

Session V<sub>a</sub>: **Service Management**

Chair: Rachid Guerraoui, *EPFL/HP*

---

# A Distributed Near Real-Time Billing Environment

---

TINA'99  
April 14, 1999

Joel J. Fleck, II  
Senior Strategic Architect  
Communications Industry Business Unit  
Hewlett-Packard  
Phone: +1 732.562.6109  
Email: joel\_fleck@hp.com



---

## Outline of Talk

---

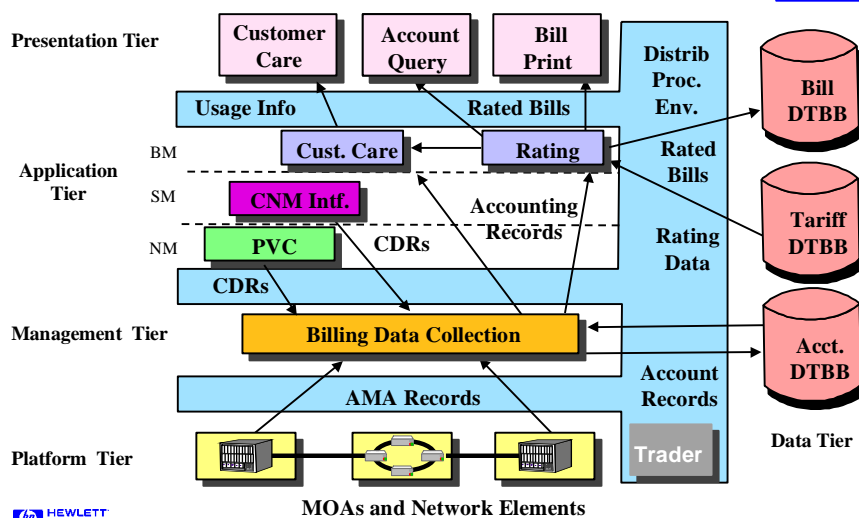
- Description of Problem
- Solution Overview
- Solution Architecture
- Simulation of Architecture and Results
- Conclusions and Futures



## Solution Objectives

- Solution Architecture should provide:
  - Near real-time billing (to support credit card, credit verification, pre-paid plans, interactive customer query, customer profiling, ...),
  - Flexible, modular software design (to facilitate deployment of new service without impact on existing ones),
  - High availability (downtimes similar to network infrastructure components),
  - Scalable platform (deployment to service providers from small (< 100 CDRs/sec) to very large (>10,000 CDRs/sec) ,
  - Interfaces to existing provisioning and bill data storage systems.

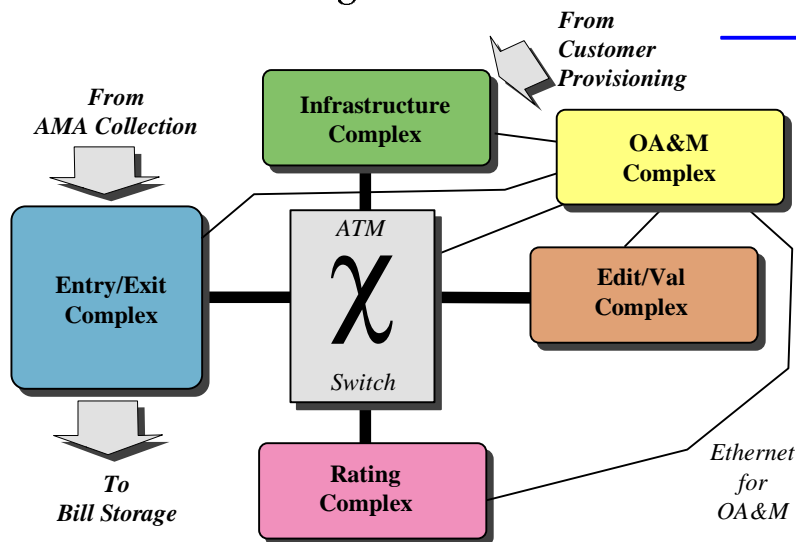
## Billing Architecture



## Outline of Talk

- Description of Problem
- [Solution Overview](#)
- Solution Architecture
- Simulation of Architecture and Results
- Conclusions and Futures

