CICS Integration & Optimization: Tales from the Trenches

Russ Teubner
HostBridge Technology
russ@hostbridge.com

Abstract

CICS users are loyal to their apps – and for good reason! However, they also need to integrate these same applications with an ever widening array of web and cloud-based resources. If that weren't enough, every year they are under pressure to add value, support new workload and reduce the cost of ownership. That's a tall order. This session will highlight two customers who used a common tactic to enhance the value of their existing CICS investments.

Customer Case Studies

- **Customer A**
  - **Industry: Telecommunications (US)**
  - **Very high daily/consistent transaction volume**
  - **Long-standing investment in COBOL-based socket apps**
- **Customer B**
  - **Industry: Financial Services (International)**
  - **Very high transaction volume on one day each month (and in compressed time period)**
  - **Long-standing investment in PL/I-based socket apps**

Common Objectives

- **Both customers had common objectives**
- **Business Objectives**
  - **Respond to <u>competitive pressures</u> in their industry**
  - **<u>Lower incremental cost</u> of high-volume CICS application processing (i.e., marginal value > marginal cost)**
  - **Move new/additional workload <u>to System z</u> and <u>reinforce CICS TS</u> as the most cost effective platform for their business**
- **Technical Objective (at least their hope)**
  - **Streamline System z and CICS integration paths**
  - **Reduce the CPU burn (GP) associated with socket applications and infrastructure**

SHARE in San Francisco

Perfect R&D Situations

- Well defined business objectives
- An initial theory as to what the technical issues might be
- Strong in-house CICS talent
- Load testing infrastructure in place
- Good CICS tools on hand
- Test LPAR/region available
- Had a spare cubicle

Timing Was Opportune

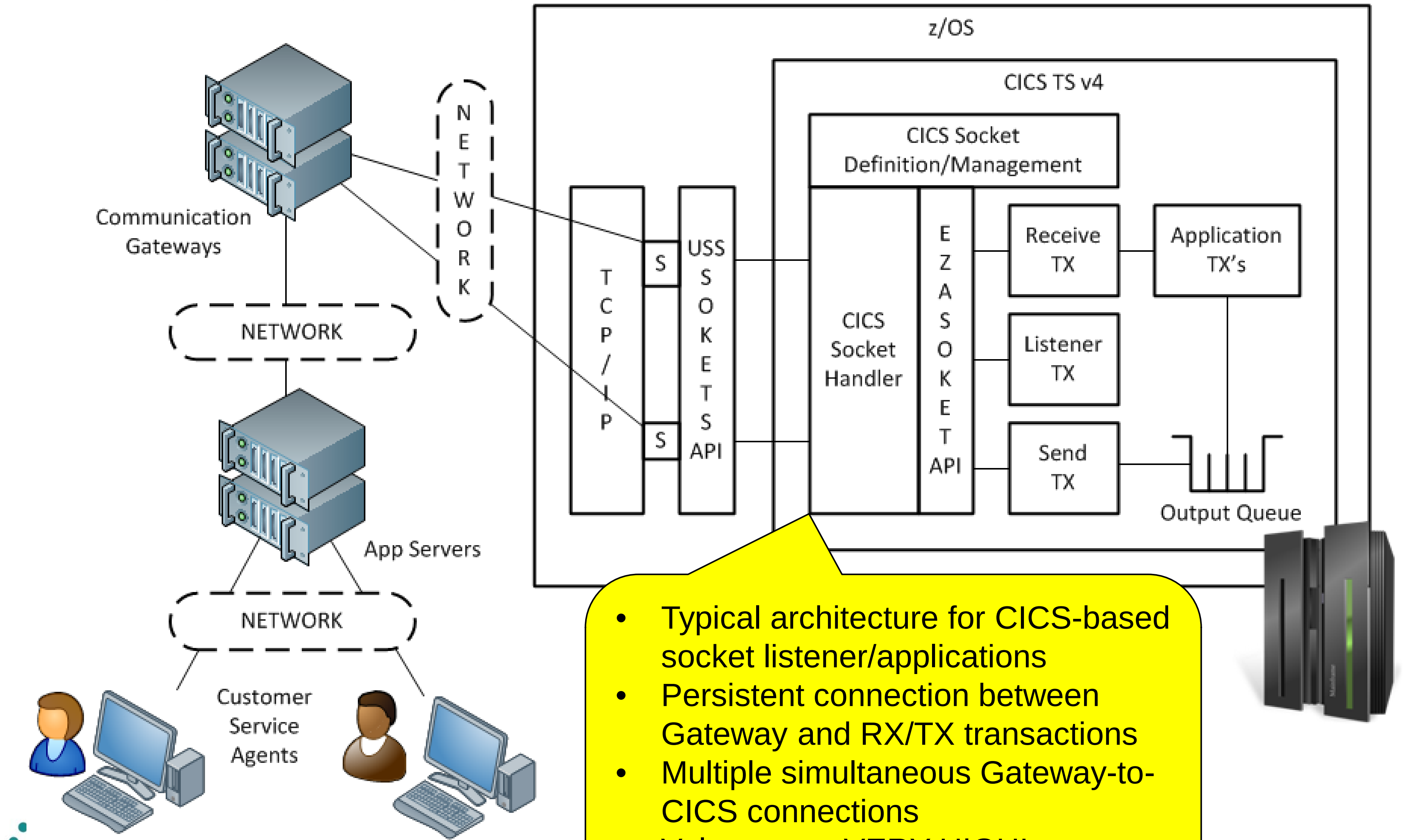
- **Customers were continuing to state their concern about doing more for less**
- **We had just delivered zIIP-enabled versions of our products, and our heads were filled with fun facts related to:**
  - **z/OS, USS, LE, WLM, SRBs, zIIP**
  - **CICS TS v4 Open Transaction Environment**
  - **Sockets**

- **Other factors:**
  - **We are zealots regarding integration of CICS apps/data as part of web/cloud-based infrastructure**
  - **We are committed to delivering functionality <u>under</u> CICS**
  - **I didn't want to stop writing code (zIIP project was too fun)**

Cut to the Chase

- What we learned was surprising and the results were unexpected (in a good way)
- We ended up exploiting CICS TS v4 OTE and z/OS to create a solution
- I want this to be knowledge you can use:
  - The approach is generally applicable to any CICS customer who has socket apps
  - The higher your volume, the more it matters
- Yes… I'm "a vendor" but please forget that for now – I'm speaking as a CICS developer

