7500**
    - **Throughput was poor**
    - **CPU cycles were high**



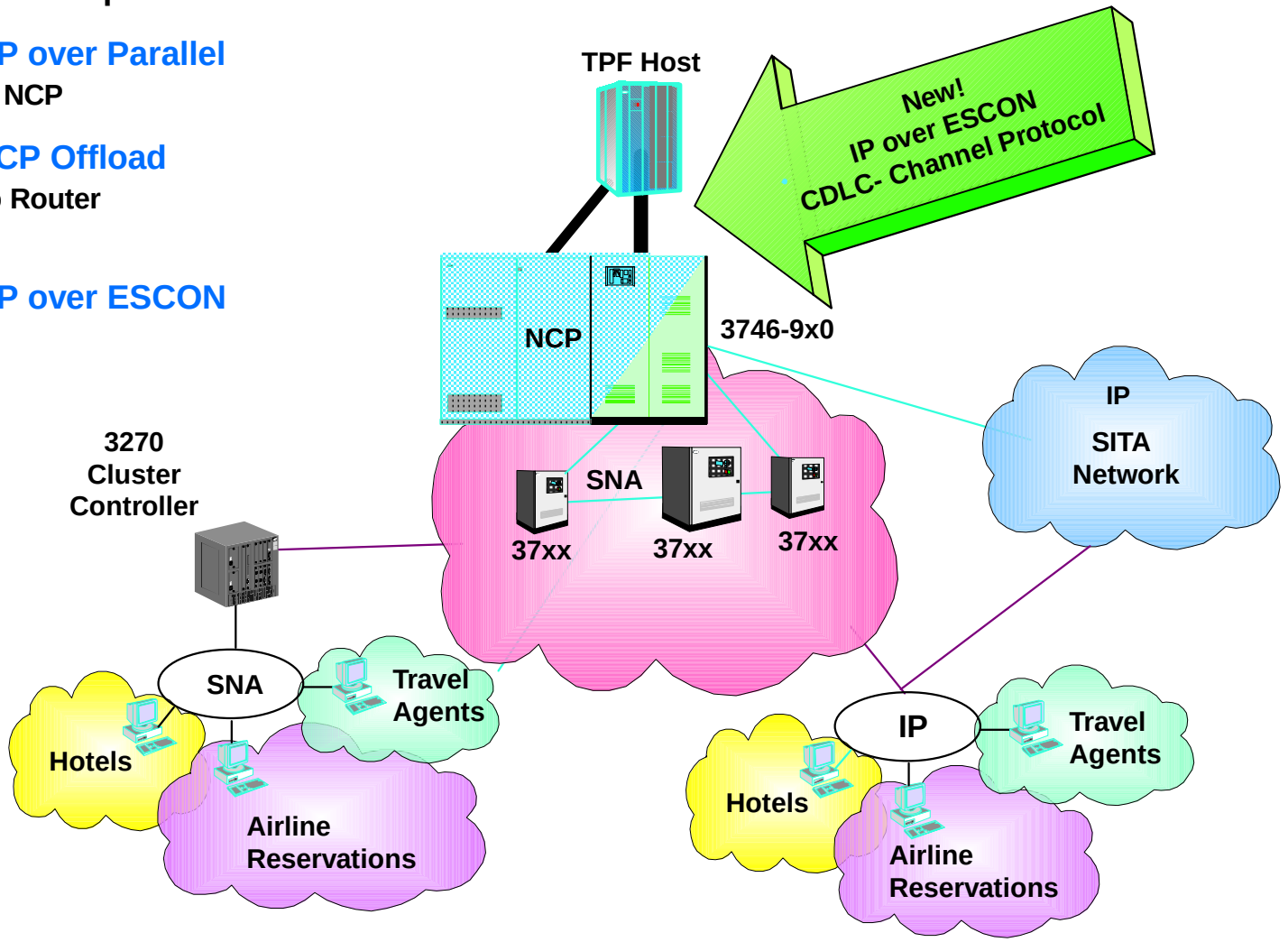
# New IP Access to TPF Host

- **Available NOW:**
  - **New TPF support for Native IP over Parallel or ESCON Channels**
    - **Only available via 3745 or 3746 Model 900 or 950**
    - **Throughput is Superior**
    - **CPU cycles are low**

# New Native TCP/IP Stack for TPF*

**Only three TPF IP options:**

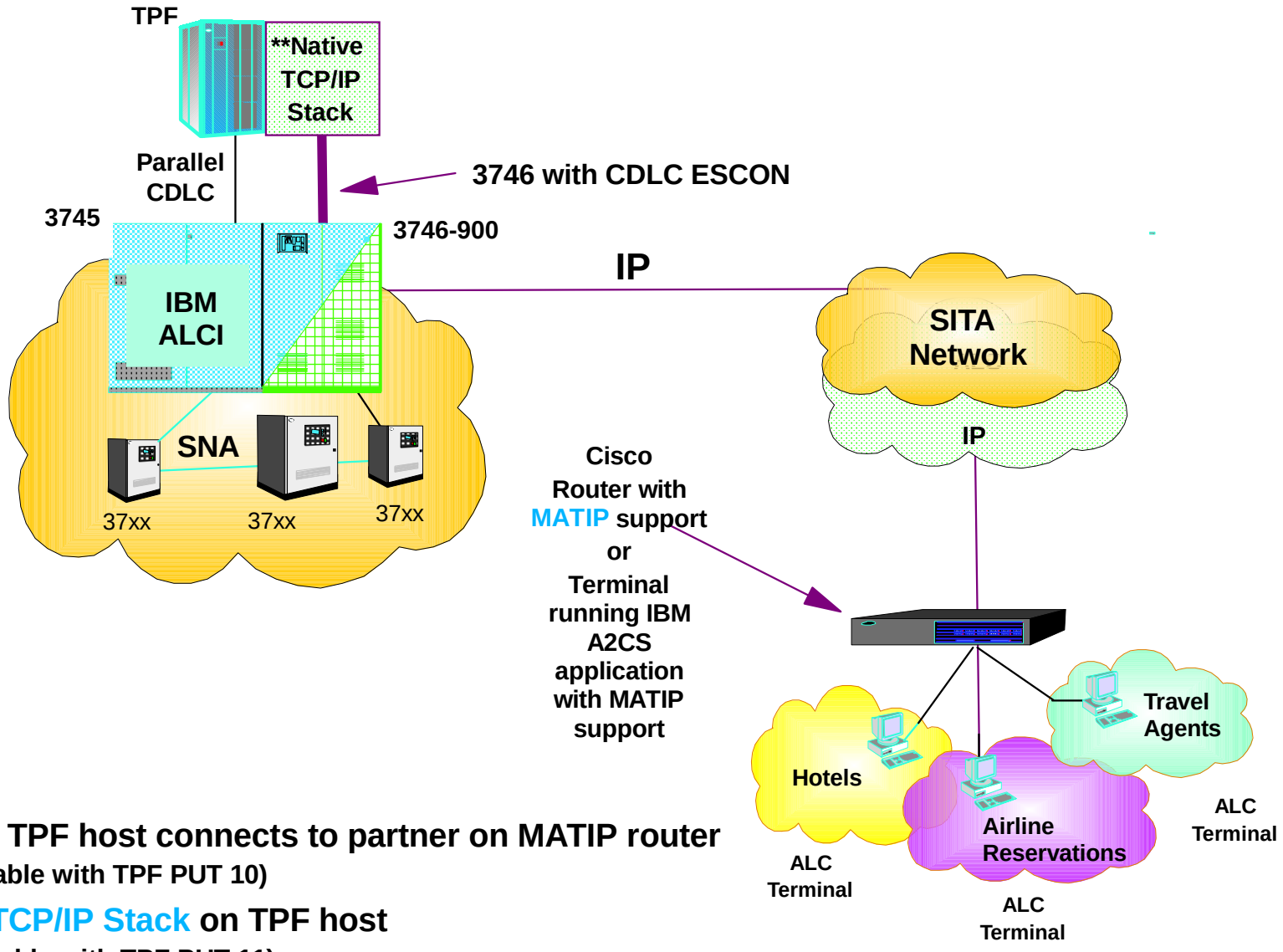
- **Native IP over Parallel**
  - 3745 NCP
- **IP via TCP Offload**
  - Cisco Router
  - 3172
- **Native IP over ESCON**
  - 3746