## Bill Processing Module Architecture



---

## Basic Solution Components

---

- Definitions:
  - **Complex**: A scalable building block implemented by a coherent set of processes that provide and utilize common functionality (processing, data and communications)
  - **Trader**: Facility that provides "best-fit" location and naming transparencies; the "glue" that provides communication path between and within complexes

---

## Benefits of Using an Architecture built from Complexes and Traders

---

- **Availability**: Provide multiple levels of support by:
  - Transparently providing multiple processors in each complex,
  - Transparently providing multiple complexes in the system,
  - Transparently reconfiguring processors between/within complexes.
- **Scalability**: Performance can be increased in two ways:
  - Adding additional processors to a complex, and
  - Adding additional complexes to the system.
- **Flexibility**: New or upgraded functionality can be added without affecting existing functionality and system operations.

---

## Proposed Complexes Were Derived from Assumptions and Objectives

---

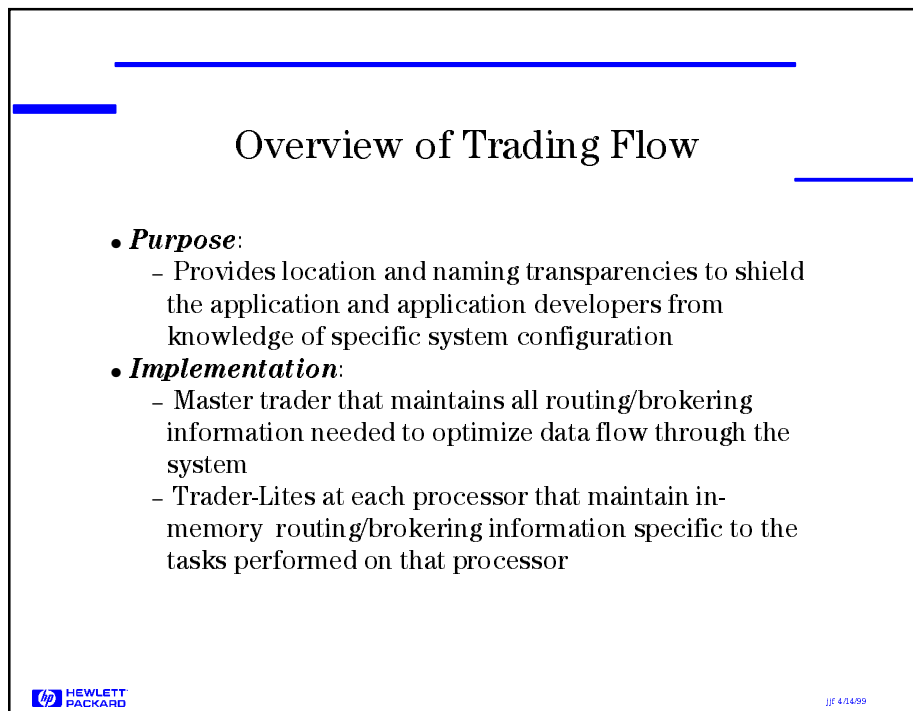
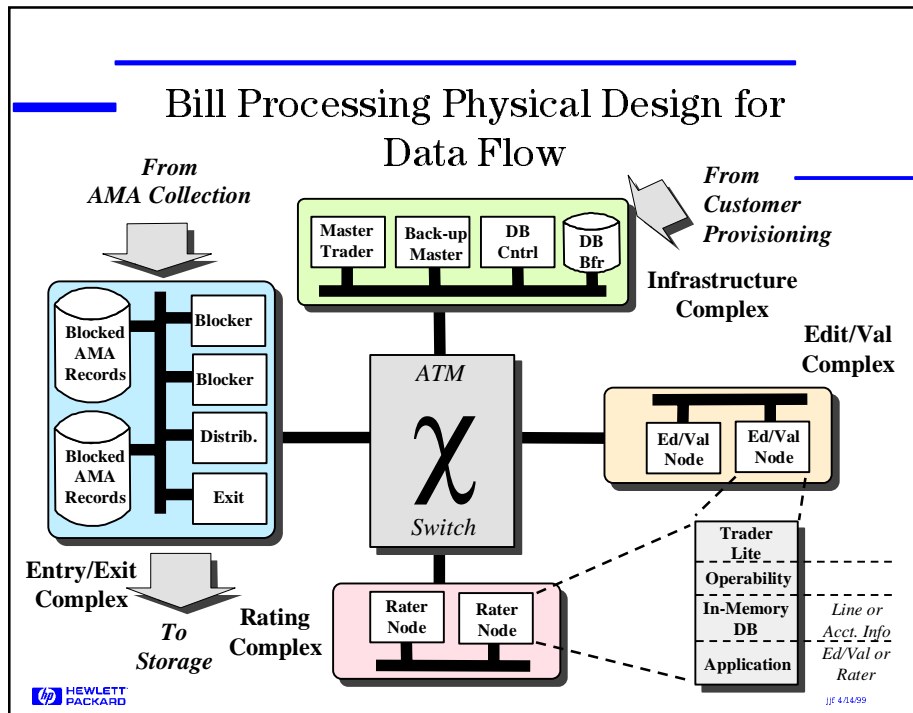
- **Near Real Time Billing**
  - Rating Complex
- **Additional Feeds in Future**
  - Entry/Exit Complex
- **Support for In-memory DB and interface to Cust. Prov.**
  - Infrastructure Complex
- **Maintainability**
  - OA&M Complex
- **No Lost Records**
  - Entry/Exit Complex
- **100% Billable Records**
  - Edit/Val Complex
- **Scalability/Flexibility**
  - Infrastructure Complex