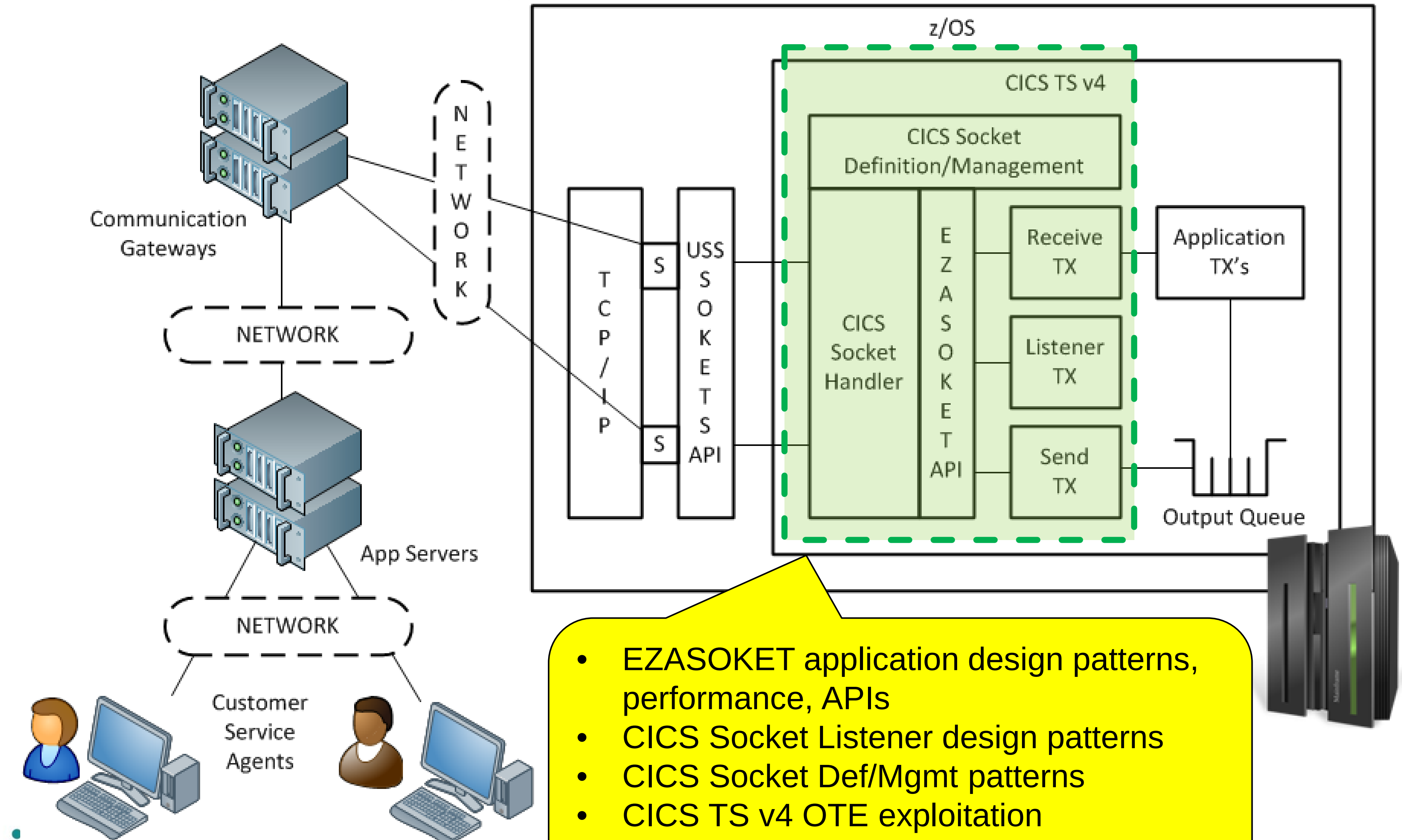
Customer A - Initial Conditions



- Typical architecture for CICS-based socket listener/applications
- Persistent connection between Gateway and RX/TX transactions
- Multiple simultaneous Gateway-to-CICS connections
- Volume was VERY HIGH!

SHARE in San Francisco

Research Focus



- EZASOKET application design patterns, performance, APIs
- CICS Socket Listener design patterns
- CICS Socket Def/Mgmt patterns
- CICS TS v4 OTE exploitation
- z/OS USS exploitation

CICS Socket Support

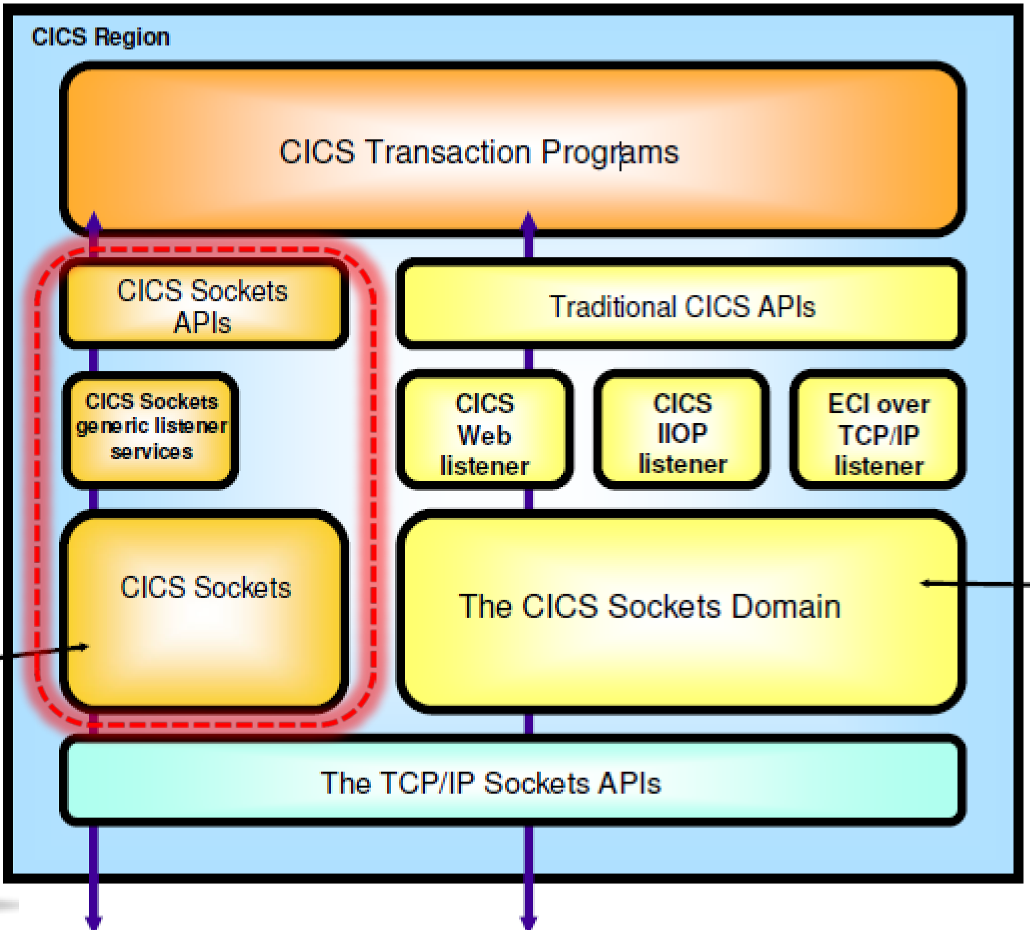
- Provided as part of z/OS Communications Server
- What it includes:
  - Socket APIs
    - C language API
    - Sockets Extended API (aka, EZASOKET or EZACICSO)
    - Original COBOL API (aka, EZACICAL)
  - Listeners: standard and enhanced (i.e., CSKL); or user-written
  - Definition and management components (e.g., EZAO)
- A <u>well-documented workhorse</u>, but…
- It's been around a long time (circa 1992)
- Older than CICS OTE
  - Thus… much of it's original architecture
- Reengineered to support OTE
  - But… the general approach of the original architecture persisted

Thus, I'm NOT referring to CICS TS features which use the CICS Sockets Domain.

CICS Sockets ≠ Sockets Domain

A CICS Sockets transaction has direct access to the TCP/IP socket and can issue native sockets calls to receive and send data over the socket.

These services are based on the Sockets Extended sockets APIs (provided by Communications Server)



A CICS Sockets Domain transaction does not have direct access to the socket, but communicates with CICS Sockets Domain services to receive a request and to send a reply over a socket.

The listeners are the 'servers' as seen from a TCP/IP perspective.

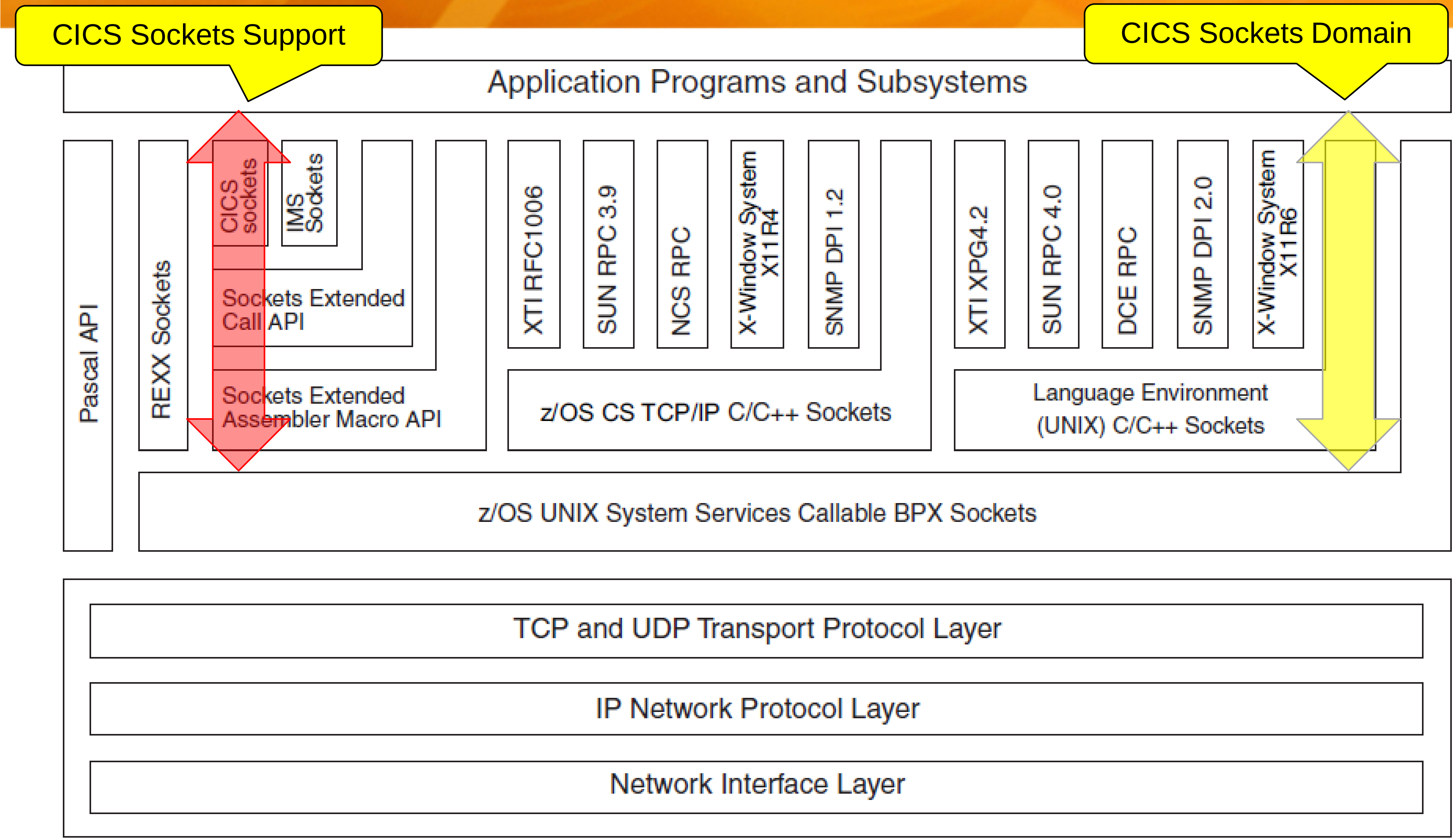
These services are based on the UNIX System Services C/C++ sockets API (provided by Language Environment) and the UNIX System Services callable APIs

