INNOVATION: THE INDIGO CHILD OF SYSPLEX DISTRIBUTION SHARING OUR EVOLUTION

Sigfrido Perdomo
sperdomo@dtcc.com
The Depository Trust & Clearing Corporation

Tuesday, Aug 3, 2010
Session 6913

THE INDIGO CHILD OF SYSPLEX DISTRIBUTION SHARING OUR EVOLUTION

SHARE
Technology · Connections · Results

"STACK'S IDENTITY – STATIC TO DYNAMIC"

SYSPLEX Distribution
For Applications with Data Sharing Capacities &
Application IP Centricity
Has the Potential to
Double One's:
Availability
Reliability
Scalability
Resource Balance
Optimizing Performance

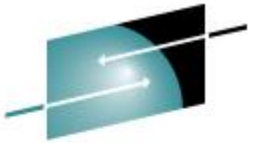
However

Not so easy for Legacy Applications that
Target the Stack's Identity
Over 85% of our clients do so

Thus the birth of the Indigo Child Of SYSYPLEX DISTRIBUTION:
STATIC TO DYNAMIC

THE INDIGO CHILD

PRELUDE: TCPIP / SYSPLEX



SHARE
Technology · Connections · Results

Topics: Some of the topics we'll discuss are –

PRELUDE To The "Indigo Child":

TCPIP / SYSPLEX Distribution

- Dynamic VIPAs & Their Interrelations
- VIPARANGE – (AKA VIPA Follow) & Applications
  - Special considerations with VIPARANGE & BIND(s)
  - IP Application Addresses & SUBNETS / DVIPA models In OMPROUTE
  - Connectivity Tester - New Developed Code (CONNTEST – VS – Telnet)
- SYSPLEX Distributor & Applications
  - The Ephemeral Structure - EZBEPORT
  - BASEWLM / SERVERWLM & Relevant Profile statements
  - VIPAROUTE & XCF Relief
- SYSPLEX Distribution Confirmation
  - Confirmation & Simulation methods
  - Some DVIPA commands
- SYSPLEX Distribution Broken Symmetry – F/W's And Statefulness
  - The Cause
  - The Resolution – Policy Based Routing.
- IPL Relief Order
- Deeper Levels Of Availability
  - VTAM & TCPIP – ARM'd
  - OMPROUTE – ARM Wrapped

PRELUDE: TCPIP / SYSPLEX

Topics: Some of the topics we'll discuss are -

TCPIP / Innovation: Static – To - DYNAMIC

- **Why Static – To - Dynamic**
- **The Goal**
- **Why not NAT to a different IP address versus converting the Stack's Identity / Primary Interface, from a static VIPA to A Dynamic VIPA.**

PHASE ONE: Stack's Identity / Primary Interface Becomes a DYNAMIC VIPA →95% There

- **Parameters & Listener Applications**
- **Some Hurdles To Over Come**
  - **PINGS (ICMP) & TRACERTE (UDP)**
  - **NSLOOKUP / DNS**
  - **Firewall Updates for OSA(s) Subnets**
  - **Connectivity Tester - (Still In The Mix)**

PHASE Two: Stack's Identity / Primary Interface becomes a Distributed Dynamic VIPA –

Related Parameters

- **Special Case Of DB2 & MQ**
  - **New DB2 IP Application Centric Infrastructure**
  - **Code to track Migration**
- **The SYSPLEX Integrity Checker - (Rolling IPL(s) and/or LPAR / Stack failures – New Developed Code)**
- **The Stack's Identity becomes a Distributed Dynamic VIPA!!!!!! → 100% There!!!**
- **Sample of Exploitation**

TCPIP SYSPLEX Distribution–

DYNAMIC VIPA(s) & Their Interrelations

- ***VIPA Evolution:***

- IP address availability in z/OS can be viewed as **a progressive evolution**. **Static** VIPA eliminated the problem of an IP address being associated with a **single** networking hardware **point** of failure.

- **Dynamic** VIPA allows us to **move** such IP addresses in event of an application, TCP/IP stack, or LPAR failure.

- Dynamic **XCF** provides automatic generation of links to provide an IP **layer of connectivity** within the **SYSPLEX**

- The SYSPLEX **Distributor** can be viewed as a continued evolution of connectivity improvements. It is a combination of the high **availability** features of DVIPA and the **workload optimization** capabilities of **WLM**.

  - **Distribution is a function of a specific IP address associated with a Port that given applications Listen on.**
  - **The key is shared data.**

TCPIP SYSPLEX Distribution–

VIPARANGE (AKA VIPA FOLLOW) & APPLICATIONS

- ***VIPARANGE - With & Without Binds:***
- VIPARANGE is for **a range** of addresses, max of 256 or min of 1, relative to the **subnet mask association**.
- Subnet Mask within the VIPA DYNAMIC Block statement has **nothing to do with routing.**
- It only tells TCPIP what **addresses** are **eligible** to use for **dynamically** creating Device / Link statements and adding the addresses to the **bottom of the stack's HOME list.**
- The **key** to VIPARANGE is that it **floats** with a given application, as long as said statements are on both, (or more), participating stacks & **no** manual **intervention**. – It's a cousin to ACB modeling.
- **How** is VIPARANGE associated with an Application:
  - We **BIND** via the **PORT** statement.
  - An **APF** authorized **program** issues the **SIOCSVIPA ioctl**()
  - A **utility** like **MODDVIPA** is executed on behalf of the address space – A **pre-step** in the **JCL**.
- BIND - **Without VIPARANGE**. This is good if **two instances** of an application – **different job names**, are **listening on the same port, on the same stack, different IP addresses.** FTP would be an example.

TCPIP SYSPLEX Distribution–

DYNAMIC VIPA(s) & Their interrelations

- ***DYNAMIC VIPA:***

- Some thoughts regarding **DVIPA**, (VIPADEFINE / VIPABACKUP), **DDVIPA**, **VIPARANGE** / **BIND**, (AKA VIPAFOLLOW - that's a more meaningful name in my mind for VIPARANGE) and **BIND without** a VIPARANGE – and future directions.
- If there are **two application instances** in a **data sharing** capable environment, then **DDVIPA** - Distributed DVIPA. is a good option. This will provide load balancing and higher availability.
- If there is **one instance** of the application, then use **BIND / VIPARANGE** - let the VIPA follow the address space. (Either the address space BINDs or we do it for them). **Why this method versus VIPADEFINE / VIPABACKUP?**

  - **Example**: If we do a VIPADEFINE / VIPABACKUP - and BIND the address space, LPAR-1/ LPAR-2, the only way the address space can ever come up on LPAR-2 is if the **stack failed** on LPAR-1 or **OBEY(s**) were issued. (One obey to delete the VIPA on LPAR-1, one to bring it **back** to LPAR-1 from LPAR-2 – **OBEY issued on LPAR-1**).

  - My point is, given how workloads can move around, we'd be **out of the loop** if the VIPA, followed the address space.

TCPIP SYSPLEX Distribution–

VIPARANGE (AKA VIPA FOLLOW) & APPLICATIONS

- ***BINDS without VIPARANGE & Rolling IPLs Mix –***
- ***A Story:***

- In **rolling IPLS,** DVIPA , (define / backup),moved from LPAR-1 to LPAR-2.
- There were two **same-named applications / STC(s**) each on **LPAR-1 & LPAR-2**. They were listening on "**IN_ADDR_Any – 0.0.0.0." and same port.**
- It **just happened** that a down stream server had an automation script that kept trying to connect to the IP Address that was **normally advertised from LPAR-1**. It happened to connect when the **DVIPA was temporarily** on **LPAR-2** during the **rolling IPL**.
- The bind was successful on LPAR-2.
- LPAR-1 came back up and **took the DVIPA**. All was normal except that BIND on LPAR-2 was in an established state and of course the default for the DVIPA was 'NON-Disruptive". So **data was being gathered from LPAR-2 versus LPAR-1**

- ***The Solution:***

- We **bound** the address spaces to **LISTEN on the DVIPA**, via the PORT statement, relative to the **respective LPAR.**

TCPIP SYSPLEX Distribution–

VIPARANGE (AKA VIPA FOLLOW) & APPLICATIONS

- ***Special Considerations – VIPARANGE (Non-Disruptive) & BINDs –***
- ***The Scene – LPAR-1 & LPAR-2 part of the same PLEX:***
  - **(IF IT CAN HAPPEN, IT WILL HAPPEN!!!!)**

- An MQ region is **listening** and bound to LPAR-1.
- A special connectivity tester – **very quick test**, uses a **Socket BIND on LPAR-2**.
- Be aware that the IP address created as result of a bind, **can disappear and not be advertised** from the stack where say an MQ region is listening on.
- This can happen if on a participant stack **another BIND** for the **same IP address** occurs.
- If that happens, the **IP address is flagged with an "I"** under the home list of the original stack and now the same IP address will be **advertised from the participant stack.**
- When the application, e.g. connectivity tester, completes on the participant stack, then that IP address will **not be advertised at all.**
- **And All sessions relative to that DVIPA are Lost! ---- "I Hate When That Happens!!!"**
- The only way to get that IP address back to the original stack & advertised is **via another BIND.**
- This is the case if VIPARANGE with the default - (or stated) of **NONDISRUPTIVE** is in place.
- ***Solution:*** The way to prevent that is to code VIPARANGE **DISRUPTIVE**.
- **Weird right?**

TCPIP SYSPLEX Distribution–

VIPARANGE (AKA VIPA FOLLOW) & APPLICATIONS

- ***IP Application Addresses & Subnets / DVIPA Models / OMPROUTE***

- **Application centricity** is enabled via VIPARANGE & SYSPLEX Distribution (More on SD later)
- We have subsets of internal IP addresses used by applications that do **not** face the outside world
- We have subsets of external IP addresses used by applications that face **the outside world**.
- Our Environment has **Cross CEC / Site** SYSPLEX LPARS for optimal **availability & recovery**.
- Our external IP addresses have a 30 bit subnet mask – a **mini network per application**.
- Each SITE has a **unique Class C** subnet to facilitate IP advertisement for **site transference**:
  - SITE-1: 192,168.AA.xx - Applications **normal residence**
  - SITE-2: 192.168.BB.yy - Applications **normal residence**
- This scheme optimizes OMPROUTE & routing advertisements should single or multiple **applications relocate** from SITE-1 to SITE-2, while others remain on SITE-1.