---

## Outline of Talk

---

- Description of Problem
- Solution Overview
- **Solution Architecture**
- Simulation of Architecture and Results
- Conclusions and Futures



---

## Overview of Data Distribution Flow

---

- **Observation:**
  - High speed data (low-latency) access must be able to function during low-speed data updates,
  - Customer provisioning systems typically provide high-latency data access updates
- **Solution:**
  - Infrastructure Complex:
    - ▼ Provides in-memory staging of customer provisioning data
    - ▼ Provides local storage of customer data in case of processor re-assignment or new processor addition

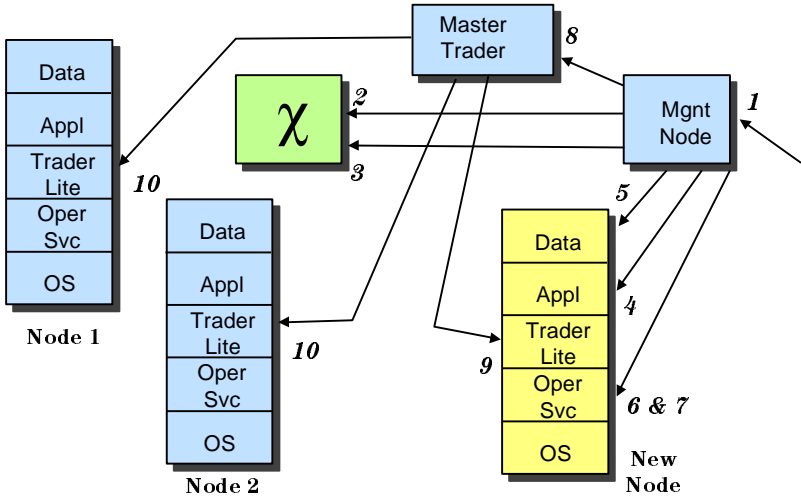
---

## Overview of Transaction Mechanism

---

- **Problem:**
  - Need to ensure "No Lost Records"
- **Solution:**
  - Specify an Entry/Exit complex that:
    - ▼ Groups incoming records into blocks
    - ▼ Queues these blocks upon entry to the bill processing system
    - ▼ Dispatches blocks to next appropriate processing complex
    - ▼ Maintains status of blocks within bill processing system
    - ▼ Dispatches billed information upon reception of billed records
    - ▼ Removes dispatched blocks from queue