*Available via TPF PUT 11

# TPF Support for MATIP*

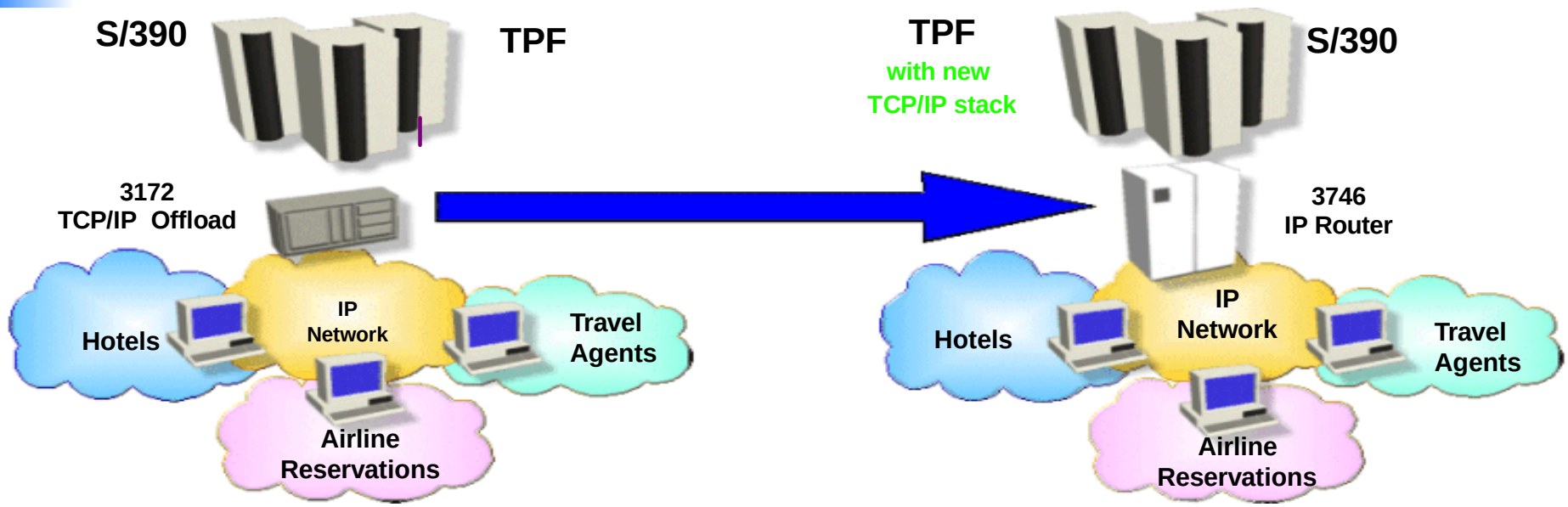
Map ALC To IP = MATIP




*MATIP on TPF host connects to partner on MATIP router
- (available with TPF PUT 10)

**Native TCP/IP Stack on TPF host
- ( available with TPF PUT 11)

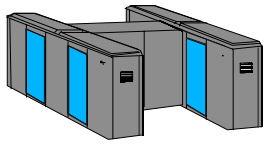
# IBM 3746 Supports TPF Native IP



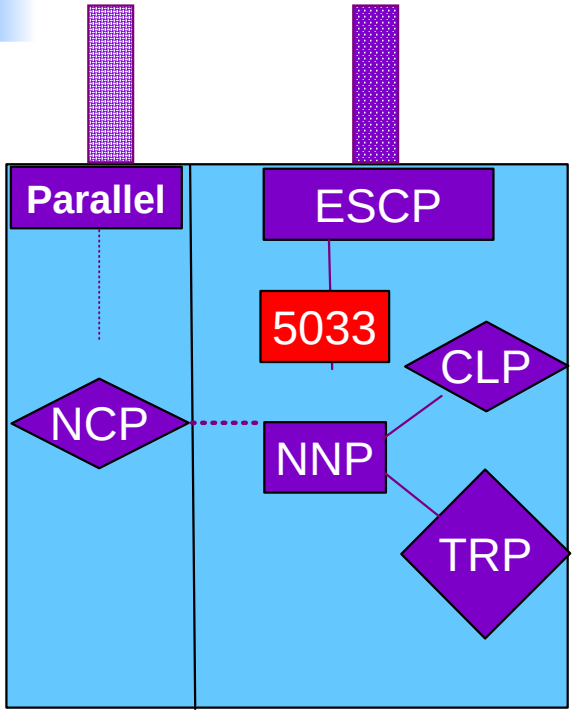

- 3746 CDLC channel support Improves TPF TCP/IP Performance
  - Replace 3172s or other offload platforms in data center
  - Take advantage of existing 3746 platform and management function