TCPIP SYSPLEX Distribution–

VIPARANGE (AKA VIPA FOLLOW) & APPLICATIONS

- ***IP Application Addresses & Subnets / DVIPA Models / OMPROUTE***
- We Use **DVIPA Wildcard / Models in OMPROUTE** – “One statement keeps the all”:
  - 192.168.AA.* subnets 255.255.255.252
  - 192.168.BB.* subnets 255.255.255.252
- ***Some Examples:***
  - Model to be used for all DVIPA's in 192.168.AA.0 Network

```
OSPF_Interface
 IP_Address=192.168.AA.*
 Name=dummyAA1
 Subnet_Mask=255.255.255.252

OSPF_Interface
 IP_Address=192.168.BB.*
 Name=dummyBB1
 Subnet_Mask=255.255.255.252
```

TCPIP SYSPLEX Distribution–

VIPARANGE (AKA VIPA FOLLOW) & APPLICATIONS

- ***Connectivity Tester (CONNTEST) – VS – TELNET***

  (Another interesting Hurdle toward IP Application Centricity)

- ***The Problem:***
  - **TELNET** was used by our **NEO** Group to confirm **F/W & Connectivity.**
  - As we continued to evolve into Application IP Centricity, - where an **application listens on a specific** IP address, the ability to inject a source IP address assigned to an application is **mandatory**.
  - IP Application centricity **stymied** because TELNET has **no SRCIP injection** capability.
- ***The Resolution - CONNTEST:***
  - A new tool has been developed that **simulates an application call** when used in diagnostics to confirm connectivity, F/W rules, etc, from our mainframes.
  - A socket program to perform an application **connection** request
  - Has the capability to **inject** a Source IP
    - **3 parameters**: Destination IP, Port & SRCIP (if present)
    - If **third** parameter (SRCIP) is **not** passed, it then uses the Host Primary interface (**GET HOST_ID**)
    - Confirms the SRCIP that is passed is actually being **advertised** from the **stack** invoked.
    - Can be executed from **NETVIEW or TSO**.
    - See APENDIX for more details

TCPIP SYSPLEX DISTRIBUTION – THE Ephemeral Structure

Our Story Continues:

If We Build It… They Will Come:

- **Build The Structure – (**Like For VTAM**):**

SETXCF START,POL,POLNM=PPLX01,TYPE=CFRM

*** Confirm The Structure – MVS:**

D XCF,STR,STRNAME=EZBEPORT

*** Expect something like:**

STRNAME: EZBEPORT
DUPLEXING REBUILD

CONNECTION NAME	ID	VERSION	SYSNAME	JOBNAME	ASID	STATE
NETID.LPAR-1	01	000100D2	DI001	NET	0047	ACTIVE
NETID.LPAR-2	02	000200C9	D 003	NET	004A	ACTIVE

*** Confirm The Structure – VTAM:**

D NET,STATS,TYPE=CFS,STRNAME=EZBEPORT,LIST=ALL,SCOPE=ONLY

*** Expect something like:**

NETID.LPAR01 IS CONNECTED TO STRUCTURE EZBEPORT

*

LIST	DVIPA	SYSNAME	TCPNAME	# **ASSIGNED PORTS**
1	192.168.XX.1	TCPLPAR1	TCPPAR1	**2345**
1	192.168.XX.2	TCPLPAR2	TCPPAR2	1839

*** Confirm The Structure – VTAM:**

D NET,ID=ISTTRL,E

*** Expect something like:**

TRLE = ISTT0103 STATUS = ACTIV CONTROL = XCF
TRLE = ISTT0102 STATUS = ACTIV CONTROL = XCF

NOTE: z/OS 1.9 – 2 APARS: PK77995 / UK42896 & PK38073 –Ephemeral Port Leakage

TCPIP SYSPLEX DISTRIBUTION – BASEWLM / SERVERWLM & Relevant Profile Statements

SHARE
Technology · Connections · Results

SYSPLEX DISTRIBUTION - & Some Thoughts:

- SYSPLEX Distributor's **key benefit**, seems to be not so much in the overhead it incurs but the balance of CPU achieved via the address spaces / **applications it distributes** workload to.
- **Higher availability** functions from the client's point of view. **- One Application Face to Clients**.
- We've seen **CPU balance off within two CEC's** - two Sites, as more applications have jumped on **the data sharing band wagon** – (more on those applications later).
- **Queuing** at some of these address spaces / applications, is also **less as the workload is spread.**

How It Flows:

- SYSPLEX Distributor's balancing is such that the **Distributing stack**, from where the DDVIPA is being advertised, **always receives inbound traffic**
- If SD engages a target participant stack, then **outbound traffic** will flow from that stack - and one will see **sessions established on the participant stack**.
- When the Distributing stack receives an inbound packet, it datagram forwards it to the target stack.
- **Cross CEC** traffic will use the XCF unless **VIPAROUTE** is enabled to use another path. (more later)
- **In a way**, that kind of forwarding can be perceived as **asymmetric routing** – (A bit more on that later too)
- VIPADEFINE / VIPABACKUP are part of the **definitions required** for SD.

TCPIP SYSPLEX DISTRIBUTION – BASEWLM / SERVERWLM & Relevant Profile Statements

SHARE
Technology · Connections · Results

BASEWLM / SEVERWLM:

- **BASEWLM** is recommended for applications that **serve as Gateways.** It's a view from the LPAR Performance index
- The reason for this is that Gateways may **not know** the state of the applications they service. Some examples of such address spaces / applications are:
  - TN3270 TELNET Server would be an example of this
  - DCAS that interfaces with Legacy CICS
  - FTP Daemon address space spawns via a FORK(0) for an FTP server the new session
- **SERVERWLM** represents the health of the application from WLM's point of view.
- There are **two** key elements:
  - The Application's importance **relative to its service class**
  - How well the application is **performing** compared to the **WLM goals** for that application
  - Additionally, WLM provides an interface to allow the application to provide:
    - **Abnormal** transaction completion rate
    - **Application health**, a value in the range 0-100% (100% being optimal)

TCPIP SYSPLEX DISTRIBUTION – BASEWLM / SERVERWLM & Relevant Profile Statements

SHARE
Technology · Connections · Results

Relevant Profile statements – (Note: full explanations not in the scope of this presentation)

See z/OS Comm server for full information – Chapter 8

- **IPCONFIG:**
  - SOURCEVIPA
  - SYSPLEXROUTING
  - DATAGRAMFWD
  - DYNAMICXCF
  - TCPSTACKSOURCEVIPA 192.168.XX.YY → The Stacks Identity / Primary interface.
- **GLOBALCONFIG:**
  - SYSPLEXMONITOR
  - DELAYJOIN → **Wait for OMPROUTE**
- **SRCIP:**
  - JOBNAME ABC1234 **192.168.AA.yy** → For Application Centricity
  - JOBNAME ***** **192.168.XX.YY** → The Stacks Identity / Primary interface.