# Details for Node Addition



# Outline of Talk

- Description of Problem
- Solution Overview
- Solution Architecture
- **Simulation of Architecture and Results**
- Conclusions and Futures

---

## Billing Simulator Design

---

- Simulator modeled following:
  - Entry/Exit Complex
  - Editing/Validation Complex
  - Trader portion of Infrastructure Complex
  - Trader-lite
  - Management Complex
- Goal of simulator:
  - “Proof of Concept” for architecture
  - Provide confidence that performance objectives could be met
  - Evaluate impact of blocking size on performance
  - Demonstrate distributed trading

---

## Billing Simulator Results:

No Character Editing

---

Packet Size	CDRs Per Sec	Mbytes Per Sec
256	514	131,522
512	1,006	257,545
1,024	1,917	490,656
8,192	6,339	1,622,821
16,384	7,802	1,997,318
24,576	8,319	2,129,636
32,768	8,419	2,155,222

---

## Billing Simulator Results:

Editing Every Other Character

---

Packet Size	CDRs Per Sec	Mbytes Per Sec
256	530	135,665
512	1,006	257,545
1,024	1,855	474,954
8,192	5,523	1,413,876
16,384	6,408	1,640,368
24,576	6,848	1,753,049
32,768	7,644	1,956,833

---

## Outline of Talk

---

- Description of Problem
- Solution Overview
- Solution Architecture
- Simulation of Architecture and Results
- **Conclusions and Futures**

---

## Conclusions

---

- Trader/trader-lite and trading time from ~ 1 sec to 50 millisecs
- Blocking size of between 28Kbytes and 32Kbytes yielded optimal results
- Throughput between 7.25 and 9 K CDRs per sec
- A system designed with the Complex/Trader/Trader-lite Mechanisms results in designs that are:
  - **Scalable** from small system implemented on a single host through very large systems.
  - **Flexible** to support the addition of new or upgraded features with no impact on existing system operations.
  - **Highly available** through the capability of supporting rapid re-configuration to work around failed components/functionality.

---

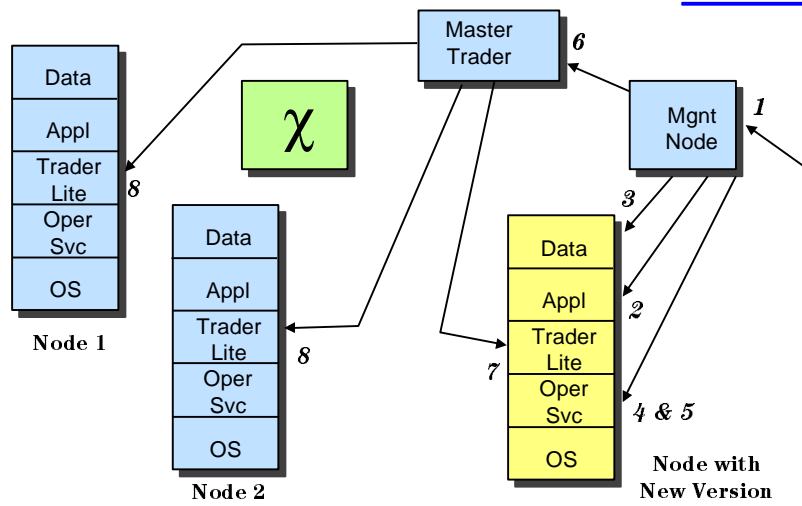
## Thoughts for the Future/Modifications