# Prerequisites:

TPF Host



- **TPF 4.1**
  - PUT Level 10 for MATIP support
  - PUT Level 11 for Native IP Stack




- **3746-/900/950**
  - TRP (for Token Ring) or CLP (for Frame Relay)
  - Network Node Processor (NNP)
  - ESCON Processor (ESCP)
  - IP Feature (5033)

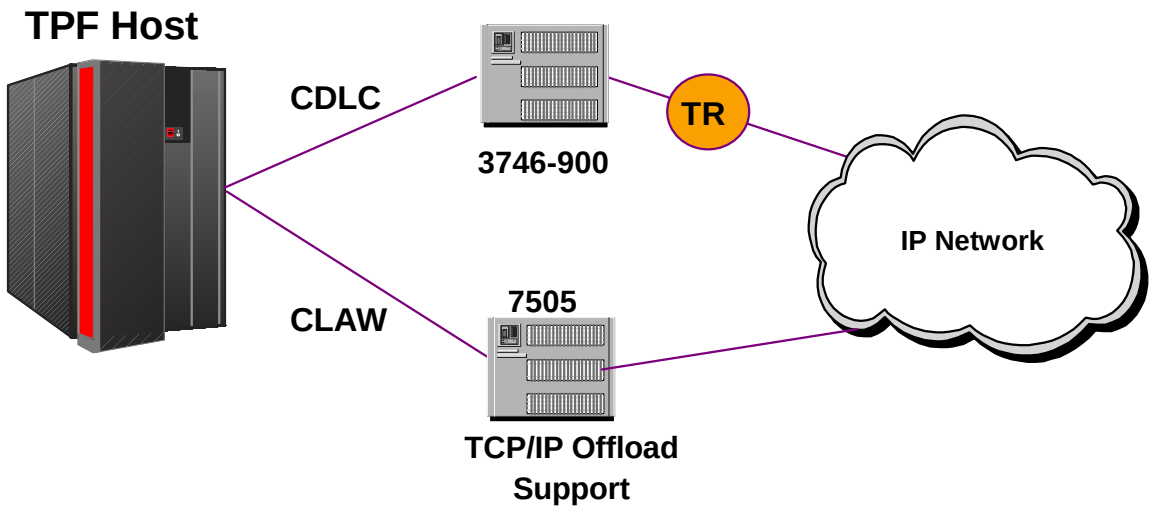
**Note: TPF's Native IP support is also available over NCP's Parallel channels but Performance was not measured in this configuration**

# Compare: TCP/IP Offload and Native Stack Test

- **Test Environments**
  - **TPF TCP/IP Offload using a Cisco 75xx Router with CLAW protocol over the channel**

    **versus**

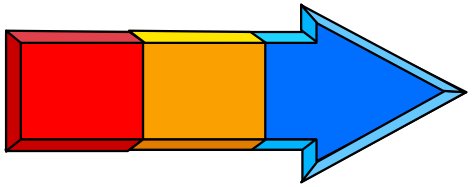
  - **TPF TCP/IP Native IP Stack using a 3746-900 Controller with CDLC protocol over an ESCON channel.**



# Test Results Summary

- **Test Results Using the 3746 with the new TPF IP Stack compared to the TCP Offoad function**
  - **Path Length Test Results - Client Side of TPF System**
    - **CPU utilization was 3.5 to 38.2 times less on the I-stream traffic**
  - **Path Length Test Results - Server Side of TPF System**
    - **CPU utilization was 4.9 to 30.7 times less on the I-stream traffic**
  - **READ Loop Test Results**
    - **Performance (Message rates) were 2.3 to 16.3 times higher**
  - **Using TCP/IP offload support:**
    - **SELECT causes extra I/O operations**
    - **Message rate (messages per second) on the socket dropped by 62%**
  - **Using TCP/IP native stack support:**
    - **SELECT does not cause any I/O**
    - **No change in the message rate**

# Test Scenarios (back-up)



# Test Case Scenarios

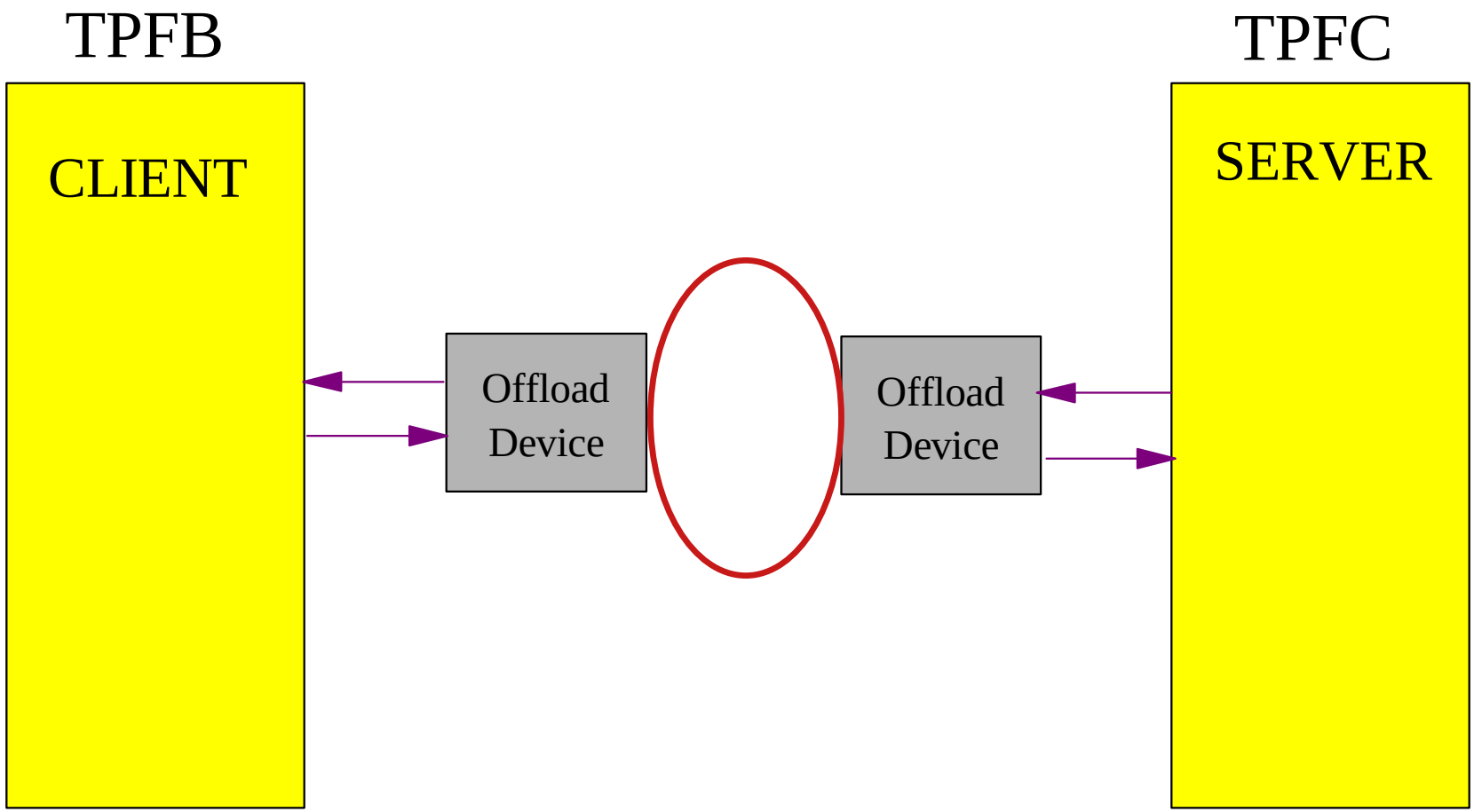
- **Read Loop Test**
  - TCP/IP Offload and TCP/IP Native Stack
  - Message Rate
  - Message Rate Comparison
  - Bytes per Second
  - Test Results
- **Path Length Test Environments**
  - Test Results Client Side and Server Side TPF System
  - Test Summary
- **Select/Read Test Environment**
  - TCP/IP Offload and TCP/IP Native Stack
  - Test Results
- **TCP/IP Native Stack Stress Test Results**