VIPADYNAMIC:

- **VIPARANGE** → **DISRUPTIVE 255.255.255.255**
- VIPADISTRIBUTE
- **VIPAROUTE** → **Use for XCF Packet Relief**
- VIPADEFINE /VIPABACKUP
- **SYSPLEXPORTS** → **Used for outbound ephemeral ports – EZBEPORT XCF structure**

TCPIP SYSPLEX DISTRIBUTION – VIPAROUTE & XCF IPL Relief

- **SYSPLEX Distributor** uses internal logic to **optimize data forwarding** when engaging a Target / Participant Stack:
  - **IUTSAMEH** if two stacks under the **same MVS** Image
  - **HIPERSOCKETS** when enabled **intra – CEC**.
- Initially, **XCF** used for data forwarding when engaging a Target Stack – Inter-CEC.
- XCF **not most optimal** route for SD forwarding traffic.
- More applications are **joining the XCF** band wagon – DB2, MQ, CICS
- A Better way was needed – **VIPAROUTE** is introduced –z/OS 1.7:
  - Use Optimal paths with **low Cost / metrics** to forward DDVIPA packets:
    - **MPC connections**
    - **High SPEED** Gigabit Ethernet segments using **OSA-Express**
- ***How it works:***
  - Generic routing encapsulation (GRE) using an **Advertised IP address from the target** stack as specified in the VIPAROUTE statement – A **static VIPA is recommended** – we used our APPN/EE static VIPA.
  - In our case the **OSA's** to get across CEC have the lowest cost via OMPROUTE / OSPF
    -

TCPIP SYSPLEX DISTRIBUTION – VIPAROUTE & XCF IPL Relief

SHARE
Technology · Connections · Results

VIPAROUTE Examples & Confirmation :

- From the profile – It's a **subset of the VIPADISTRIBUTE** statement
- Only needs to be specified once – (**Relative to target Stack**)
- The Parameters are The XCF IP address of the participant stack & an advertised address from that stack – **Static VIPA is recommended** (We use static EE VIPA)

- ***Sample***:

  VIPADISTRIBUTE 192.168.ZZ.JJ PORT LLLLL
  DESTIP 192.168.EE.CC 192.168.EE.DD
  VIPAROUTE 192.168.YY.XX 192.168.AA.BB (Target XCF IP Address / STATC VIPA)
  VIPADISTRIBUTE 192.168.ZZ.JJ PORT PPPP

- ***Sample command***: NETSTAT VIPADYN:

  VIPAROUTE is enabled
  XCF Address TargetIp RtStatus
  192.168.EE.EE 192.168.II.SS Active

- **NOTE:** ALSO RAN A PACKET TRACE USING MY PC IP ADDRESS CONFIRMING OSAS NOT XCF USED

TCPIP SYSPLEX DISTRIBUTION – VIPAROUTE & XCF IPL Relief

SHARE
Technology · Connections · Results

How it works - Continued:

- The dynamic XCF address is still required be **configured**.
- SYSPLEX Distributor continues to use a dynamic **XCF** address for **signaling** to TCP/IP stacks in the SYSPLEX
- Also, there are **several functions** that continue to depend on dynamic XCF connectivity for intra-SYSPLEX communications:
  - XCF Structures such as:
    - **EZBEPORT –** Ephemeral port coordination
    - **EZBDVIPA –** SYSPLEX Wide Security **- SWSA**
    - **Policy Agent** services
- GRE De-capsulation takes place at the **target / participant** stack
- GRE adds **28 bytes** in its encapsulation – **fragmentation** considerations warranted:

- ***Fragmentation*** *considerations*:
  - Enable MTU **Path Discovery**
  - Ensure Inter-CEC paths **favor OSA's**
  - Gigabyte Ethernet accommodates **Jumbo** packets (8992)
  - **AGAIN PATH MTU** Discovery is key to **avoid fragmentation both** towards **Clients** while maintaining Jumbo packet capacity between **Inter-CEC LPARS**

TCPIP SYSPLEX Distribution And Broken Symmetry – Asymetric routing & F/W Statefulness

- ***It's Hard Times For Everyone:***
  - The SYSPLEX **Distributor** and Related DDVIPA Applications worked **in Harmony** for about **two** years when an unexpected turn of events Hit.
  - While all was **well** In **Main Street Mainframe Boulevard**, A **topological** Event **further down** stream caused SYSPLEX Distribution to **ceased working**.
  - **ENTERPRISE WIDE!**
- ***How it Happened:***
  - One typical **Monday morning**, users started to complain about slower response times, DB2 applications were **not** being distributed and more.
  - We Began investigating **the health** of the SYSPLEX Distributor

TCPIP SYSPLEX Distribution And Broken Symmetry – Asymetric routing & F/W Statefullness

- ***Its Hard Times For Everyone:***

TCPIP SYSPLEX Distribution And Broken Symmetry – Asymetric Routing & F/W Statefulness

- ***How it Happened - Continued:***
  - Key Indicators from the Distributor's point of view clearly indicated target stacks were **unresponsive**.
  - This was the case on **all** our SYSPLEX environments – Production, QA, Development, SYSTEMS PLEX.
  - **Key** indicators are :
    - **TSR:** Responsive rate between the distributor and the target stack
    - **TCSR:** Connectivity Success rate to the target stack
    - **CER:** Connection establishment rate between sever and client
    - **SEF:** Server accepting new work
  - Following are some command examples that led us to our **shock.**

SYSPLEX Distribution Confirmation

- ***Sample Commands – Health of the SYSPLEX Distributor***
  - Here's a sample of a **HEALTHY** Environment::
  - NETSTAT **VDPT**:
    - Dynamic VIPA Destination Port Table:

Dest IPaddr	DPort	DestXCF Addr	Rdy	TotalConn	WLM	**TSR**	Flg
192.168.BB.11	12345	192.168.AA.YY	001	000**18152**	05	**100**	B > **LPAR-1**
192.168.BB.11	12345	192.168.AA.ZZ	001	000**18180**	02	**100**	B > **LPAR-2**

  - Here's a sample of a **UNHEALTHY** Environment::
  - NETSTAT **VDPT**:
    - Dynamic VIPA Destination Port Table:

Dest IPaddr	DPort	DestXCF Addr	Rdy	TotalConn	WLM	**TSR**	Flg
192.168.BB.11	6789	192.168.AA.YY	001	000**23185**	05	**100**	B > **LPAR-1**
192.168.BB.11	6789	192.168.AA.ZZ	001	000**00000**	02	**000**	B > **LPAR-2**

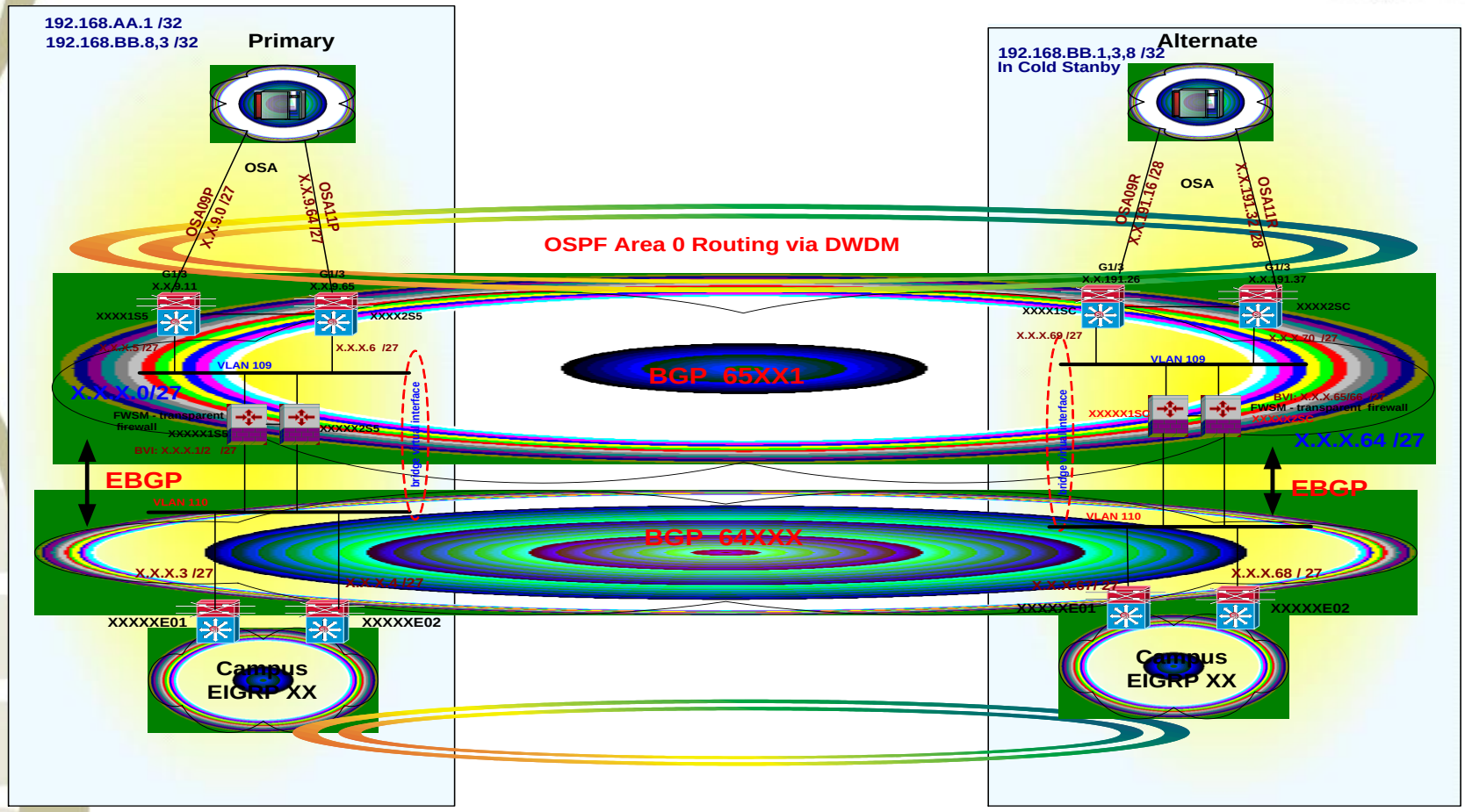
  - **NOTE**: See Appendix for more DVIPA commands:

TCPIP
The Lost of SYSPLEX Distribution – A Crises

SHARE
Technology · Connections · Results

- ***The Cause:***
  - After many traces, dumps and meetings **OH MY…** with other infrastructure teams it turned out that distribution ceased due to the new MPLS core network firewalls and a perception of **asymmetric** routing.
  - There are **two sets of firewalls, per site**. The two are in VRRP mode where one is active and the other in standby. This design's objective is to balance firewall traffic.
  - In the previous infrastructure, there were also four firewalls, one active and 3 as standby, thus in and out packets **traversed the same Firewall**.
  - **Broken symmetry** / Asymmetric routing is perceived due to the SYN packet arrives at the Distributor but the SYN ACK goes down from the participant stack. F/W's see this as **broken statefulness!**
- ***The IMPACT – Applications in SD Mode:***
  - MF Loss of dynamic higher availability, reliability, workload balance and 24/7 up time.
  - Applications impacted are:
    - TN3270
    - CITRIX
    - DB2
    - FTP
    - NDM
    - CICS (IN PROCESS OF MOVING TOWARDS DATA SHARING)
    - MQ (IN PROCESS OF MOVING TOWARDS DATA SHARING)
    - HTTP (IN PROCESS OF MOVING TOWARDS DATA SHARING)
    - WEB SHPERE application servers – Java based (Clustering via)

Campus and Mainframe Design
Routing via BGP



TCPIP
The Lost of SYSPLEX Distribution

- ***The OPTIONS:***
  - Enable our MPLS firewalls to **behave like our former firewalls'** infrastructure.
  - **CSM**, Content Switch Module, CSM. It's bidirectional, capable of inquiries to WLM information for true workload balancing. Firewall balancing could still be achieved if CSM had this capability.
    - *Note that entire DDVIPA architecture would be removed from the MF & onto CSM*
    - *Learning curve plus CISCO & IBM had not heard of **any other organizations** doing what we had hoped for.*
    - *Would cost between **$100 – to - 120K**.*
  - **Policy Based Routing** – a new feature just in time for our crisis, only available in z/OS 1.9 & above – part of communications server and **no extra money**.
- ***The SOLUTION:***
  - ***Policy Based Routing Stimulus Package!***

TCPIP

Policy Based Routing Stimulus Package (z/OS 1.9)

SHARE
Technology · Connections · Results

- ***Policy Based Routing Stimulus Package:***
- ***The Goal:***
  - We needed a way to **preserve symmetry** from the F/W's point of view; **Re-Enable** SYSPEX Distribution, at a **Minimal Cost;** in the most **timely manner;** and maintain **XCF Packet relief.** Policy-based routing provides a **routing option to send traffic** based on **source IP address**. Using policy agent, the traffic criteria and policy-based routing tables are defined to the stack for routing
- ***How it's working for us:***
  - Policy based routing is enabled on the **target stack**.
  - We created **MPC definitions** directly connecting the Distributor & Target stack.
  - We specify that DDVIPA source IP address packets be **forwarded back** to the Distributing Stack and thus use that stacks main routing table to get **back to the client**
  - This **maintains statefulness** from the F/W's point of view.
  - We also specify that if the distributing stack is not available – for whatever reason, then use the **main routing table** on the target stack.
  - The Policy agent injects a static route into TCPIP's routing table. Thus even if the policy agent **address space crashes**, the packets **continue** to be forwarded back to the distributing stack.
  - We're @ **98%** of what we have because the concept of **VIPABACKUP is moot** under this configuration due to **F/W statefulness** being broken upon a VIPA take back.

TCPIP

Policy Based Routing Stimulus Package
SD's Back to Work – 98%
(Available z/OS 1.9)

- ***Policy Based Routing Stimulus Package:***

  Routing Configuration: Environmental file; Configuration File ; Configuration routing table

Environmental file:

```
PAGENT_CONFIG_FILE=/etc/pagent.conf
PAGENT_LOG_FILE=/tmp/pagent.log
TZ=EST5EDT,M3.2.0,M11.1.0
```

Configuration File:

```
# TcpImage Statement to flush existing policies
  TcpImage TCPIP FLUSH PURGE 60
# LogLevel Statement
   Loglevel 511
# RoutingConfig Statements
  RoutingConfig /etc/pagent_Routing.conf
```

Config Routing Table: There are **three** parts to it:

Routing Rule
RoutingAction
RouteTable

NOTE; we group in one section the DDVIPA addresses or else there would be multiple rules and actions. It's called **IpSourceAddrGroupRef** We have:

One Rule to hold them;
One Action to reference them
One Routing table to establish them…
And in this state, BIND them.

TCPIP

Policy Based Routing Stimulus Package And SD's Back to Work – 98% (Available/OS 1.9)

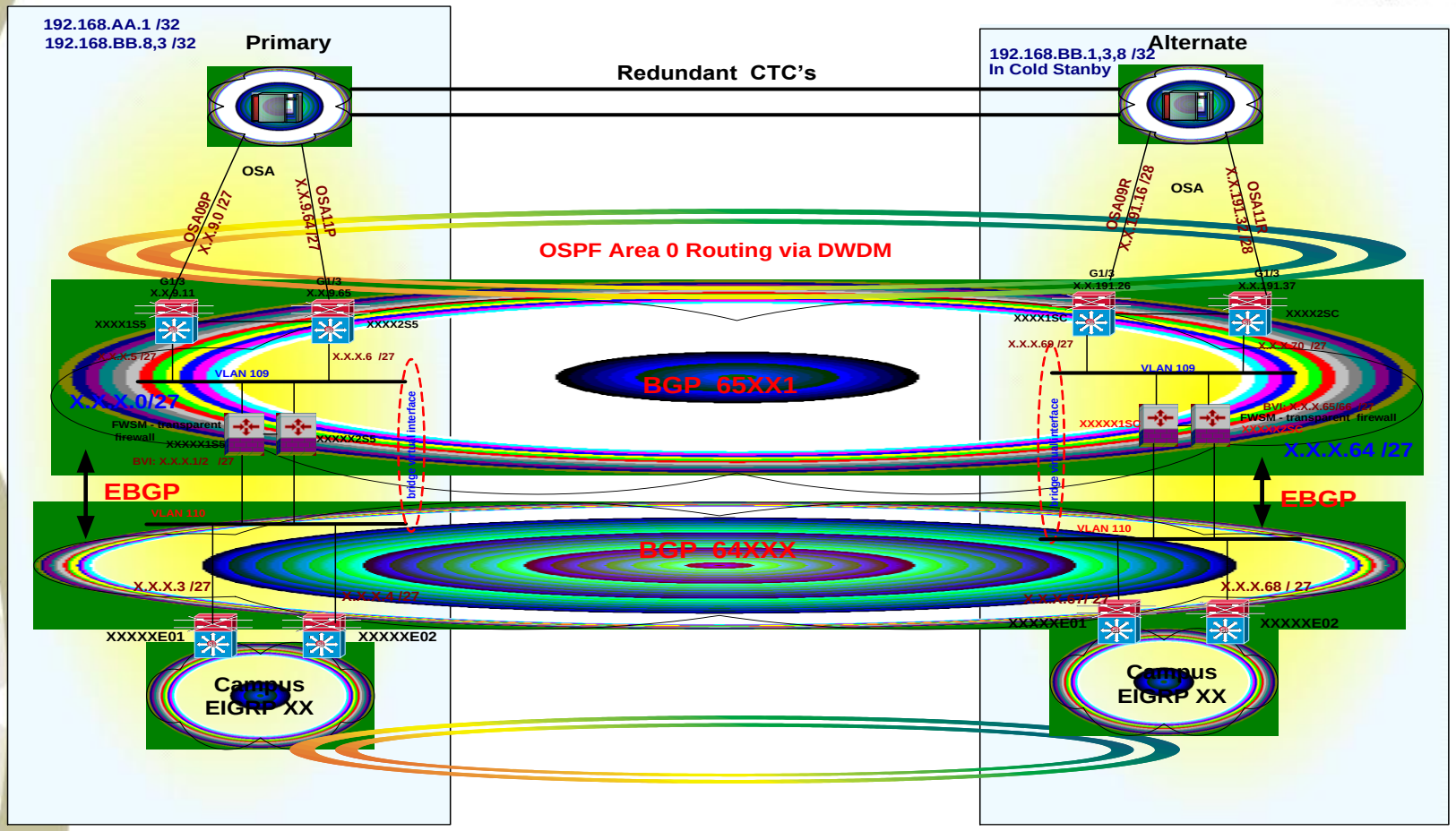
SHARE
Technology · Connections · Results