A conversational model - or a request/reply model

A request/reply model

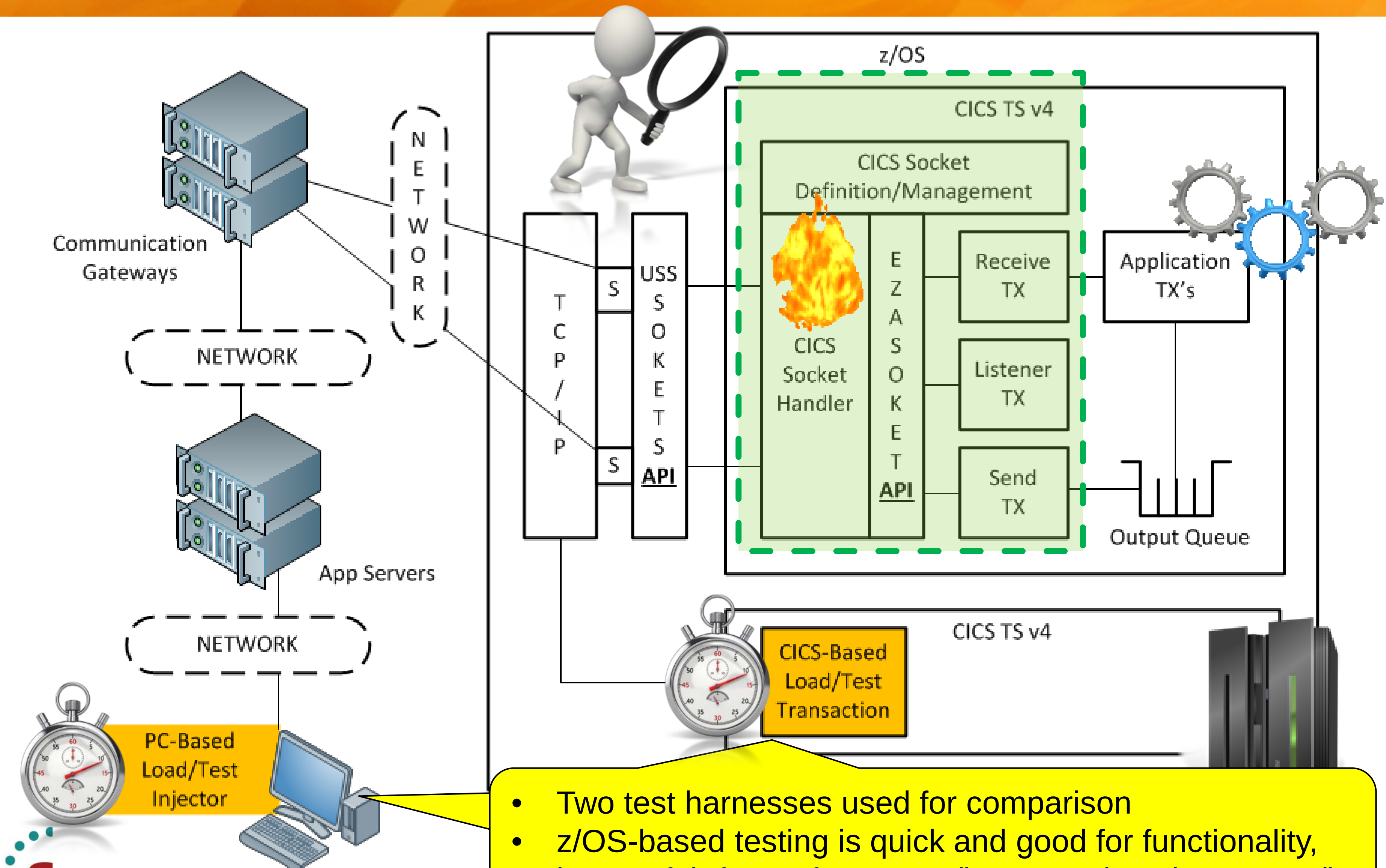
SHARE in San Francisco

CICS Sockets Pathway



z/OS Communications Server, IP Sockets Application Programming Interface Guide and Reference

Test Methodology



- Two test harnesses used for comparison
- z/OS-based testing is quick and good for functionality, but not fair for performance (hyper-sockets is too good)

Standard Test Cycle

- Each test cycle caused the gateway to:
  - Open 2 sockets via Listener TX
  - Send/Receive TXs started to handle socket I/O
  - Generate 2,500 request-response iterations (no delays)
  - Each request caused a LINK to a customer program
  - Bytes in/out modeled for average production use case
- Benchmarks run:
  - 1 concurrent test cycle
  - 5 concurrent test cycles
    (10 sockets and 12,500 iterations)

Selected to keep total region-level CPU use to a manageable level on test LPAR

- Objectives:
  - Measure region-level CPU burn for various configurations
  - Differentiate between CPU burn associated with Socket apps and Socket infrastructure

Tooling Developed

- **It's difficult to get a snapshot of a CICS region's total resource consumption that is:**
  - **high-resolution (microseconds)**
  - **low-overhead**
  - **Immediate**
  - **Includes zIIP and zAAP**
- **Ended up developing two tools:**
  - **A CICS transaction to provide a summary of MVS ASSB timers (HBZT)**
  - **A CICS XMNOUT exit to log transaction metrics via WTO**
- **The combination allowed us to:**
  - **drive testing fast**
  - **quickly assess results from all angles**
- **Special thanks to:**
  - **Larry Lawler (UNICOM)**
  - **Ed Jaffe (Phoenix Software)**
- **For info on HBZT, see me after session**

CPU Measurement (HBZT)

Session B - Gamma - [24 x 80]

File Edit View Communication Actions Window Help

```
              CPU USAGE FOR ADDRESS SPACE: ASID=003F,APPLID=CICSA

ACTUAL values at 2012/07/31 23:39:06.068080

ASSB 'Programming Interface' values (*=not normalized):
ASSBASST................ 00:00:00.000000  Additional SRB Service Time
ASSBPHTM................ 00:00:00.385582  Preemptable-class SRB Time
ASSBPHTM_BASE........... 00:00:00.000000  ASSBPHTM at end of previous jobstep
ASSB_IFA_PHTM........... 00:00:00.000000  zAAP-only equiv of ASSBPHTM
ASSB_ZIIP_PHTM.......... 00:00:00.378829  zIIP-only equiv of ASSBPHTM
ASSB_SRB_TIME_ON_CP..... 00:00:00.288598  CP time in SRB mode
ASSB_TASK_TIME_ON_CP.... 00:00:02.473032  CP time in task mode
ASSB_TIME_IFA_ON_CP..... 00:00:00.000000  zAAP time on CP (non-enclave)
ASSB_TIME_ZIIP_ON_CP.... 00:00:00.000000  zIIP time on CP (non-enclave)
ASSB_TIME_ON_IFA........ 00:00:00.000000* zAAP time (non-enclave)
ASSB_TIME_ON_ZIIP....... 00:00:00.000000* zIIP time (non-enclave)
Other ASSB values of interest:
ASSB_ENCT............... 00:00:00.006224  Std CP time (enclave)
ASSB_IFA_ENCT........... 00:00:00.000000  zAAP time (enclave)
ASSB_ZIIP_ENCT.......... 00:00:00.378829  zIIP time (enclave)

        This program may be freely copied and used in object code form.
      Copyright (c) 2011 HostBridge Technology, LLC -- www.hostbridge.com
  ENTER=Update, PF1=Baseline, PF2=Toggle Mode, PF5=Update+Baseline, CLEAR=Exit
MA        B                                                            01/001
```

ACTUAL mode upon entry

CPU Measurement (HBZT)

Session B - Gamma - [24 x 80]