**(*) TPF Development Test Results presented at TPF Users Group Meeting, October 1999**

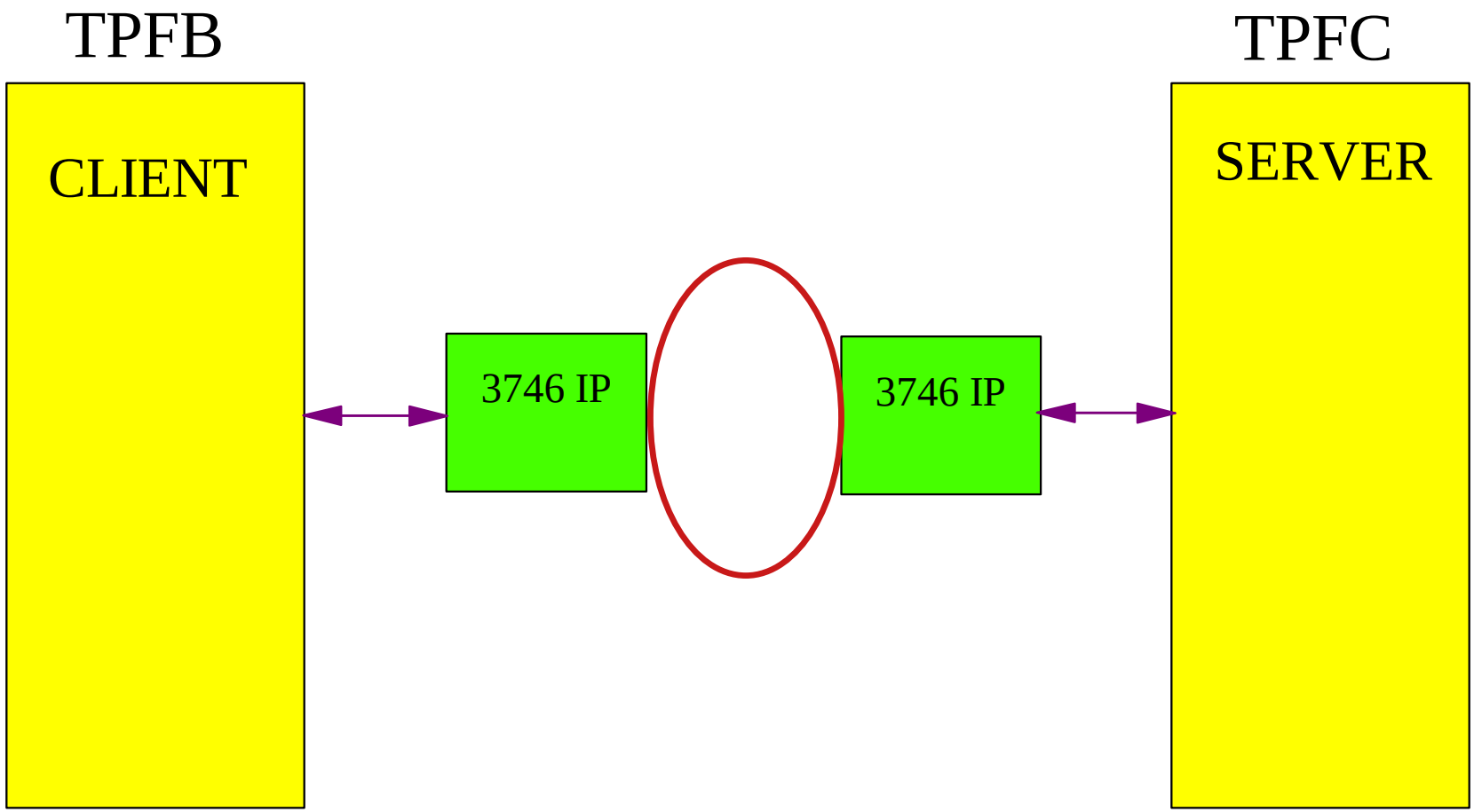
# Test Environments

- Two TPF systems (TPFB and TPFC) running multiple I-streams
- Socket client applications reside in TPFB
- Socket server applications reside in TPFC
- Cisco 7500 offload devices were used to test TCP/IP offload support
- IBM 3746 IP Feature was used to test TCP/IP native stack support
  - **IBM 3745 IP function were tested for connectivity, but 3746 IP Feature was used to gather performance test data**