- **Sample of Configuration routing table grouping DDVIPA's**

```
RoutingRule Rule1
{
  IpSourceAddrGroupRef  LPAR2-addrGroup
  RoutingActionRef      LPAR2-SDPBR-Routing
  Priority              100
}
# SD-PBR-Routing source ip address groups.
# - Group DDVIPAs for policy based routing.
#
IpAddrGroup LPAR2-addrGroup
{
  IpAddr
  {
    Addr 192.168.AA.ZZ
  }
  IpAddr
  {
    Addr 192.168.AA.CC
  }
RoutingAction  LPAR2-SDPBR-Routing
{
  RouteTableRef       L2RteTbl
  UseMainRouteTable   Yes
}

#
RouteTable L2RteTbl
{
Route 192.168.XX.YY    =              MPC123EL  MTU 1500
Route Default          192.168.XX.YY MPC123EL  MTU 1500
}
```

Campus and Mainframe Design
InterCEC Routing via CTC's



DTCC	Thursday, February 26, 2009
	Author: NSP & IDM
IDM	Campus Migration

SYSPLEX Distribution Confirmation & Simulation Methods

- ***SYSPLEX Simulation Methods with Policy Based Routing:***

- ***In this section there are simulations where*** **– L1 is Distributor / L2 is Target stack**:

  - Confirm / Level Set SYSPLEX Distribution mode as normal
  - Simulate SYSPLEX Distributor engages target stack for workload balancing
  - Simulate SYSPLEX failure on L1 - Distribution & advertisement are now from L2.
  - Re -normalize SYSPLEX Distribution mode
  - Simulate a PAGENT & OMPROUTE crash

  * SYSPLEX CONFIRMATION – NETSTAT commands I found most useful

NOTE: SEE APPENDIX FOR DETAILS

IPL RELIEF ORDER & DEEPER LEVELS OF AVAILAILITY

- ***IPL relief Order:***

  - There was an incident where **all LPARS were down** and **LPAR-2 was brought** up before LPAR-1.
  - Connectivity was interrupted **until LPAR-1** Came back up.
  - **IPL Relief** Order was introduced by specifying DYNAMIC VIPA BLOCK along with all **required VIPA Distribute** statements on LPAR-2.
  - The order of statements are important: (See appendix for further details)

    - VIPADEFINE MOVEABLE IMMEDIATE
    - VIPABACKUP 1 MOVEABLE IMMEDIATE
    - VIPARANGE DEFINE MOVE DISRUPT
    - VIPADISTRIBUTE 192.168.37.11 PORT 50005 DESTIP 192.168.12.33 192.168.12.34 > **NOTE: NO VIPAROUTE ON TARGET STACK**

- ***Deeper Levels Of Availability:***

  - **VTAM & TCPIP** – ARM'D – (AUTMATIC RECOVERY MANAGER POLICIES – (Handled By MVS Group)
  - **OMPROUTE** - **ARM Wrapped** – In Production OMPROUTE took an OC1 and was restarted by ARM saving us much down time. (Of course a dump was produced – it was related to the LE environment.)

  - **See APPENDIX for more details & JCL**

INNOVATION: THE INDIGO CHILD

- ***WHY "STATIC TO - DYNAMIC:***

Our Enterprise is **under the Umbrella** of IP Application Centricity, distancing new Applications away **from the Stack's Identify**.

Thus data sharing applications & SYSPLEX Distribution Benefit our clients via:

- Doubling availability
- Reliability
- Scalability
- Resource Balance
- Optimal transaction throughput
- Cross CEC CPU Life Extension

HOWEVER

85 % Of our Internal & External Clients Still Target the Stack's Identity

Like an In & Out revolving mainframe Door

Innovation: the Indigo Child
SYSPLEX Distribution

Innovation: the Indigo Child SYSPLEX Distribution

- ***Our Goal:***

- Our entire efforts of **evolving from "Static - To - Dynamic**", as it applies to the Stack's Identity / Primary Interface, has been and continues to be **the highest exploitation the SYSPLEX Distributor**. This pertains to enabling DDVIPA capabilities relative to the primary interface of the stack.
- This is specifically aimed to **legacy applications** that are constrained to use it exclusively**. About 85%** of our internal and external clients **BIND to the stacks Identity**.
- **Posture** the Stack's Identity to be DDVIPA capable, for data sharing capable applications, provides a re-doubling of availability while further balancing the IP workload.
- Provide SYSPLEX **wide integrity relative to rolling IPL(s**), Stack and/or LPAR Failure –
- This became relevant as about **75% of our applications are data sharing capable** but the remainder are not, thus **pegged** to the given LPAR. Consideration had to taken to maintain their integrity in light of VIPA Take Over.
- Maintain Network **connectivity tests available** for Network Engineering group as TELNET, TRACERTE PINGS, NSLOOKUP / DNS resolution **– UDP , are impacted.**
- **All of the above while maintaining Transparency to our Internal & external Clients!**

Innovation: the Indigo Child SYSPLEX Distribution

- ***Why not NAT to a different IP address versus converting the Stack's Identity / Primary Interface, from a static VIPA to A Dynamic VIPA.***
- Converting the stack's identity from static to dynamic versus NATing, from our experiences, we've had issues with NATing, as other groups are involved.
- Also The Firewalls have had issues being in synch with our various sites.
- Inclusive to this, some bubble/extranet networks,, will do double/triple NATing. NATing here will add yet another level of complication in the environment when troubleshooting.
- issues with this type of double NATing, not to mention getting all the parties in synch when trouble shooting. Inclusively, the application Can the application folks support this in a seamless manner? If so, MQ, NDM, Websphere, FTP, etc, would each have a new IP(in line w/application centricity); So we're talking several manual NAT's(the target port would dictate the IP address)
- Our business that supports the financial industries, both national and global markets - our calculations indicate that in excess of **337G of TCPIP bytes target our primary interfaces, with a transactional value approaching US$2.5 Trillion in value flow through our veins on a typical day;** we deemed it too risky to expose our production environments to multi-tier diagnosis and support, should anything affect our many thousands of internal and external clients from connecting to the stack's identity, a static VIPA.
- As such, the innovative solution to provide the remainder of our legacy clients and applications - approximately 85% our of connections in production, the dynamics of SYSPLEX Distribution was achieved by converting the stack's identity to a Dynamic VIPA. Once there, it was a short step to convert it to Distributed Dynamic VIPA for data sharing capable applications.
- This maintained while providing SYSPLEX integrity for applications that were singly pegged to a given LPAR and also using the stacks identity.
- The key to this was to make the transition **100% transparent**, thus minimizing exposure or leakage of responsibility / support, to other entities external to the mainframe.
- We conducted what we call ' Dress Rehearsals with representatives of all mainframe disciplines and key external clients. It was a kin to when we conduct **Disaster Recovery** tests. Many proof of concepts to senior management were performed, but all noted the gains of such an endeavor.

Innovation: the Indigo Child SYSPLEX Distribution

PHASE ONE: Stack's Identity / Primary Interface Becomes A Dynamic VIPA – 95% There

Parameters and Listener Applications:

A) TCPIP Profile:

1) **Remove LPAR Static VIPA Statement from Home section**

2) **Remove Device / Link statements for static VIPAS**

3) **Remove Primary interface statement**

4) **Add TCPSTACKSOURCEVIPA statement to Profile (Part of IPCONFIG section) ***

5) **Add SRCIP statement for address spaces that BIND first and Then Connect – MQ was susceptible to this****

6) **Add in SRCIP block at the end JOBNAME * 192.168.XX.Y (The stacks Identity) *****

7) **Add VIPADEFINE statement for the stack IP address to Profile**

8) **Create OBEYPRXX member with HEX ADDRESS of Prime Interface – ******

   a) Example: PRIMARYINTERFACE VIPLC0A8YYXX

   b) VIPL – required as prefix to the to HEX representation of IP address of stack's identity.

   c) 192.168. YY.XX in HEX: C0A8YYXX

9) **Add DELAYSTART in AUTOLOG section for all tasks started except OMPROUTE.**

10) **DO NOT Add VIPABACKUP Statement for the stack IP address in Partner PLEX Profile – in Phase Two.**

Innovation: the Indigo Child SYSPLEX Distribution

PHASE ONE: Stack's Identity / Primary Interface Becomes A Dynamic VIPA – 95% There

Parameters and Listener Applications (Continued):

B) OMPROUTE CONFIG & MORE:

11) **Remove LPAR Static VIPA Statement from OMPROUTE CONFIG.**

a) We Use **DVIPA Wildcard / Models in OMPROUTE** – "One statement keeps the all":

```
OSPF_Interface
 IP_Address=192.168.AA.*
Name=dummyAA1
Subnet_Mask=255.255.255.0
```

12) **Add automation for Obey of Primary Interface to DVIPA**

a) After - EZD1214I INITIAL DYNAMIC VIPA PROCESSING HAS COMPLETED FOR TCPIP

b) OPENED OBEYFILE FILE 'DSN.TCPIP.PROFILE(OBEYPRXX)'

13) Add **LISTENONADDRESS** statement to SMTP CONFIG

14) OSNMPD PW.SRC file - made **IP addresses conform to class 'B' subnets**

a) This was to avoid having to add **first IP address under the HOME** list per LPAR.