File Edit View Communication Actions Window Help

```
             CPU USAGE FOR ADDRESS SPACE: ASID=003F,APPLID=CICSA

DELTA values from 2012/07/31 23:37:56.619510 to 2012/07/31 23:39:06.068080

ASSB 'Programming Interface' values (*=not normalized):
ASSBASST................ 00:00:00.000000  Additional SRB Se
ASSBPHTM................ 00:00:00.370396  Preemptable-class SRB Time
ASSBPHTM_BASE........... 00:00:00.000000  ASSBPHTM at end of previous jobstep
ASSB_IFA_PHTM........... 00:00:00.000000  zAAP-only equiv of ASSBPHTM
ASSB_ZIIP_PHTM.......... 00:00:00.369743  zIIP-only equiv of ASSBPHTM
ASSB_SRB_TIME_ON_CP..... 00:00:00.145086  CP time in SRB mode
ASSB_TASK_TIME_ON_CP.... 00:00:01.083711  CP time in task mode
ASSB_TIME_IFA_ON_CP..... 00:00:00.000000  zAAP time on CP
ASSB_TIME_ZIIP_ON_CP.... 00:00:00.000000
ASSB_TIME_ON_IFA........ 00:00:00.000000* zAAP time (non-e
ASSB_TIME_ON_ZIIP....... 00:00:00.000000* zIIP time (non-e
Other ASSB values of interest:
ASSB_ENCT............... 00:00:00.000652  Std CP time (enclave)
ASSB_IFA_ENCT........... 00:00:00.000000  zAAP time (enclave)
ASSB_ZIIP_ENCT.......... 00:00:00.369743  zIIP time (enclave)

        This program may be freely copied and used in object code form.
      Copyright (c) 2011 HostBridge Technology, LLC -- www.hostbridge.com
  ENTER=Update, PF1=Baseline, PF2=Toggle Mode, PF5=Update+Baseline, CLEAR=Exit
MA      B                                                               01/001
```

PF2 toggles mode

Immediate view of ASSB values

CPU Measurement

Session B - Gamma - [24 x 80]

File Edit View Communication Actions Window Help

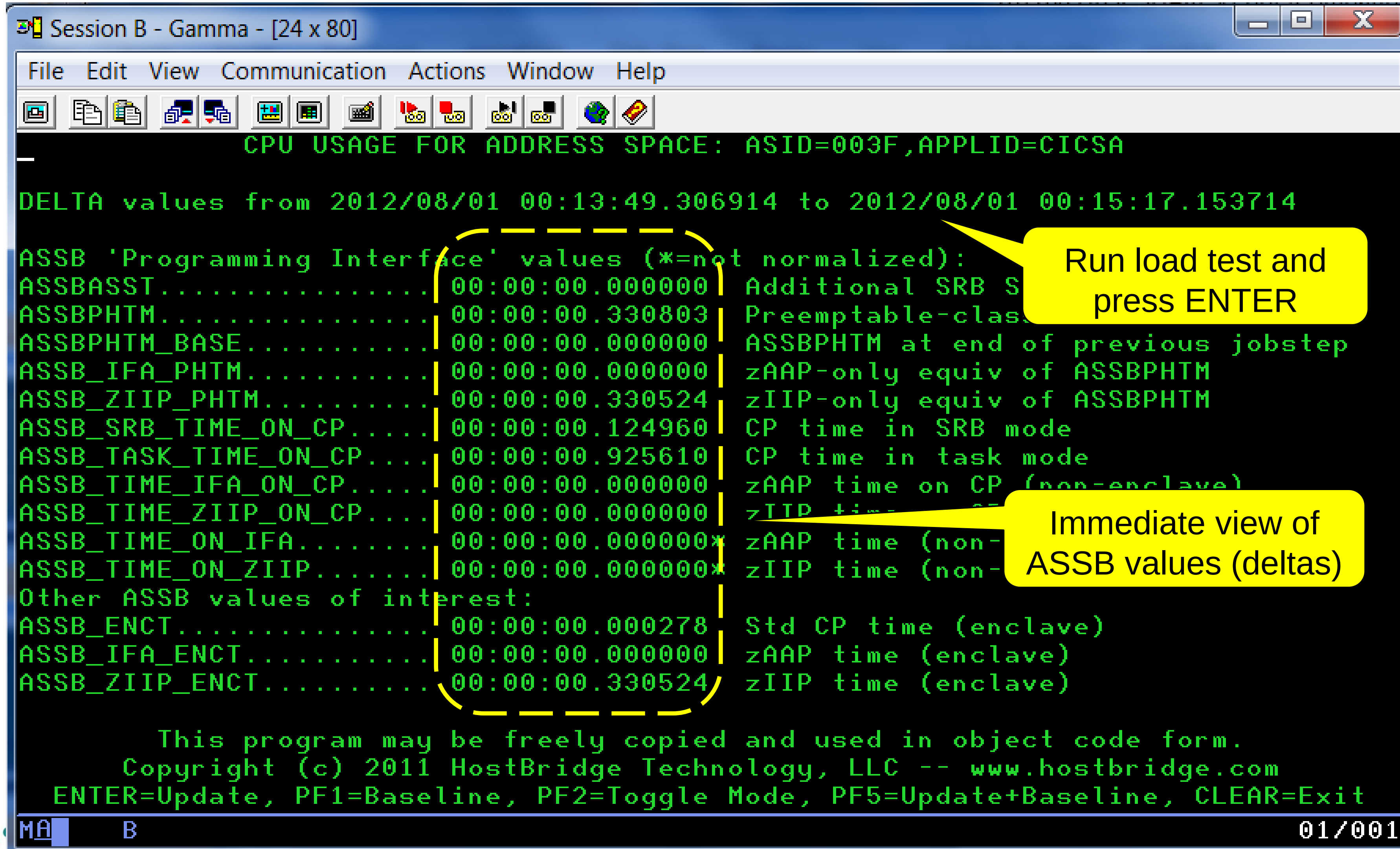
```
              CPU USAGE FOR ADDRESS SPACE: ASID=003F,APPLID=CICSA

DELTA values from 2012/08/01 00:13:49.306914 to 2012/08/01 00:13:49.306914

ASSB 'Programming Interface' values (*=not normalized):
ASSBASST................ 00:00:00.000000  Additional SRB Se
ASSBPHTM................ 00:00:00.000000  Preemptable-class SRB Time
ASSBPHTM_BASE........... 00:00:00.000000  ASSBPHTM at end of previous jobstep
ASSB_IFA_PHTM........... 00:00:00.000000  zAAP-only equiv of ASSBPHTM
ASSB_ZIIP_PHTM.......... 00:00:00.000000  zIIP-only equiv of ASSBPHTM
ASSB_SRB_TIME_ON_CP..... 00:00:00.000000  CP time in SRB mode
ASSB_TASK_TIME_ON_CP.... 00:00:00.000000  CP time in task mode
ASSB_TIME_IFA_ON_CP..... 00:00:00.000000  zAAP time on CP (non-enclave)
ASSB_TIME_ZIIP_ON_CP.... 00:00:00.000000  
ASSB_TIME_ON_IFA........ 00:00:00.000000* zAAP time (non-e
ASSB_TIME_ON_ZIIP....... 00:00:00.000000* zIIP time (non-e
Other ASSB values of interest:
ASSB_ENCT............... 00:00:00.000000  Std CP time (enclave)
ASSB_IFA_ENCT........... 00:00:00.000000  zAAP time (enclave)
ASSB_ZIIP_ENCT.......... 00:00:00.000000  zIIP time (enclave)

        This program may be freely copied and used in object code form.
      Copyright (c) 2011 HostBridge Technology, LLC -- www.hostbridge.com
  ENTER=Update, PF1=Baseline, PF2=Toggle Mode, PF5=Update+Baseline, CLEAR=Exit
MA      B                                                              01/001
```