# TCP/IP Offload Test Environment



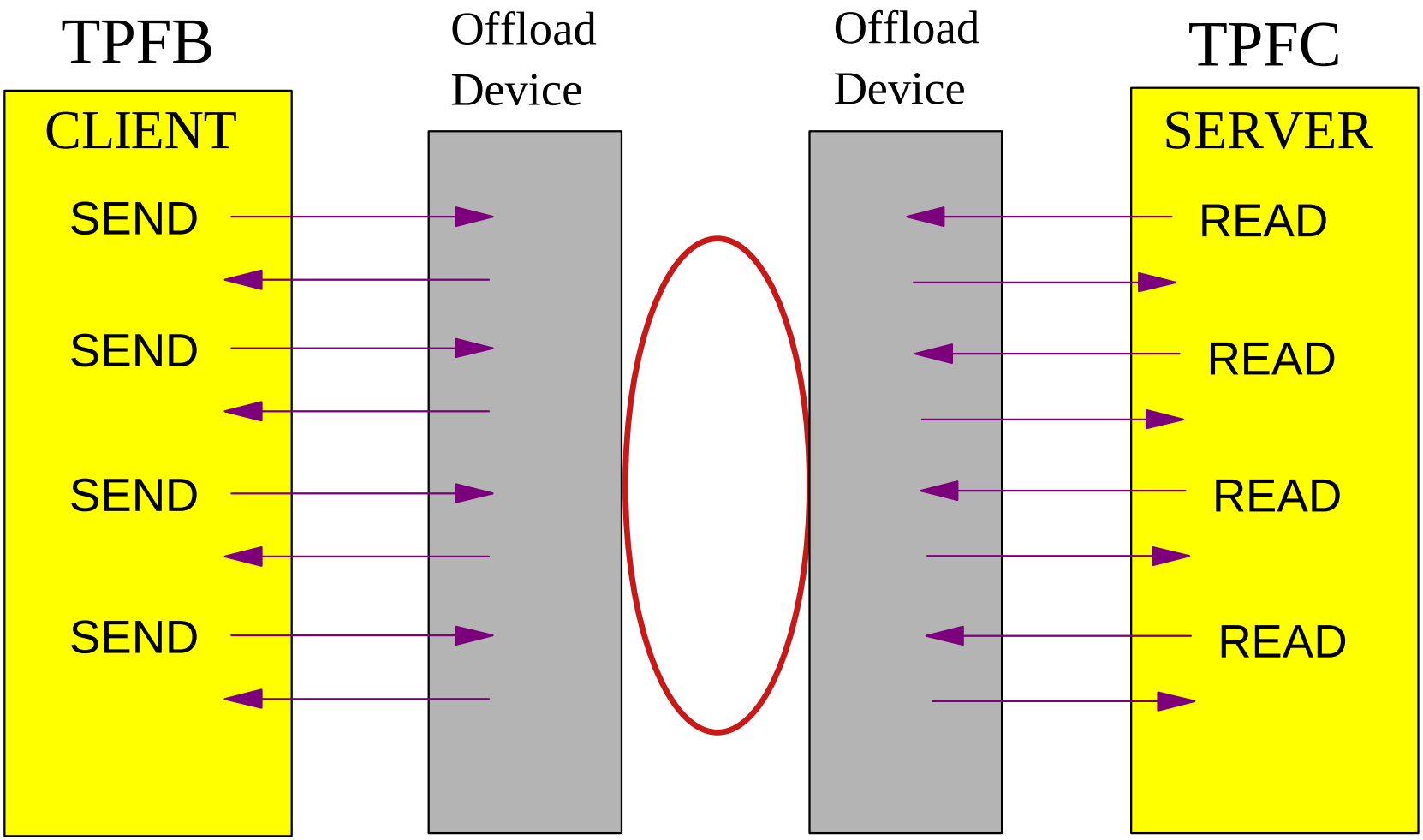
# TCP/IP Native IP Stack Test Environment