15) **BIND PASCL applications** – UPSTREAM is an example, to primary interface via PORT statement.

Innovation: the Indigo Child SYSPLEX Distribution

- ***PHASE ONE: Stack's Identity / Primary Interface Becomes A Dynamic VIPA – 95% There***

- ***SOME HURDLES TO OVERCOME:***

- ***C) PINGS (ICMP) & TRACRETE (UDP):***

1) PING & TRACERTE uses as SRCIP Address the interface the packet goes out of. Use the SRCIP Parameter to inject the primary interface IP address:

a) TSO PING XXX.YYY.AA.X (SRCIP 192.168.XX.YY
b) NETVIEW PING XXX.YYY.AA.X SRCIP 192.168.XX.YY
OR
PING XXX.YYY.AA.X - Z 192.168.XX.YY

a) TSO TRACRTE XXX.YYY.AA.X (SRCIP 192.168.XX.YY
b) NETVIEW TRACERTE XX.YYY.AA.X SRCIP 192.168.XX.YY
OR
TRACERTE XXX.YYY.AA.X - Z 192.168.XX.YY

Innovation: the Indigo Child SYSPLEX Distribution

- ***PHASE ONE: Stack's Identity / Primary Interface Becomes A Dynamic VIPA – 95% There***

- ***SOME HURDLES TO OVERCOME (continued):***

- ***D) NSLOOKUP / DNS (UDP):***

  1) Our DNS servers are in the CAMPUS – VS – Main Frame.
  We addressed this issue via Firewall UPDATES.

- ***E) FIREWALL UPDATES :***

  1) OSAs - We added the Class 'C' subnets of our OSAS(S) to the Firewalls.

  2) By Doing this we accomplished 3 things:

  a) General Internal PINGS now did not need the SRCIP to be injected into the packet.
  b) General Internal TRACRTE now did not need the SRCIP to be injected into the packet.
  c) NSLOOKUP / DNS work fine.

  3) HOWEVER – FOR EXTERNAL CLIENTS, TRACERTE REQUIRES THE SRCIP INJECTION.

***** *Our Enterprise FLAGSHIP for these activities is 'INSIDE THE STACK' - - It handles SRCIP injection as required.*

Innovation: the Indigo Child SYSPLEX Distribution

- ***PHASE ONE: Stack's Identity / Primary Interface Becomes A Dynamic VIPA – 95% There***

- ***SOME HURDLES TO OVERCOME (Continued):***

- ***F) Connectivity Tester (CONNTEST) – VS - TELNET***

  As previously discussed we had to develop **CONNTEST,** as we evolved towards IP Application Centricity.

  As it turned out, It would also be required for our 'Static – To – Dynamic ' conversions**!!!**

  In Brief:

  - **TELNET** was used by our **NEO** Group to confirm **F/W & Connectivity.**
  - As we continued to evolve into Application IP Centricity, - where an **application listens on a specific** IP address, the ability to inject a source IP address assigned to an application is **mandatory**.
  - IP Application centricity & 'Static – To – Dynamic – came to a grinding halt **TELNET** uses **RAW sockets** logic and is not a true TCP protocol. **SRCIP** would be **interface** it went out of.

Innovation: the Indigo Child SYSPLEX Distribution

- ***PHASE TWO: Stack's Identity / Primary Interface becomes a Distributed Dynamic VIPA –***

- ***Required Parameters:***

  - After all had been set and done, and many hurdles overcome, to achieve the final step seemed an easy affair…

    IN GENERAL , JUST 2 LINES OF CODE AS NOTED BELOW On Distributor Stack & 1 line on Target:

  * Under the VIPADYNAMIC BLOCK Simply add the IP address of the Stack's Identity along with the PORT that represent data sharing capable applications Such as PORT 23:

    VIPADISTRIBUTE SYSPLEXPORTS 192.168.XXY PORT 23
    DESTIP 192.168.AA.BB 192.168.AA.CC

    VIPABACKUP On Target Stack

BUT NOOOOOO!

Innovation: the Indigo Child SYSPLEX Distribution

- ***PHASE TWO: Stack's Identity / Primary Interface becomes a Distributed Dynamic VIPA –***
- ***The Special Case Of DB2 & MQ:***

Background:

Some **DB2 & MQ** was still **anchored** to the Stack's Identity.

Rolling IPL(s) would cause the Stack's Identity to temporarily **be taken and advertised** from the participating stack.

Down stream servers would **constantly try to establish** a session with the stacks Identity & DB2 – That had now moved to the participating stack , **if the connection was bro**ken at any time.

As Previously mentioned, there are **two sets of firewalls, per site**. The two are in VRRP mode where one is active and the other in standby. This design's objective is to balance firewall traffic.

- ***The Problem:***

  - Down Stream servers would **establish a session with DB2 on LPAR-2**, Target stack..
  - LPAR-1, the distributing Stack, comes back up; **takes back its Identity.**
  - **State Fullness** would be broken.
  - But the **sessions stayed established** & the MQ region needed by DB2 / DERIV server was always on LPAR-1 and **not capable of data sharing**.

Innovation: The Indigo Child SYSPLEX Distribution

PHASE TWO: Stack's Identity / Primary Interface becomes a Distributed Dynamic VIPA –

The Special Case Of DB2 & MQ (Continued):

The SOLUTION:

* The SYSPLEX **Integrity Checker** – Newly Developed Code. It Confirmed that **no lingering sessions** remain on The Participating Stack or the Target stack **that did not belong there**.

* It kicked in when the participating cross CEC **stack(s) had joined the SYSPELX Group.**

* Performed via NETVIEW . **MORE ON THIS LATER.**

* **NEW DB2 IP Application Centric Infrastructure.**

* **Newly Developed Code to Track the Migration - PER PLEX. (VIA SMF 119 & NSLOOKUP)**

Innovation: The Indigo Child SYSPLEX Distribution

- ***NEW DB2 IP Application Centric Infrastructure:***

Below is a Sample:

TPLX	DVIPA	DDVIPA	
DNSDB2DBADS		**192.168.XX.YY1**	50040
DNSDB2DB1A	192.168.XX.YY2		
DNSDB2DB2A	192.168.XX.YY3		
DNSDB2DB3A	192.168.XX.YY4		
DNSDB2DBBDS		**192.168.XX.YY5**	50050
DNSTDB2DB1B	192.168.XX.YY6		
DNSDB2DB2B	192.168.XX.YY7		
DNSDB2DB3B	192.168.XX.YY8		
DNSDB2DBDDS		**192.168.XX.YY9**	50020
DNSDB2DB1D	192.168.XX.YYA		
DNSDB2DB2D	192.168.XX.YYB		
DNSDB2DB3D	192.168.XX.YYC		

Innovation: The Indigo Child SYSPLEX Distribution

- ***PHASE TWO: Stack's Identity / Primary Interface becomes a Distributed Dynamic VIPA –***

SYSPLEX Integrity Checker Specifications:

- The SYSPLEX Integrity Utility. This utility will **know the state** of the PLEX and perform cleanup of any connections that should not be on a given stack.

Example: The Primary interface of LPAR-1 should never be in established states on LPAR-2, under normalized conditions**.**

Innovation: The Indigo Child SYSPLEX Distribution

SHARE
Technology · Connections · Results

SYSPLEX Integrity Checker Specifications (Continued):

1) Written in REXX, it will be executed 3/ 5 mins after NETVIEW initializes - giving time for other applications, such as DB2 to initialize.

2) **Three inputs:**

a) Output from NETSTAT CONN for established states under the local sockets section

b) **Input dataset** relative to each PLEX participant, indicating **which address(es) that should not be** in established local states:

I.E. 192.168.XX.1, LPAR- 1, should never be in an established state - Local Socket, on 192.168.XX.2, LPAR-2, under normal operations.

c) Participant's SYSID - needed to know when PLEX is normalized.

3) How often should the code execute?

a) It will kick **off 3 / 5 minutes after NETVIEW is initiated** - Automatic timer in NETVIEW;
b) It **must know if the PLEX is normalized**. That would be when all relevant stacks have joined the SYSPLEX Group. (VIA - **D TCPIP,,SYSPLEX,VIPA**DYN)

b.1) Has Participant stack has joined and is active

b.2) If "**YES", execute cleanup** routine and terminate

b3.) **If not, go into a WAIT state** for 1 min and check again. Continue this LOOP until participant stack is ACTIVE → then execute section b.2.

c) When a match is found in its cleanup routine, it will inform us - possibly via email.

(See Appendix for more details)

Innovation: The Indigo Child SYSPLEX Distribution

SHARE
Technology · Connections · Results

The Stack's Identity Becomes a Distributed Dynamic VIPA!!!

100% There!!!