PF1 resets baseline

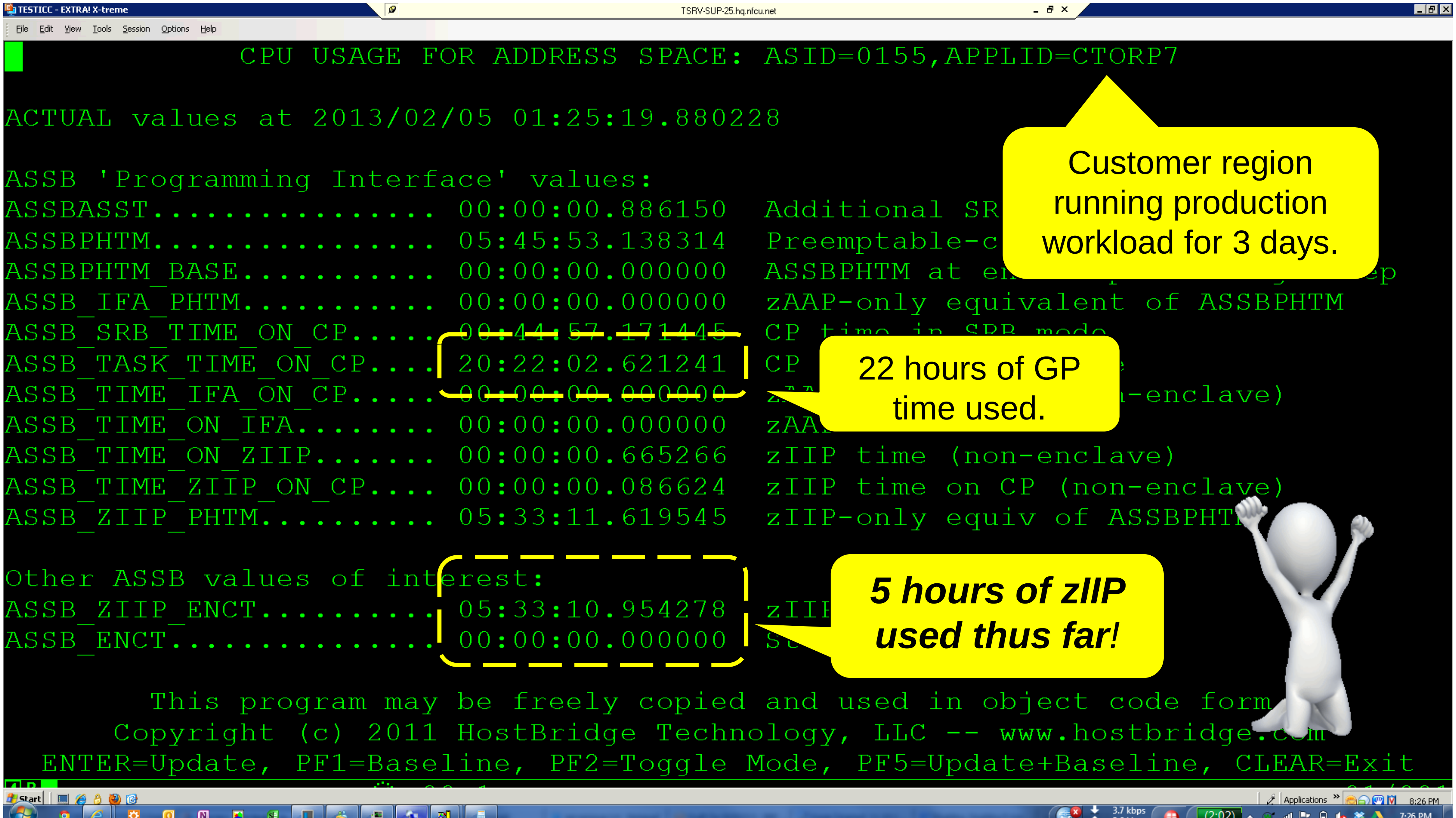
All delta values now zero

SHARE in San Francisco

CPU Measurement



CPU Measurement

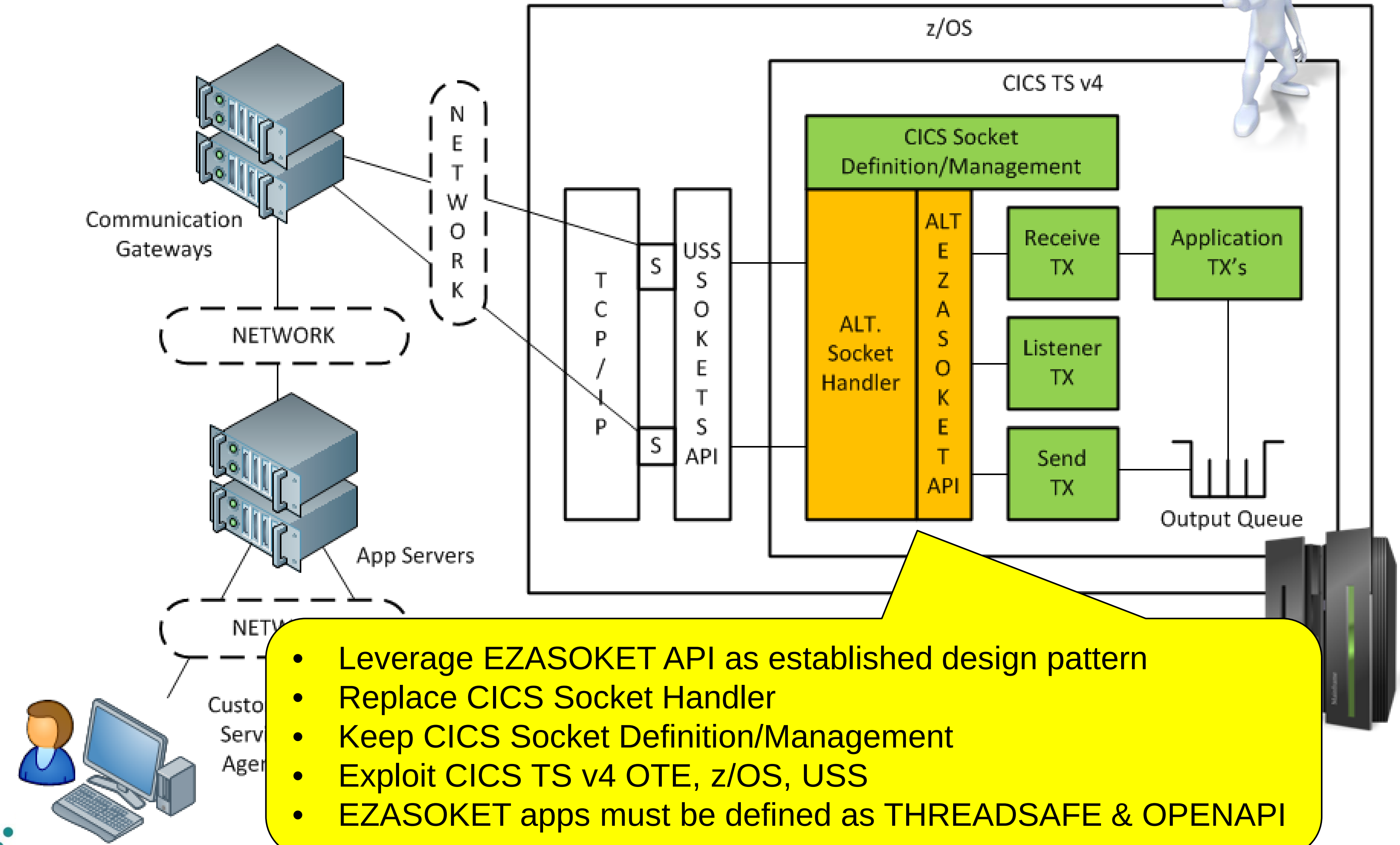


SHARE in San Francisco

Where the Data Led Us

- **Under volume testing, the CPU burn associated with the CICS Sockets Support was measurable and linear (confirmed customer's theory)**
- **I won't characterize it as "high" or "low" because the only thing that mattered was whether it could be lower (or not so linear)**
- **Thus, we began to:**
  - **Isolate various components and their impact**
  - **Consider how to provide alternative functionality (but complimentary to CICS TS)**
- **Low hanging fruit seemed to be CICS Socket Handler (via EZASOKET API)**

Customer A - Solution 1



- Leverage EZASOKET API as established design pattern
- Replace CICS Socket Handler
- Keep CICS Socket Definition/Management
- Exploit CICS TS v4 OTE, z/OS, USS
- EZASOKET apps must be defined as THREADSAFE & OPENAPI