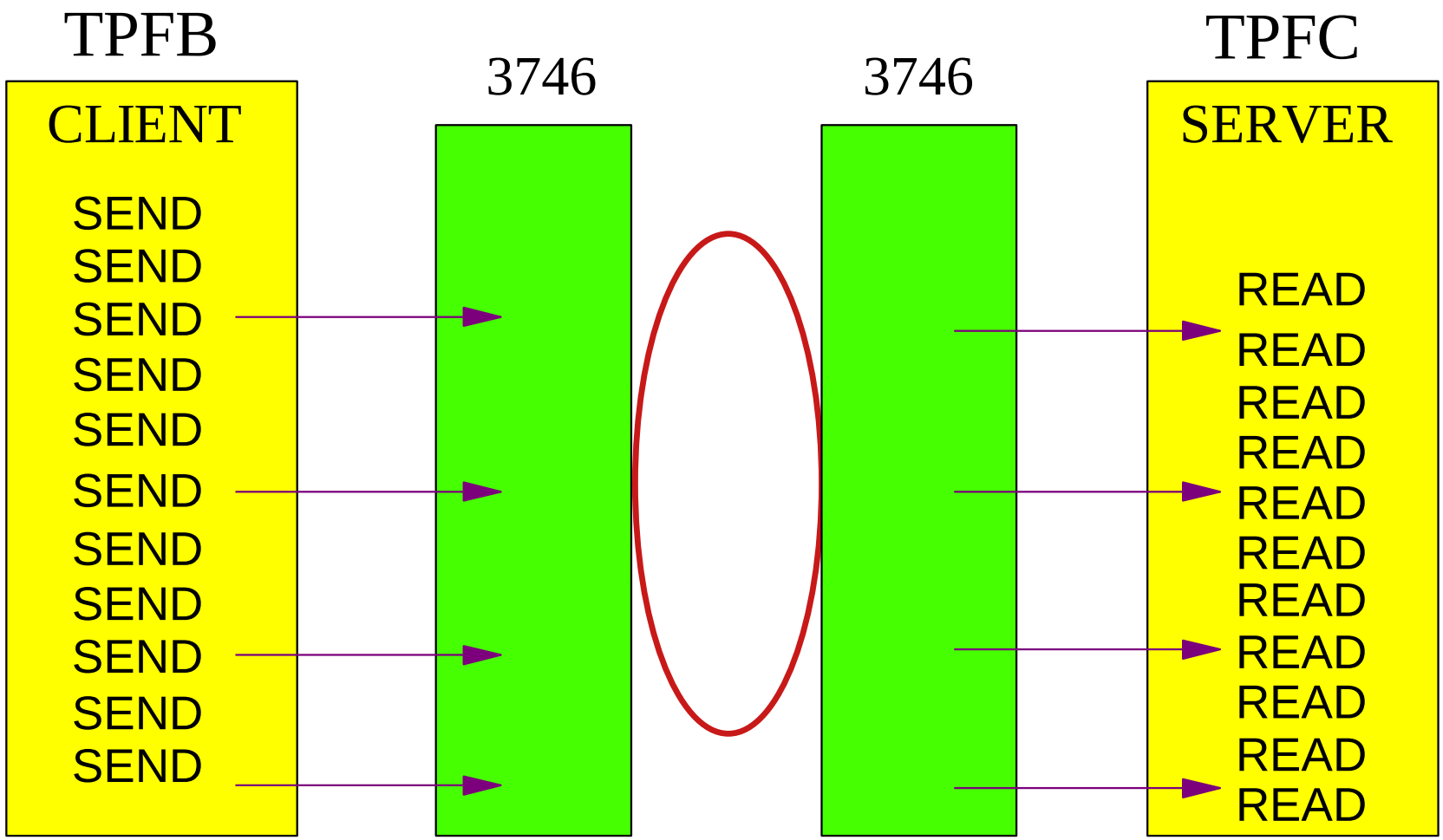
# READ Loop Testing

- Client establishes a TCP socket with the server
- Client sends one million fixed-size messages as fast as it can in a SEND loop
- Server reads the messages as fast as it can in a READ loop
- Size specified on the READ is the message size
- One message per READ call
  - If the entire message was not received, another READ is issued with the remaining size
- Test was run many times, each with a different message size