---

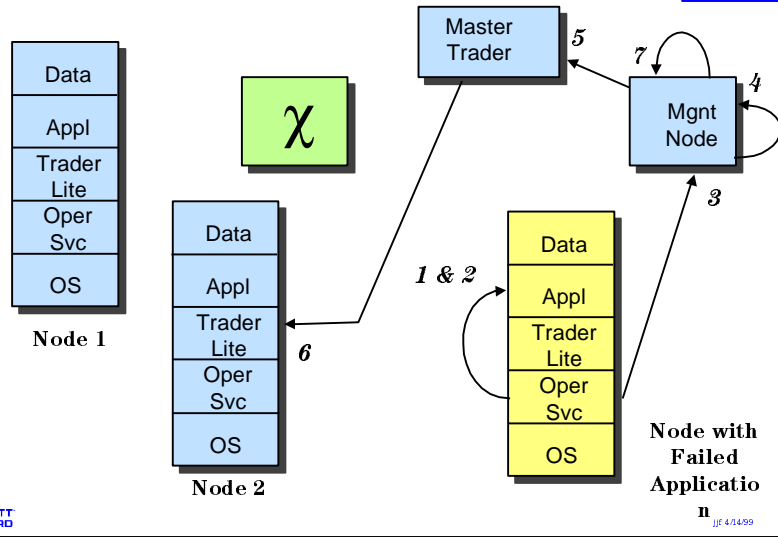
- Flag on each node to force read from Trader-Lite (i.e., flush cache)
- "Still-Alive" ping from management node
- Report from Trader-Lite to management node if a new trade is requested without cache flush command (i.e., local node detected a communication failure)
- Ability for Trader-Lites to mark destination "bad"; (maybe a "don't select" constraint field always available)
- Investigate usage of Software Fault Tolerant Technology for Software Fault Tolerance within Complex Modules
- Integrate framework with billing store, customer care access, and customer interactive access

# Backup Slides

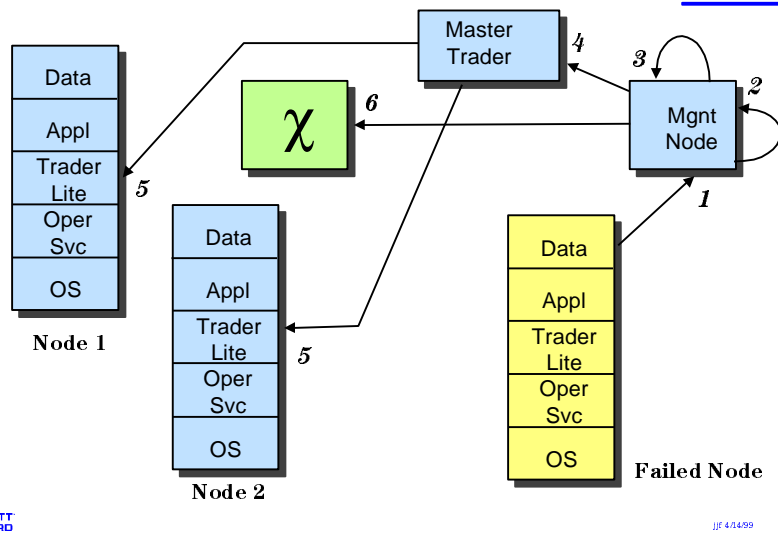
## Details for New Application Version Addition



## Details for Application Failure



## Details for Node Failure



## **Implementation and Interoperability experiences with TINA Service Management Specification**

**Sohail Rana**

**BT Labs.**

Martlesham Heath, Ipswich, UK.

sohail.rana@bt.com

M.A. Fisher  
BT Labs, Martlesham Heath  
Ipswich, UK  
mike.fisher@bt.com

C. Egelhaaf  
GMD-FOKUS  
Kaiserin-Augusta-Allee 31, D -10589  
Berlin, Germany  
egelhaaf@fokus.gmd



Sohail Rana, BT Labs, TINA 99, 12/4/99

1

### **Talk Summary**

- Project Overview
- Service Management Platform Build
  - TINA Retailer-Consumer Reference point Implementation
    - Implementation agreements
    - Different components
    - Test procedures
    - Results
  - TINA Retailer-Provider Reference point Implementation
    - ....