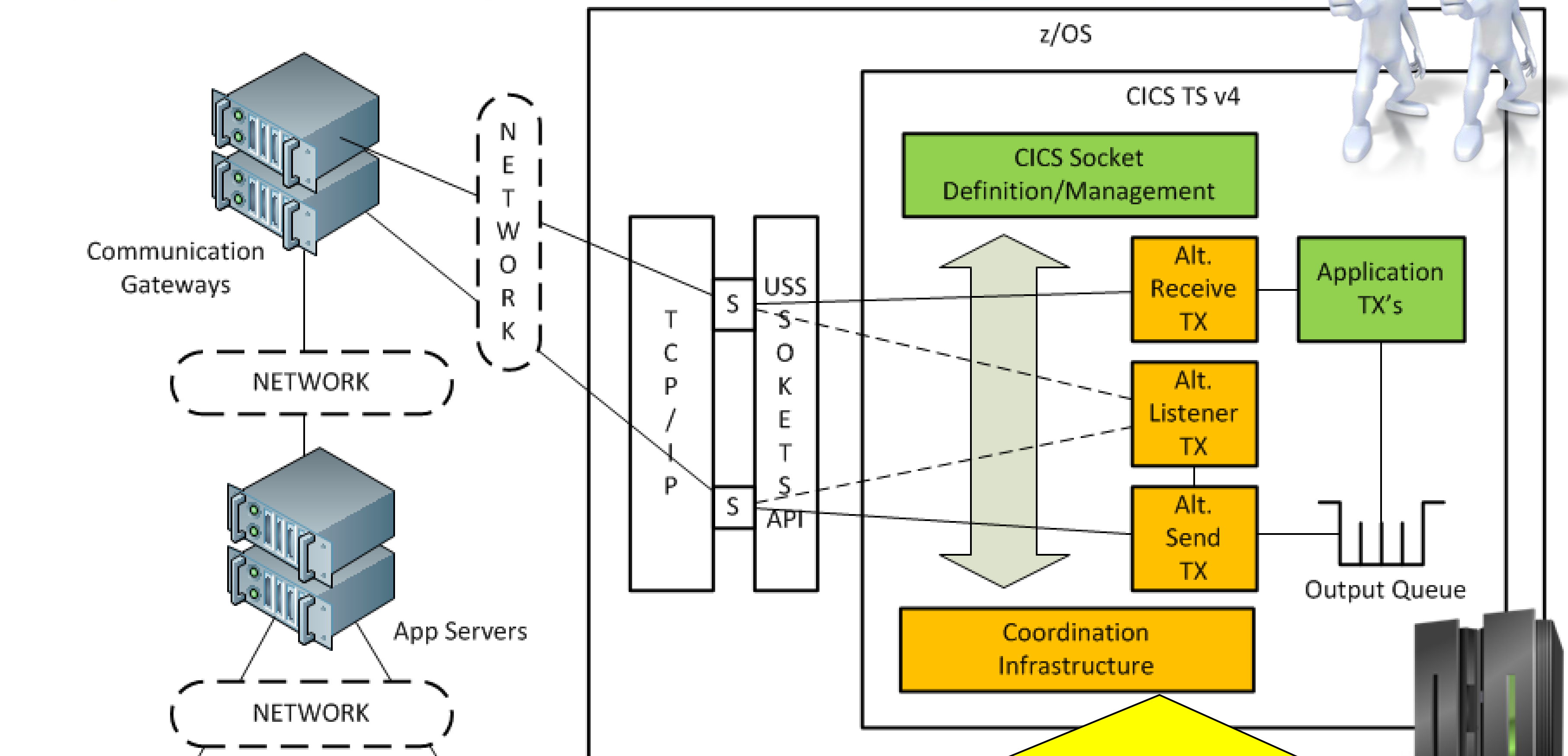
SHARE in San Francisco

Solution 1 Assessment

- **Good…**
  - **The Alt. Socket Handler lowered GP CPU burn associated with Socket I/O**
  - **All it required was a re-link of apps that used EZASOKET API (with alternate load module)**
  - **Transparent to existing user-written Listeners, Sender and Receiver TXs**
- **However…**
  - **EZASOKET API emulation seemed to be a bit of needless overhead (e.g., parameter marshaling and transformation)**
  - **zIIP enablement opportunity wasn't optimal due to task switching**
- **But wait…**
  - **The design patterns for CICS-based Listeners, Receivers and Senders are fairly common**

SHARE in San Francisco

Customer A - Solution 2



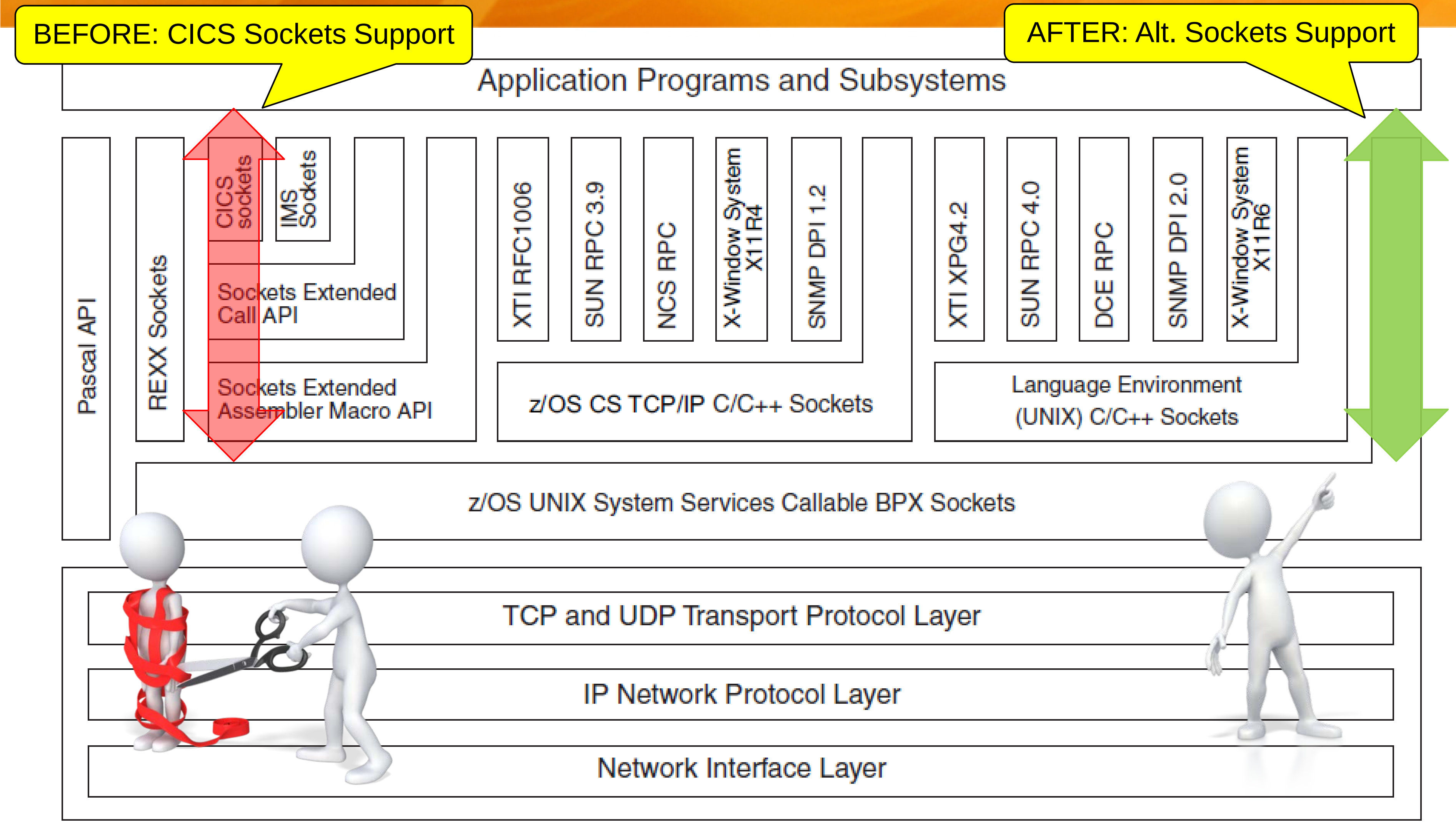
- Replace Listener, Receive, Send TX with equivalent/generic alternatives
- Eliminate EZASOKET API as a design pattern
- Keep CICS Socket Definition/Management
- Exploit CICS TS v4 OTE, z/OS, USS, zIIP

Solution 2 Assessment

- Very Good…
  - GP CPU burn associated with Socket I/O went down further
  - EZASOKET API emulation eliminated (all components use native sockets)
  - Transparent to the customer's applications
  - CICS Socket definition/management leveraged
    - EZAO still used to Configure, Start, or Stop Listeners
- zIIP enablement potential maximized
  - Minimal task switching
  - Customer application code not zIIP enabled (per IBM-ISV T&C's)

SHARE in San Francisco

Pathway - Old vs. New



z/OS Communications Server, IP Sockets Application Programming Interface Guide and Reference

Test Results

TEST: concurrent instances=1; total requests=2500							
Standard Socket Infrastructure (EZA-based)							
	Send TX		**Recv TX**		**Total**		
	(GP)		**(GP)**				
1	140714		332702				
2	138355		317988				
3	141509		336017				
Avg	140193		328902		469095		
Alt. Socket Infrastructure (ziip=n)							
	Alt Send TX		**Alt Recv TX**				
	(GP)		**(GP)**				
1	128676		285711				
2	125736		271014				
3	119938		240784				
Avg	124783		265836		390620	-17%	
Alt. Socket Infrastructure (ziip=y)							
	Alt Send TX		**Alt Recv TX**				
	(GP)	**(zIIP)**	**(GP)**	**(zIIP)**			
1	94956	48131	165486	114161			
2	94766	48759	165751	114349			
3	94049	47391	159752	111208			
4	94522	47390	155531	107856			
Avg	94573		161630		256203	-34%	**-45%**

All times in microseconds

% reduction - Old vs. New w/o zIIP

% reduction - New w/o zIIP to New w/ zIIP

% reduction - Old vs. New w/ zIIP

SHARE in San Francisco

Test Results (w/ Concurrency)