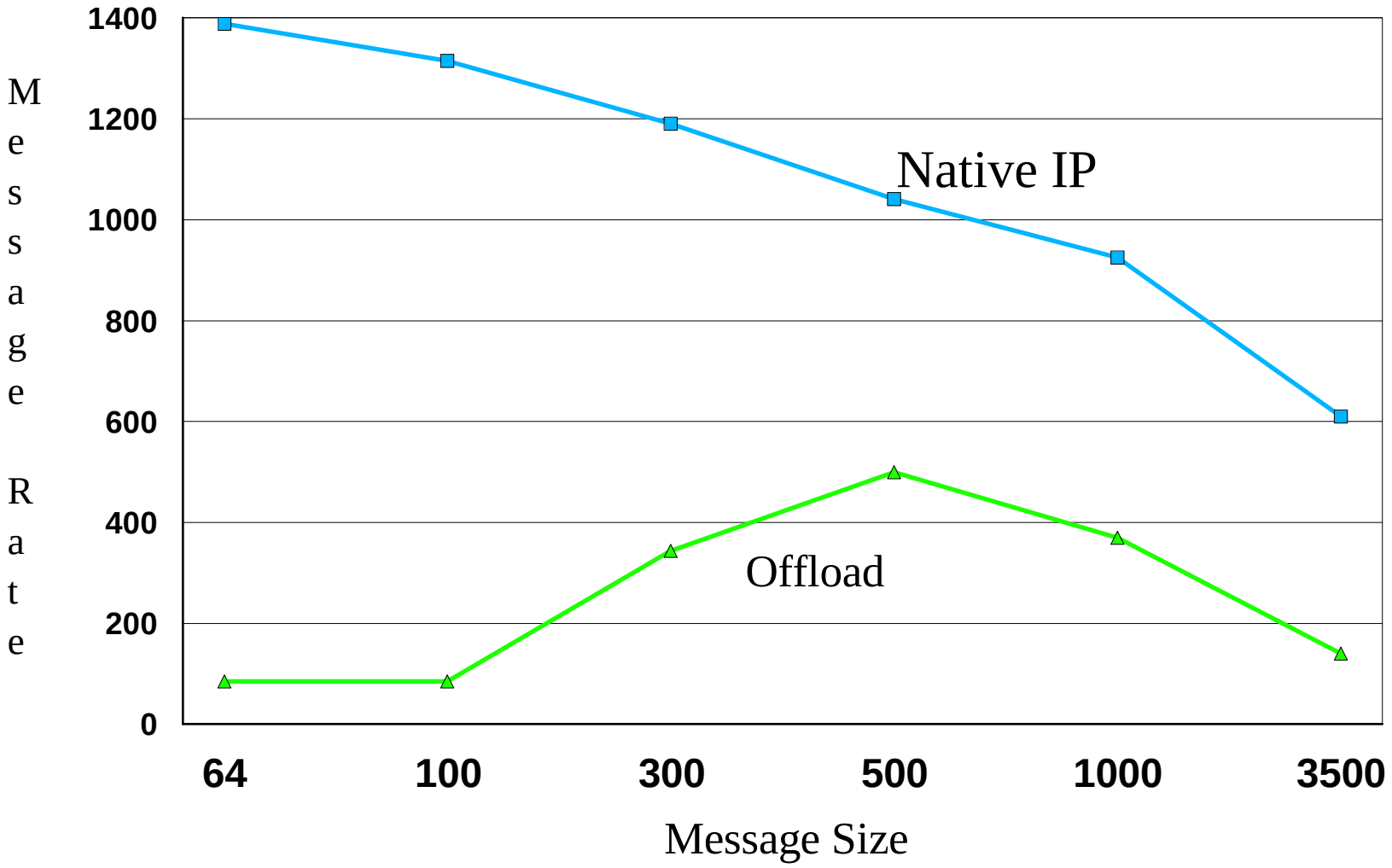
# TCP/IP Offload READ Loop Test



# TCP/IP Native IP Stack READ Loop Test



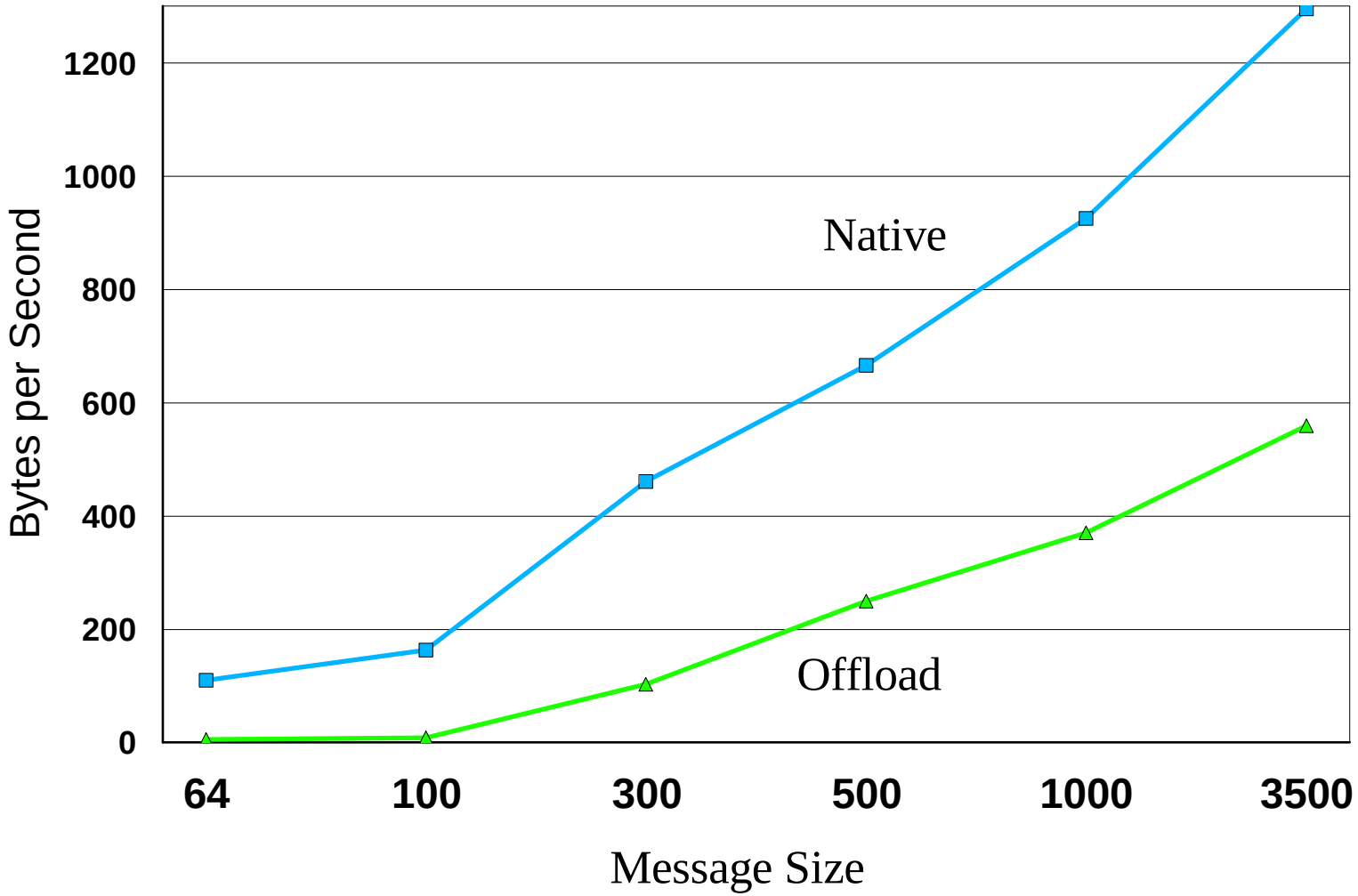
# READ Loop Test - Messages Rates



# READ Loop Test Results

- Message rate was 2.3 to 16.3 times higher using TCP/IP native IP stack support
- Using TCP/IP native IP stack support
  - **Direct correlation between the message size and the message rate**
- Using TCP/IP offload support
  - **CLAW design is not efficient for small messages**
  - **Message rate peaked at 500-byte messages**
  - **Message rate dropped off again using messages larger than 500 bytes**

# READ Loop Test - Bytes per Second



# Path Length Test Environment

- READ loop test, 500-byte messages
  - Most efficient message size for TCP/IP offload support
- TPF CPU utilization data gathered
- Application (client or server) running on one I-stream only
- The only processing done by the other I-streams was I/O interrupts from the network
- No other activity in the TPF system
  - No DASD activity, no utilities running

# Path Length Test Results - Client Side TPF System

- **Using TCP/IP native stack support:**
  - CPU utilization was 3.5 times less on the I-stream where the client application was running
  - CPU utilization was 38.2 times less on the other I-streams
  - Average CPU utilization of all I-streams was 9.55 times less
- **Sending data uses 9.55 times fewer CPU cycles using TCP/IP native stack support compared to TCP/IP offload support**
- **All results on this page were measured in CPU cycles per message**

# Path Length Test Results - Server Side TPF System

- **Using TCP/IP native stack support:**
  - CPU utilization was 4.9 times less on the I-stream where the server application was running
  - CPU utilization was 30.7 times less on the other I-streams
  - Average CPU utilization of all I-streams was 17.2 times less
- **Receiving data uses 17.2 times fewer CPU cycles using TCP/IP native stack support compared to TCP/IP offload support**
- **All results on this page were measured in CPU cycles per message**