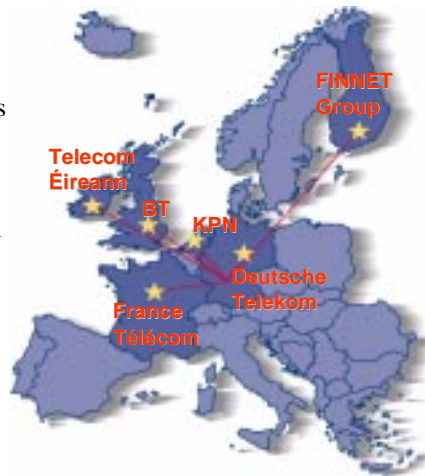


Sohail Rana, BT Labs, TINA 99, 12/4/99

2

## Project Overview

- Assessment of distributed object technologies based on experiments with commercially available products
- Experiments using CORBA middleware technologies based on architectural principles of TINA
- Conduct joint experiments at 6 locations in Europe
- Feedback to standardisation organisations
- The final Demonstration.



Sohail Rana, BT Labs, TINA 99, 12/4/99

3

## Platform Build

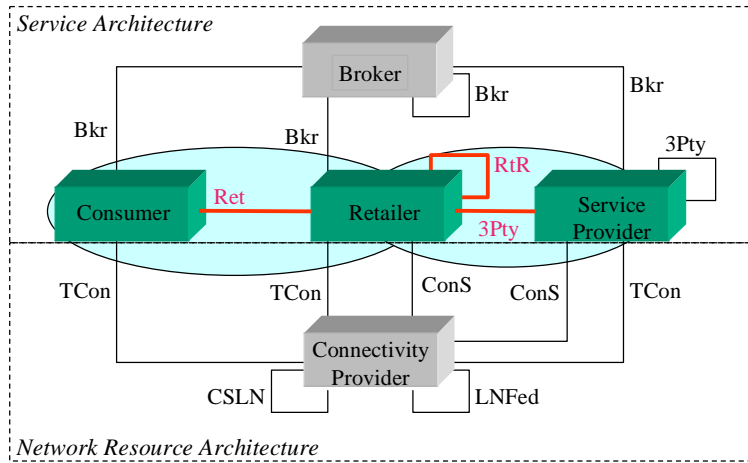
- Build a set of service management platforms based on TINA Consumer/Retailer relationship
  - heterogeneous,
  - Independently developed,
  - Connected by ISDN
- Investigate interoperability
- Implement Third party Reference point