TEST: concurrent instances=5; total requests=12500							
Standard Socket Infrastructure (EZA-based)							
	Send TX		**Recv TX**		**Total**		
	(GP)		**(GP)**				
1	609880		1226658				
2	614881		1234086				
3	617669		1259704				
Avg	614143		1240149		1854293		
Alt. Socket Infrastructure (ziip=n)							
	Alt Send TX		**Alt Recv TX**				
	(GP)		**(GP)**				
	491684		782429				
	496651		780384				
	502901		804619				
Avg	497079		789144		1286223	-31%	
Alt. Socket Infrastructure (ziip=y)							
	Alt Send TX		**Alt Recv TX**				
	(GP)	**(zIIP)**	**(GP)**	**(zIIP)**			
	417841	198962	657107	424739			
	417388	194910	613641	401113			
	409281	194758	618252	399555			
	410077	193542	600015	397736			
Avg	413647		622254		1035901	-19%	**-44%**

All times in microseconds

% reduction - Old vs. New w/o zIIP

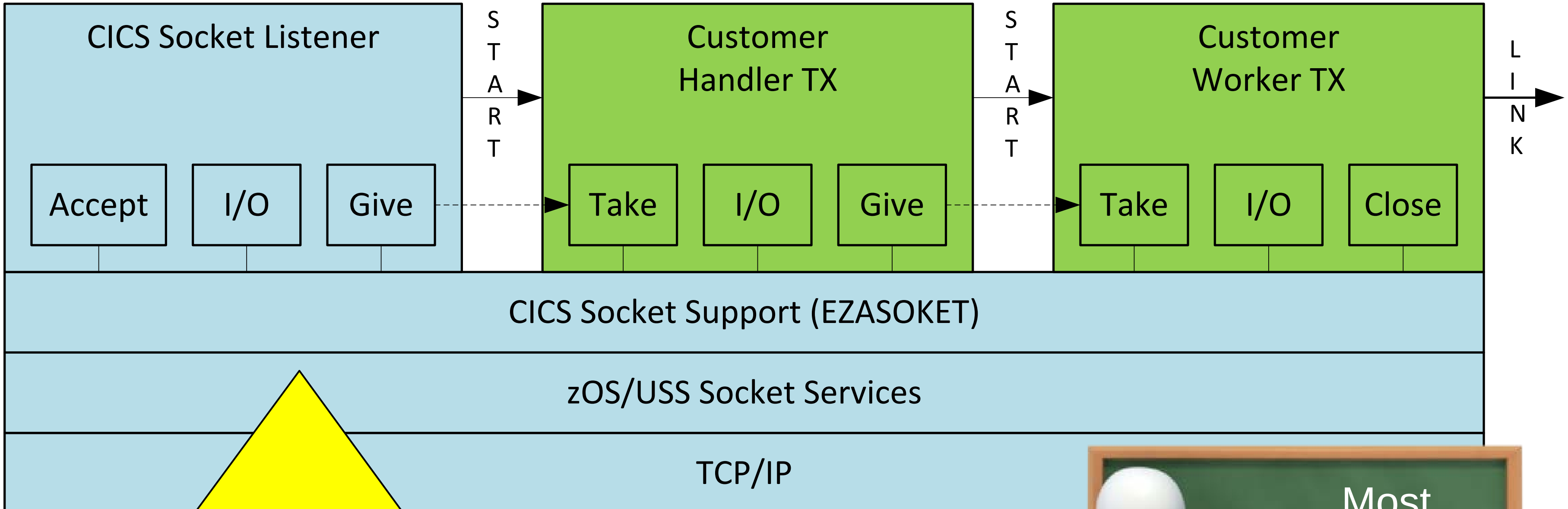
The TCP/IP stack seems to get more efficient the harder you load it

% reduction - New w/o zIIP to New w/ zIIP

% reduction - Old vs. New w/ zIIP

Customer B - Initial Conditions

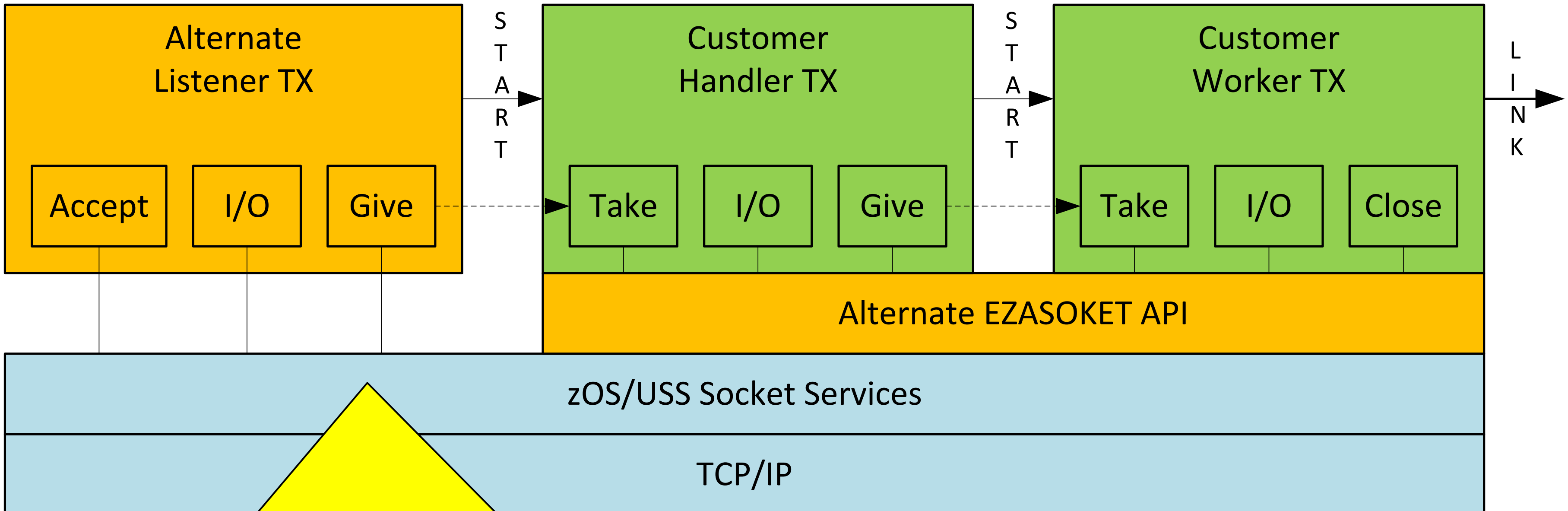
(Infrastructure outside System z similar to customer A)



- A single socket is used for both sending and receiving
- CICS Socket Listener connection requests
- Handler TX validates and categorizes requests
- Worker TX is long-lived
- Work requests serviced by LINKed-to programs