Sample Of Exploitation – 'A Story':

As we've continued to evolve and exploit our SYSPLEX Distribution environment, we became aware that **over 2500 PC's** were going to be upgraded to have a new Extra Attachmate that would've used **a new SYSPLEX Distributed IP address and Port**. About 10% of the campus had complied - **90 %** were a bit slow to **change.**

The intent was to **double their availability & balance our resources**, for campus users into the mainframe.

All that was negated and the same **goals achieved with 4 lines of code** in TCPIP, - 2 per SYSPLEX stack. This benefited the Desk Top engineering Group, been involved in pushing the new Attachmate plus supporting the campus with the new modifications.

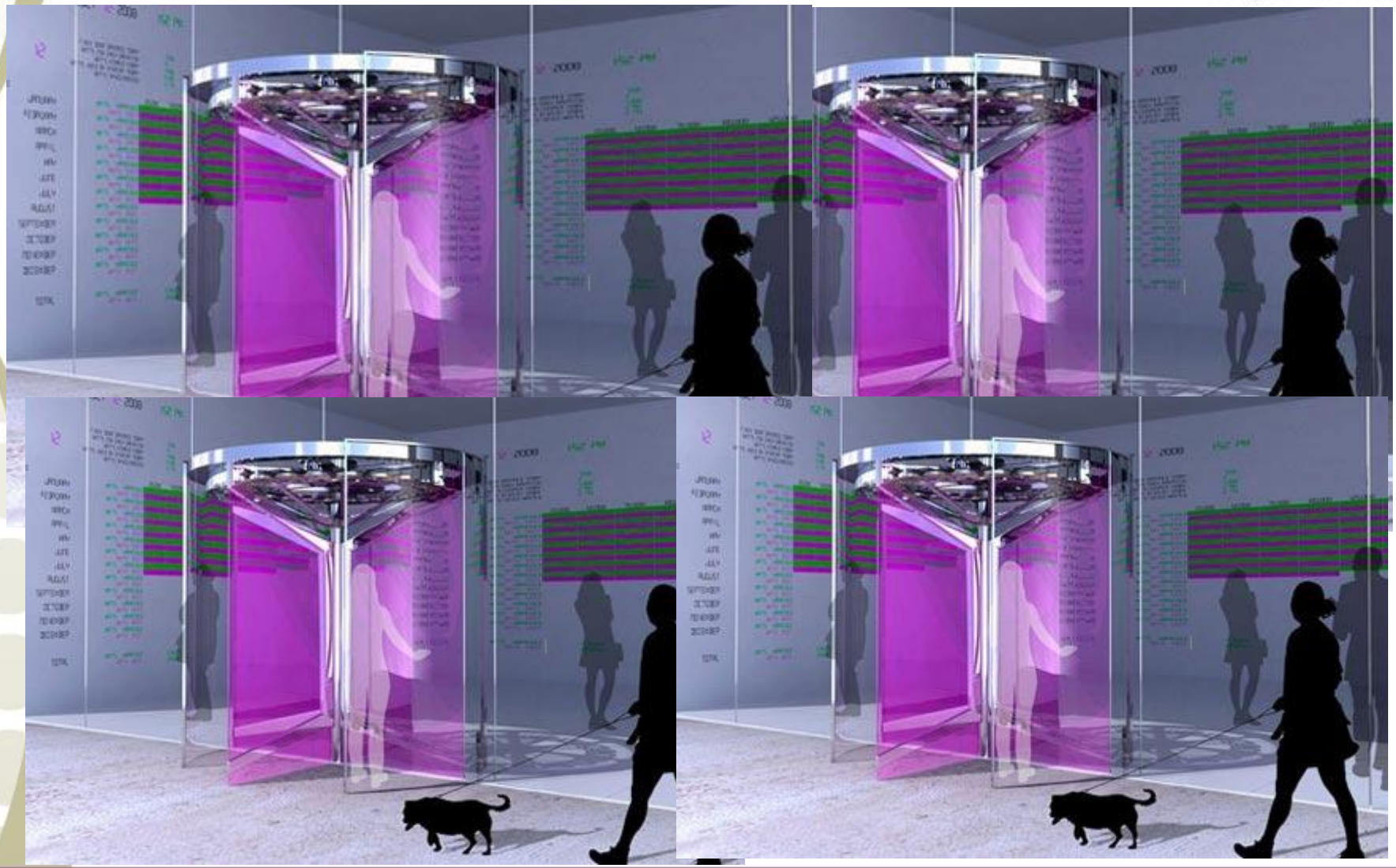
****** Over 1500 TN3270 users were now balanced across the PLEX -VS- on One LPAR.**

And there it is:

Now That was a MAJOR Voyage & FUN!!!!

And There Is Now More Than One Revolving Door…

Innovation: the Indigo Child SYSPLEX Distribution



CONCLUSION

MAY WE ALL CONTINUE TO

RECEIVE TO SHARE

&

SHARE TO RECEIVE

Appendices

- SUBJECTS:
  - **DNS naming conventions**
  - **CONNTEST Socket Program**
  - **DVIPA & Policy Based Routing Commands**
  - **IPL Relief order – Sample statements**
  - **OMPROUTE ARM'D JCL & ARM POLICY Commands.**
  - **SYSPLEX SIMULATION METHODS & POLICY BASED ROUTING**
  - **Useful NETSTAT DVIPA COMMANDS**
  - **POLICY AGENT JCL**
  - **VTAM GENERICS & SYSPLEX Distribution – Their Joining sample**

Appendices

- ***DNS naming conventions – I.E – MQ, CICS, DB2***
- Regarding DNS naming conventions, from the Network perspective, current and future directions, there are two sides to the DNS coin.
  - Facilitate connectivity for our clients enabling access without changes from their point of view. IP addresses can change with transparency to the end user
  - From the Network perspective, the new DNS naming convention will facilitate better Diagnostics as we will be able to tell at a glance where the application normally resides, the type of application and even the Job Name of the given address space
- The DNS naming conventions also will reflect if the address is a distributed address, SYSPLEX Distributor, or if it's a Dynamic VIPA that follows the given address space.
- **General: Using T-PLEX as an example**:

Bytes:
1-2 SM --- Server Mainframe

SM

For NON Distributed - DVIPA follows Region:
3- 4 TT/ PP/ ZZ/ UU/ QQ T test, P production, Z for dev, U user testing, Q internal QA
SMTT
For Distributed - DDVIPA address - SYSPLEX Distributor - one IP address serves all participant application. The uniqueness is a function of the ports....thus a unique socket, the IP address and port, the address space listens on.
3 - 5 TDS for distributed

SMTDS

The Applications:
For Distributed:
6 - 7 and/or 8

SMTDSMQ

SMTDSCIC
SMTDSDB2

For Non - Distributed - DVIPA follows the Region:
5 - 6 The given application type - MQ / CI - (FOR CICS) / DB2

Appendices

- ***DNS naming conventions – I.E – MQ, CICS, DB2***

SMTTMQ

SMTTCI

SMTTDB

The individual address space:
7 - 8

MQ - The 3rd and 4th bytes of the Job Name reflect the individual address space. -They would be the 7th and 8th bytes of the DNS name.

CICS - The 7th and 8th bytes of the Job Name reflect the individual address space . - They would be the 7th and 8th bytes of the DNS name.

DB2 - The 3rd and 4th bytes of the Job Name reflect the individual address space. -They would be the 7th and 8th bytes of the DNS name

EXAMPLE: Below is an Example of the DNS names and its relation to the address space.

Job Name;	**MUE1CHIN**	**DNS Name:**
	SMTTMQE1	
	CICPCTG1	**DNS Name:**
	SMTTCIG1	
	DB2TDIST	**DNS Name:**
	SMTTDB2T	

Anomalies that fall out of the norm, would be reviewed individually.

Appendices

- ***CONNTEST Connectivity Tester***
- *An Diagnostic socket program, (REXX), developed for IP Application Centricity where each application is assigned a unique IP address. It has the capability to inject a source IP. Previously, TELNET was used for connectivity tests, passing F/W's etc.*

- **NOTE: This is not the exact code in use but its enough to get started.**
  If needed please email me with SHARE in the subject & CONNTEST.

```
/* rexx new code                                    */
/* 05/06/08 this pgm tests connectivity to any port    */
/* from either the primary ip address (default) or     */
/* specifying a source ip address to allow for dvipa   */
/* checking.                                        */
/*******************************************************/
VAR = SOCKET('INITIALIZE','WXYZ')
/* PARSE arg dst_ADDR PORT   */
PARSE arg dstAddr PORT srcAddr
if dstAddr = '' | port = '' then DO
SAY 'INVALID SYNTAX. Correct CMD is CONNTEST <dstAddr> <PORT>'
                    exit
                    end
if srcAddr = '' then do
                x = Socket('getHostId')
                srcAddr = word(x,2)
          say 'NO SRC ADDRESS PASSED - USING HOSTID ' srcaddr
            end
VAR = SOCKET('SOCKET')
```