Sohail Rana, BT Labs, TINA 99, 12/4/99

4

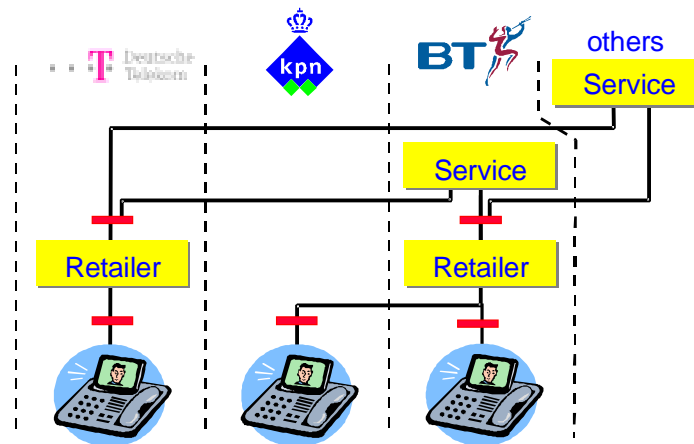
## TINA Business Model



Sohail Rana, BT Labs, TINA 99, 12/4/99

5

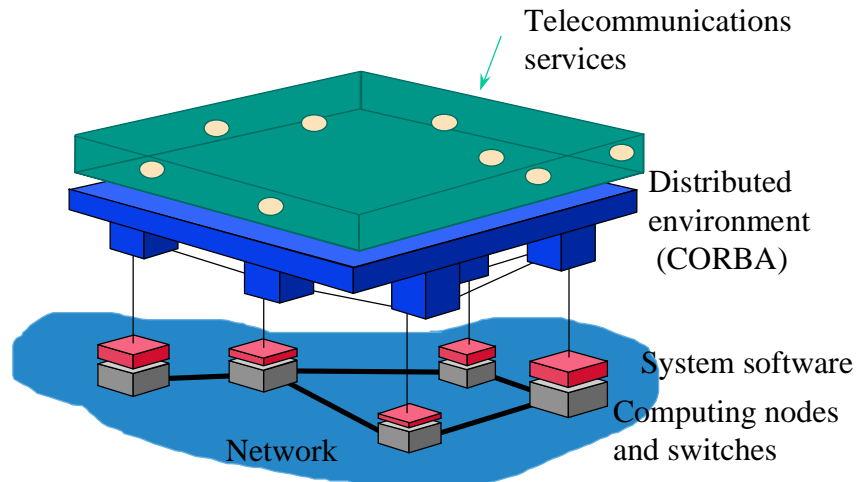
## The End Result



Sohail Rana, BT Labs, TINA 99, 12/4/99

6

## Architecture and Technology

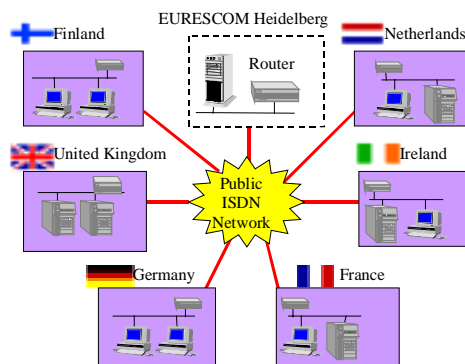


Sohail Rana, BT Labs, TINA 99, 12/4/99

7

## Network Connectivity

- LAN at each site connected to the ISDN network via 2 x N-ISDN (2B+D) Router.
- ISDN-30 router in Heidelberg
- Star topology. IP over ISDN



Sohail Rana, BT Labs, TINA 99, 12/4/99

8

## Distributed Processing Environment

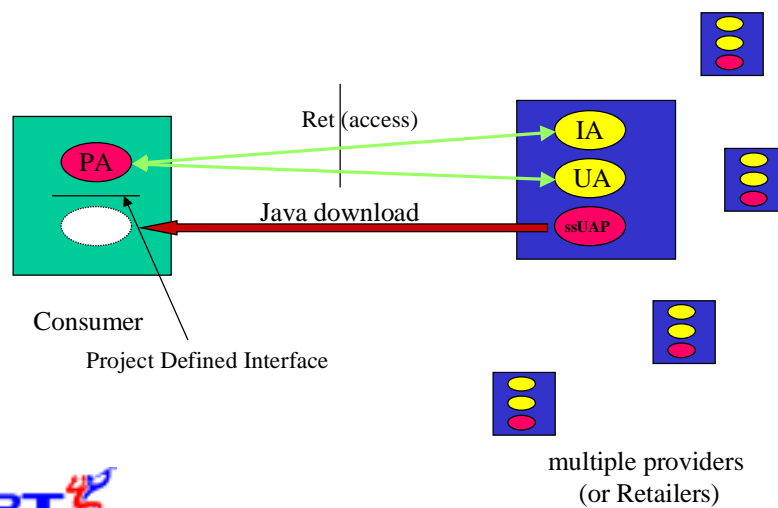
- ORBs Used
  - Iona's Orbix and OrbixWeb
  - Inprise's Visibroker for C++ and Java
  - Sun's NEO
  - HP's Distributed Smalltalk
  - Chorus COOL
  - Olivetti Oracle Research's OmniORB
- CORBA Services
  - Naming Services (Federated)