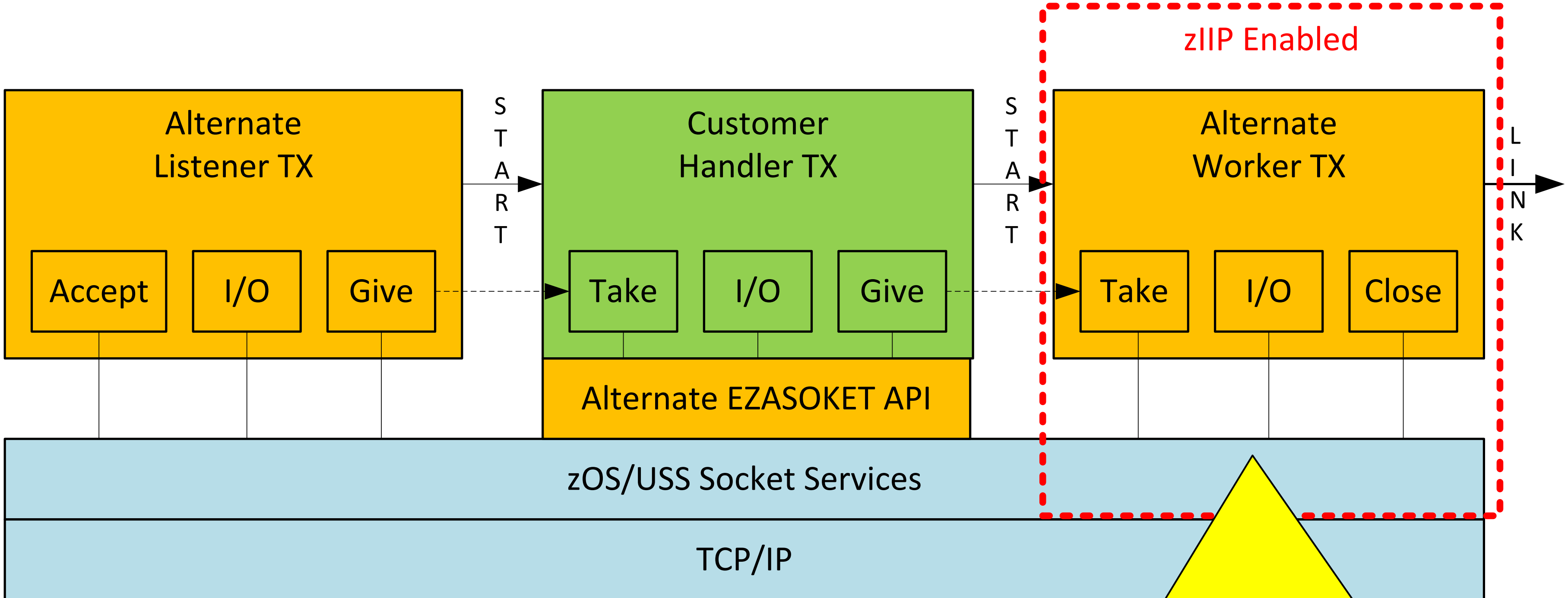
SHARE in San Francisco

Customer B - Solution 1



- Replace CICS Socket Listener
- Leverage EZASOKET API, but change implementation
- Keep CICS Socket Definition/Management
- Exploit CICS TS v4 OTE, z/OS, USS
- Programs defined as THREADSAFE & OPENAPI
- Nice GP CPU reduction, but no practical opportunity to exploit zIIP

Customer B - Solution 2

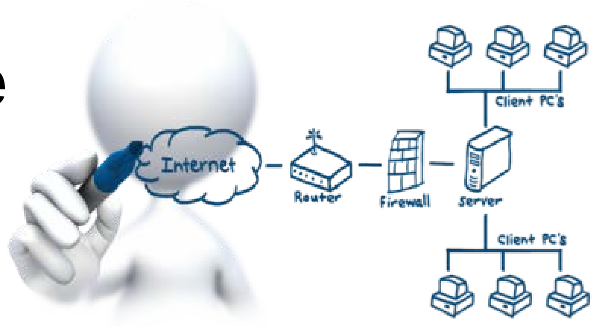


- Builds on Solution 1
- Replace Worker with generic alternative
- Eliminate EZASOKET API in Worker TX
- Allows zIIP exploitation by Worker

SHARE in San Francisco

Test Procedure

- **Each configuration was tested using a common benchmark from a distributed system:**
  - **Open socket to Worker TX (via Listener/Handler)**
  - **Send 5,000 requests (causing the same number of LINKs and responses)**
  - **Close socket**
- **Test constructed to isolate the actual GP CPU costs/savings for socket-related processing per request**
- **Test <u>not</u> constructed to determine an average percent reduction in GP CPU per request**



SHARE in San Francisco

Test Results and Calculations

	GP CPU Sec	zIIP CPU Sec
Initial Config	27.951	0.000
Solution 1	26.943	0.000
Solution 2	23.436	0.427

Averages from multiple tests of each configuration

27.951	Initial Config runs entirely on the GP (all socket I/O and app logic)
(26.943)	Solution 1 runs entirely on the GP, but measures the effect of replacing the EZASOKET API
1.008	Difference = Estimated GP CPU savings to handle socket I/O for 5,000 requests via Customer Worker TX

27.951	Initial Config runs entirely on the GP (all socket I/O and app logic)
(23.436)	Solution 2 runs socket I/O on the zIIP, and app logic on GP
4.515	Difference = Estimated GP CPU savings to handle socket I/O for 5,000 requests via Alternate Worker TX (and on zIIP)

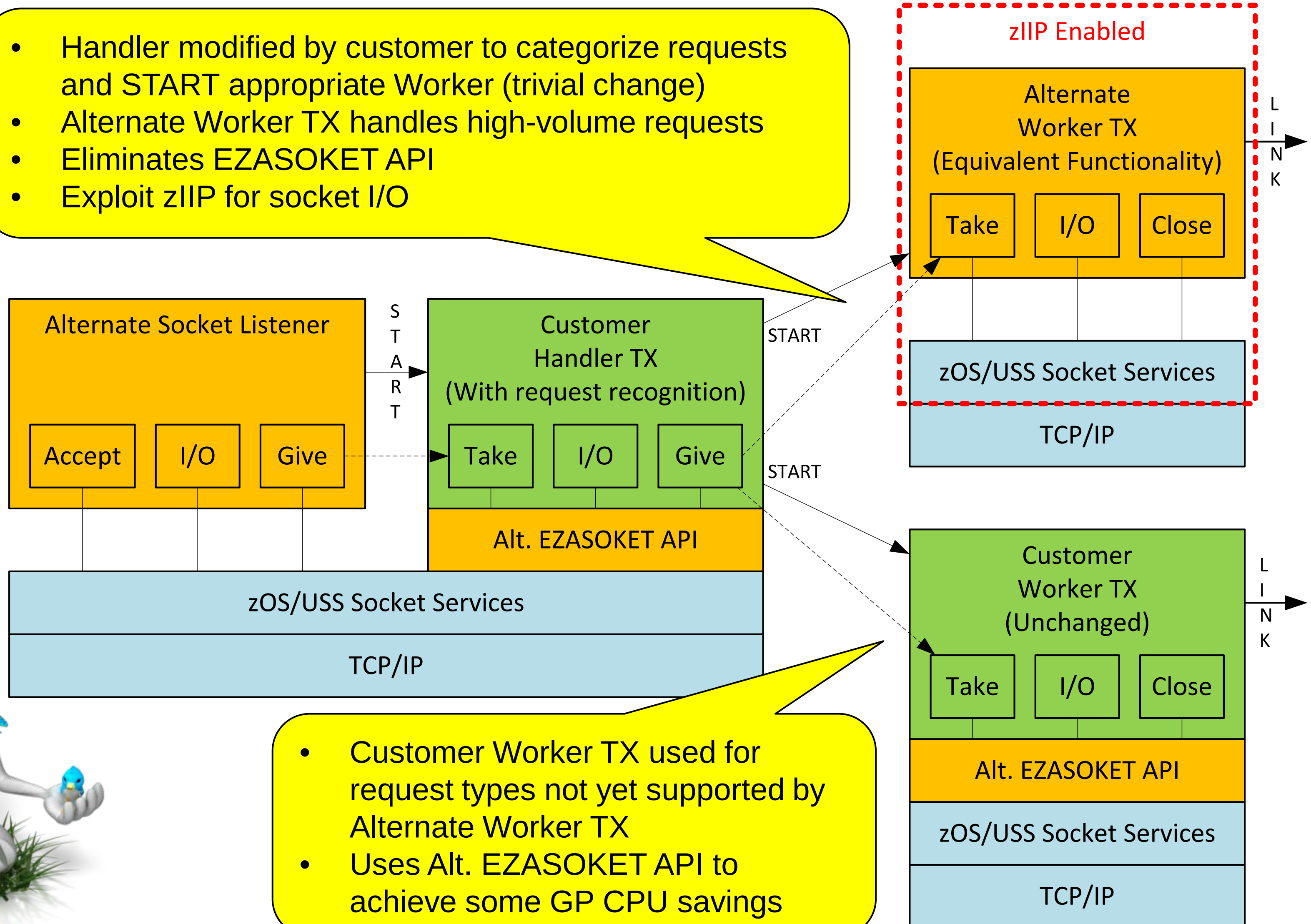
Value Proposition Model

- **What mattered most to the customer was processing new workload efficiently during their peak 4 hour period**
- **Assume:**
  - **5 million TX in max 4 hr period**
  - **20% processed via Alt. Worker TX**

5,000,000	Peak 4 hour transaction volume
20%	% of TX processed via Alt. Worker
1,000,000	TX processed via Alt. Worker
80%	% of TX processed via Std. Worker
4,000,000	TX processed via Std. Worker
903	Est. GP CPU Reduction for Alt. Worker (seconds)
807	Est. GP CPU Reduction for Std. Worker (seconds)
1,710	Total Est. GP CPU Seconds Reduced
28.49	Total Est. GP CPU Minutes Reduced during Peak Period

Customer B – Optimal Solution

- Handler modified by customer to categorize requests and START appropriate Worker (trivial change)
- Alternate Worker TX handles high-volume requests
- Eliminates EZASOKET API
- Exploit zIIP for socket I/O



- Customer Worker TX used for request types not yet supported by Alternate Worker TX
- Uses Alt. EZASOKET API to achieve some GP CPU savings

Summary

- CICS Socket Support has been a workhorse for a long time -- it's earned it's keep!
- CICS TS Open Transaction Environment continues to evolve and permit new opportunities for customers and ISV's -- thank you Hursley Lab
- An example is the Alt. Socket Support described in this presentation
- This approach is applicable to any customer who relies heavily on CICS Socket Support
  - zIIP support can only be provided by a licensed ISV
- <u>You can substantially reduce GP CPU usage associated with CICS socket applications</u>
- Oh… and the customers were very pleased