Appendices

- ***CONNTEST Connectivity Tester (continued)***

```
  SOCKETID = WORD(VAR,2)
VAR = SOCKET('bind',SOCKETID,'AF_INET 0' srcAddr)
say 'SRC ADDRESS AFTER THE BIND ' SRCADDR
SAY 'CONNECTING TO 'dstAddr 'ON PORT' PORT 'SRC ADDRESS' srcAddr
CONNAME='AF_INET '||PORT||' '||dstAddr
VAR = SOCKET('CONNECT',SOCKETID,CONNAME)
SAY 'RESULT OF CONNECTING TO 'dstAddr 'ON PORT' PORT
if var = 0 then
           say 'connection successful'
else if 2019 = word(var,1)
then say 'UNKNOWN HOST' dstaddr
else if 60   = word(var,1)
then say 'Foreign host did not respond within OPEN timeout' dstaddr
else if 61   = word(var,1)
then say 'Connection refused' dstaddr
else say var
EXIT
```

SYSPLEX Distribution Confirmation & Simulation Methods

- ***SYSPLEX Simulation Methods with Policy Base Routing***
- ***VIPAROUTE on Distributor (L1) & Policy Based Routing On Target Stack (L2)***
- ***In our tests a dedicated DDVIPA & Port was used - TELNET from campus to MF***

- Normal SYSPLEX Distribution mode:
  - DDVIPA 192.168.XX.YY advertised from L1 –
    - Confirm via: D TCPIP,,OMPR,OSPF,LIST,ALL – Towards Bottom of Display:
      ADVERTISED VIPA ROUTES
      192.168.XX.YY /255.255.255.255
  - TN3270 Address space listening on PORT 2320 – L1 / L2
  - Telnet to from campus to 192.168.XX.YY 2320 → Confirm on L1 – Better WLM PI
- Simulate SYSPLEX Distributor to engage L2 for workload balancing:
  - TN3270 Address space listening on PORT 2320 – L2 only via a Quiesce of TN3270 port 2320 on L1:
    - V TCPIP,,SYSPLEX,QUIESCE,PORT=2320 → issue from L1
    - Telnet from campus to 192.168.XX.YY 2320 → Confirm via NETSTA CONN on L2
- Simulate SYSPLEX failure on L1 - Distribution & advertisement are now from L2:
  - DDVIPA not Advertised from L2:
    - V TCPIP,,SYSPLEX,DEACTIVATE,DVIPA=192.168.XX.YY → Confirm OPR CMD on L1.
    - Policy Based routing enabled on L2.
    - Stop the MPC connections via a Stop Device command on L1.
    - Telnet to from campus to 192.168.XX.YY 2320 → Confirm via NETSTAT CONN on L2
      - D TCPIP,,ROUTE,PR=ALL → Notice route table & Reference count – 1 (Me)
    - Kill Policy Agent on L2 – Session is still active.
    - Break TELNET session and reestablish session → **Session re-established!!!!!!**
- Renormalize The environment:
  - V TCPIP,,SYSPLEX,REACTIVATE,DVIPA=192.168.XX.YY → From L1 – re-advertise DDVIPA
  - V TCPIP,,SYSPLEX,RESUME,PORT=2320 → From L1 – R-activate the TELNET Port

POLICY AGENT JCL

- ***Sample JCL & Routing Configuration: - (See Appendix for details)***

```
//PAGENT   PROC
//*
//PAGENT   EXEC PGM=PAGENT,REGION=0K,TIME=NOLIMIT,
//       PARM='POSIX(ON) ALL31(ON) ENVAR("_CEE_ENVFILE=DD:STDENV")/-
   D8
//*
//STDENV   DD PATH='/etc/pagent.env',PATHOPTS=(ORDONLY)
//*
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

SYSPLEX Integrity Checker

- ***SYSPLEX Integrity Checker – FROM NETVIEW Written in REXX:***

```
/*******************************************************/
/* rexx new code: RW / SP - SYSPLEX INTEGRITY CHECKER  */
/* 09/29/09 THIS CODE FIRST CONFIRMS THE STATE OF THE  */
/*          PLEX & EXECUTES WHEN ALL PARTICIPANTS      */
/*          STACKS HAVE JOINED THE SYSPLEX GROUP.      */
/* 09/24/09 this pgm confirms the proper vipa addresses*/
/*          are active on the correct lpar.  If by     */
/*          chance an ip addr is not supposed to be    */
/*          there, it will be dropped.                 */
/*******************************************************/
/* trace r */
init:
say 'Initializing Routine Active'
found = 'none'
count = 3
IF MVSVAR('SYSNAME') = 'LPAR-1' THEN do
   'PIPE QSAM (DSN) 'NETZ.LPAR-1.IPADDRS.SYSPLEX'',
                  '| STEM ipaddr. '
                  stack = 'LPAR-2'
```

SYSPLEX Integrity Checker

- ***SYSPLEX Integrity Checker – FROM NETVIEW Written in REXX:***

```
say 'Participant Stack is ='stack
                /*   say 'found should be NONE ='found */
end
 /*
IF MVSVAR('SYSNAME') = 'LPAR-2' THEN do
   'PIPE QSAM (DSN) 'NETT.LPAR-2.IPADDRS.SYSPLEX'',
    '| STEM ipaddr. '
    stack = 'LPAR-1'
                end
  */
suprvsr:
found = 'none'
say 'In Supervisor Routine'
say 'Number Of Times Left In - Simulating Wait States = ' count
  /* say 'found should be NONE ='found */
if count = 0 then do
   say 'Finished Simuliting Wait States - Count = ' count
   say 'Exiting!!!!!!'
   exit
end
```

SYSPLEX Integrity Checker

- ***SYSPLEX Integrity Checker – FROM NETVIEW Written in REXX:***

```
say 'Calling the Collector'
call collect
/*******************************************************/
/*******************************************************/
/*******************************************************/
call checker
  If found = 'done' then do
  say 'We FOUND IT!!!'
  say 'Calling Cleaner'
  call cleaner
  end
  else do
  /*  say 'Waiting For 1 Second & Collect Again' */
  count = count - 1
  wait 1
  call suprvsr
  end
 /*******************************************************/
 /*******************************************************/
 /*******************************************************/
```

SYSPLEX Integrity Checker

- ***SYSPLEX Integrity Checker – FROM NETVIEW Written in REXX:***

```
collect:
say 'in collection routine'
'pipe mvs d tcpip,,sysplex,vipadyn |corrwait 3 ',
'|collect ',
'| sep ',
'| locate /ACTIVE/',
'| stem msg.  '
say 'record number is ='msg.0
return
/*************************************************************/
/*************************************************************/
/*************************************************************/
checker:
say 'in checker routine'
do ii = 1 to msg.0
   parse var msg.ii w1 w2 stat w4
 /*say 'we are in record ='msg.ii */
   if  stack  = w2  then do
       found = 'done'
       say 'searching found ='found
```

SYSPLEX Integrity Checker

- ***SYSPLEX Integrity Checker – FROM NETVIEW Written in REXX:***

```
 return
    end
end
say 'Did Not Find Participant Stack - Found ='found
say 'Going Back To Supervisor To Wait & Collect'
return
 /*************************************************************/
 /*************************************************************/
 /*************************************************************/
 cleaner:
 SAY 'OK!  FINALLY IN CLEANER!'
 SAY 'OK!  FINALLY IN CLEANER!'
 SAY 'OK!  FINALLY IN CLEANER!'
 'pipe mvs d tcpip,,netstat,conn,max=* |corrwait 10 ',
 '|collect ',
 '| sep ',
 '| locate /Listen/',
 '| stem msg.  '
do ii = 1 to msg.0
   Jobn = substr(msg.ii,1,8)
```

SYSPLEX Integrity Checker

- ***SYSPLEX Integrity Checker – FROM NETVIEW Written in REXX:***

```
conn = substr(msg.ii,10,8)
  addr = substr(msg.ii,19,15)
  if substr(addr,1,1)  = '*'  then addr = msg.ii + 1
  if substr(addr,11,2) = '..' then addr = substr(msg.ii,19,11)
  if substr(addr,12,2) = '..' then addr = substr(msg.ii,19,12)
  if substr(addr,13,2) = '..' then addr = substr(msg.ii,19,13)
  if substr(addr,14,2) = '..' then addr = substr(msg.ii,19,14)
  addr = strip(addr,t,'.')  /* get rid of trailing dots */
  if jobn = 'NETEXIG ' then do
    say 'found negligence ='jobn
     'mvs d tcpip,,n,conn,cli='jobn
    exit
  end
  /* say 'jobn = 'jobn 'conn = 'conn 'addr = 'addr */
   do xx = 1 to ipaddr.0
          if addr = ipaddr.xx then do
          say 'we matched the ipaddr' addr conn jobn
          end
   end
end
```

SYSPLEX Integrity Checker

- ***SYSPLEX Integrity Checker – FROM NETVIEW Written in REXX:***

```
  say 'WELL DONE!'
   say 'WELL DONE!'
   say 'WELL DONE!'
exit
  if jobn = 'NETEXIG' then do
  'mvs d tcpip,,n,conn,cli='jobn
  exit
/*                                                  */
   /* say 'we are in record ='msg.ii */
   /* say 'w1 ='w1 'w2 ='w2 'stat ='stat 'w4 ='w4 */
```