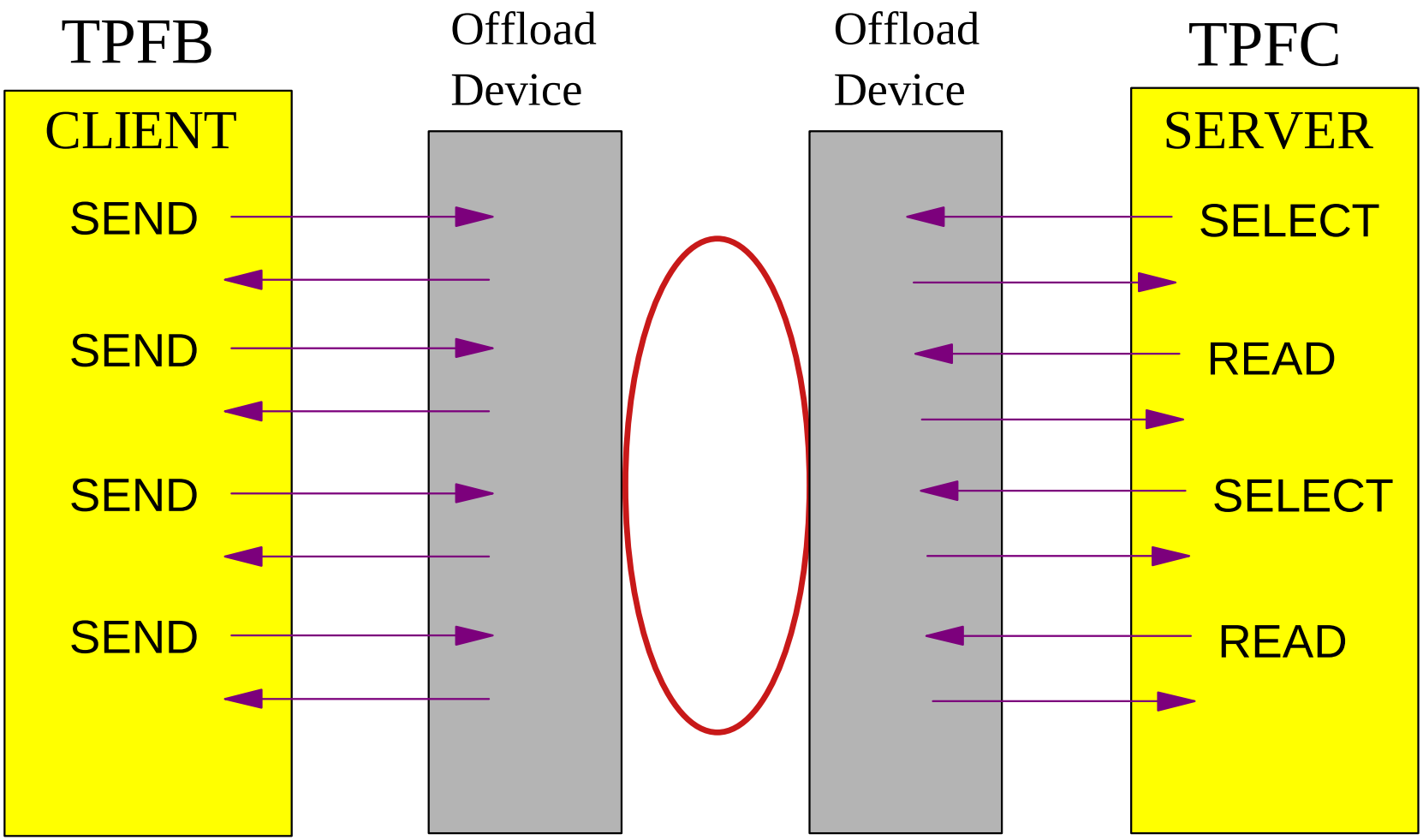
# Path Length Test Results - Summary

- **A common concern was that putting the TCP/IP stack in the TPF system would cause a large increase in path length**
  - **The results are the exact opposite**
- **TCP/IP native stack support uses 9.55-17.2 times fewer CPU cycles than TCP/IP offload support!**

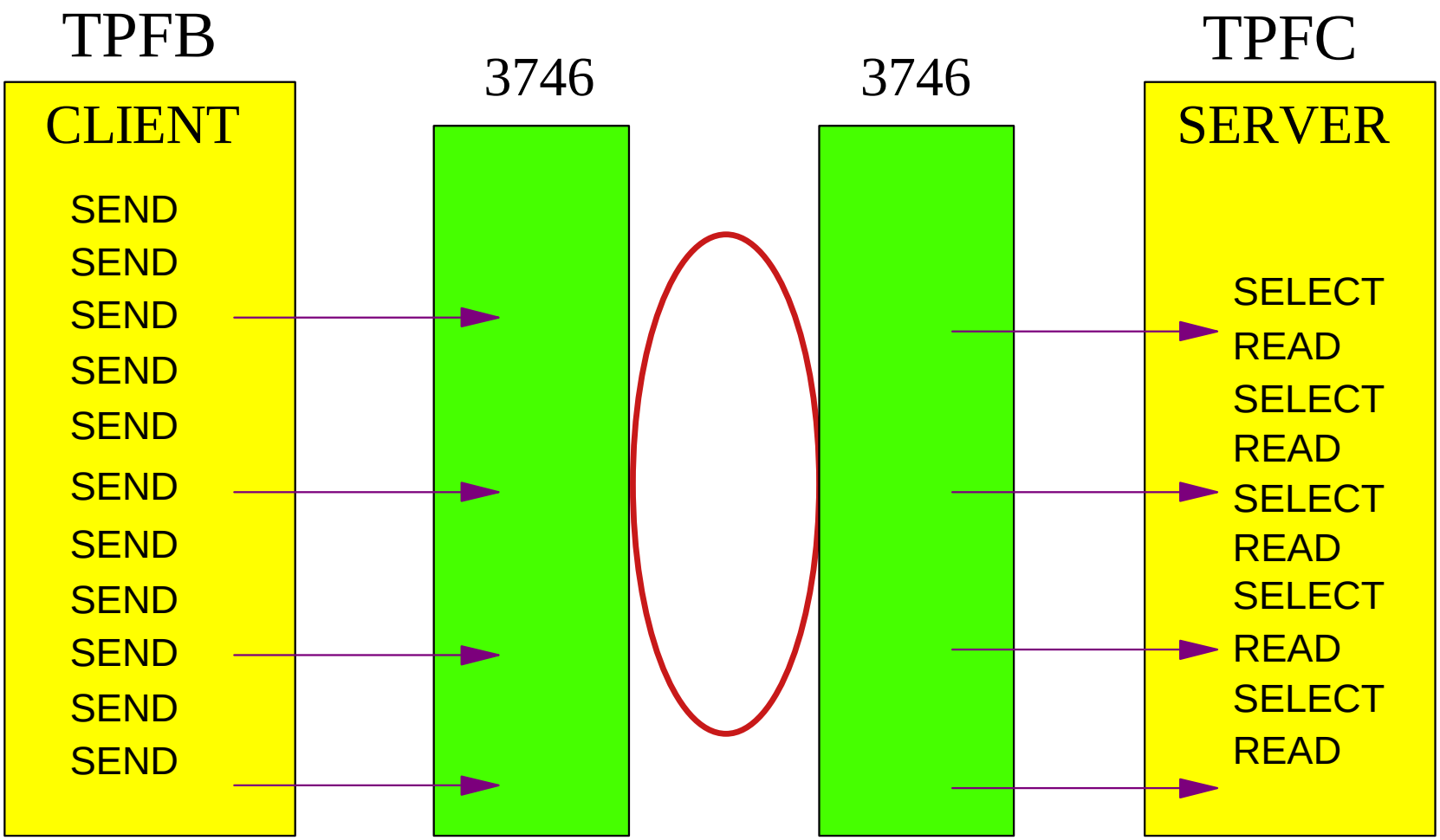
# SELECT/READ - Test Environment

- If server application requires a time out using TCP/IP offload support, a SELECT must be issued before each READ
- This is quite common in user socket applications and TPF system middleware
- The server application was changed to do SELECT/READ in a loop and its results were compared to when the server did only READs
- The purpose of this test was to determine how much of an overhead SELECT processing is

# TCP/IP Offload Support - SELECT/READ Test



# TCP/IP Native IP Stack - SELECT/READ Test



# SELECT/READ Loop - Test Results

- **Using TCP/IP offload support:**
  - **SELECT causes extra I/O operations**
  - **Message rate (messages per second) on the socket dropped by 62%**
- **Using TCP/IP native stack support:**
  - **SELECT does not cause any I/O**
  - **No change in the message rate**