Sohail Rana, BT Labs, TINA 99, 12/4/99

9

## A Scenario



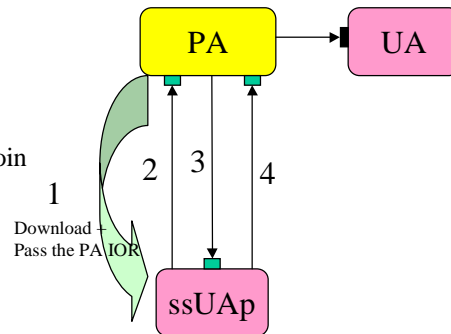
Sohail Rana, BT Labs, TINA 99, 12/4/99

10

## Interfaces between PA and ss\_UAp

### Sequence of Operations

1. Pass the initial Reference
2. Pass Reference to PA
3. Notify whether to Start or Join
4. Initiate Start or Join

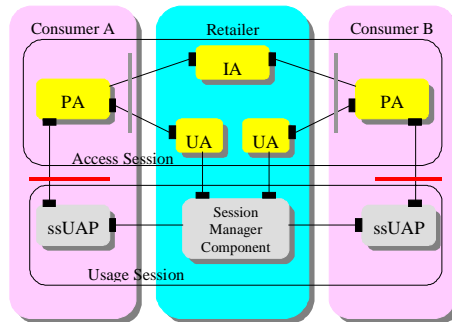


## TINA Ret Implementation

- Implementation Agreements
  - A set of scenarios and operation semantics
  - A defined subset of the TINA Ret (Access) specification (v.1.0)
  - A modification to Ret1.0 defined in this Task (i.e. listRequiredServiceComponents)
  - A set of interfaces between PA and ssUAp in the consumer domain defined.
  - The use of Java (1.1) byte code for download (ssUAP)
  - End user service can be applet or Java Application.



## TINA Ret Component



PA: Provider Agent, IA: Initial Agent, UA: User Agent ,  
 UAP: User Application  
 — : Project (P715) Defined Interfaces.  
 — : TINA Ret, — : TINA Components.  
 ■ : Standard Interface ■ : Proprietary Components.

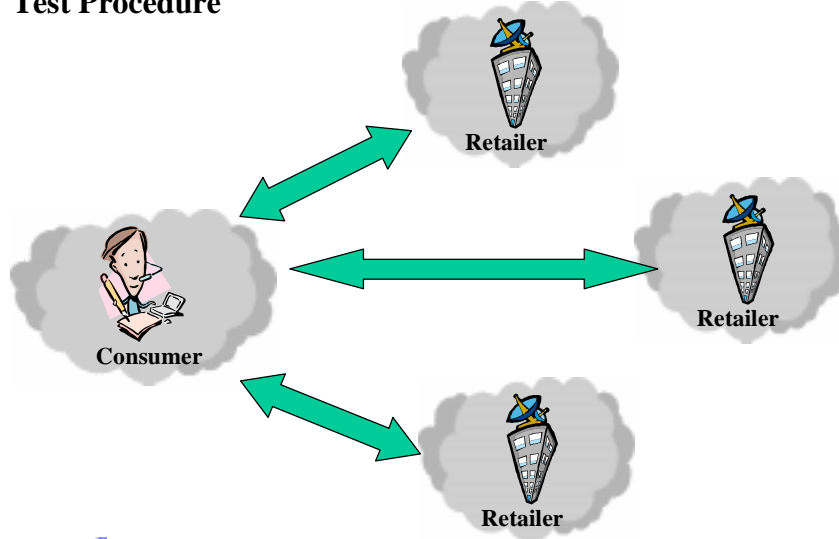


## End user Applications

- Services
  - Single user :- Counter, Media on demand, Value added web, Distributed Scheduler
  - Multiusers :- Connect4 game, Shared White board.
  - Stream based:- Video conferencing.
  - Legacy:- Audio conferencing, Virtual world, Surveillance camera.



## Test Procedure



Sohail Rana, BT Labs, TINA 99, 12/4/99

15

## Test Results

- Platform Interoperability
  - 5 Partners implemented and tested successfully (25 combinations)
  - Invitation implemented by two partners (BT and DT)
- ORB Interoperability
  - Retailer ORBs :- Orbix, OrbixWeb, Neo, DST and Visibroker for Java and C++
  - Consumer ORBs :- OrbixWeb, Visibroker for Java

**They all interworked. But ...!**



Sohail Rana, BT Labs, TINA 99, 12/4/99

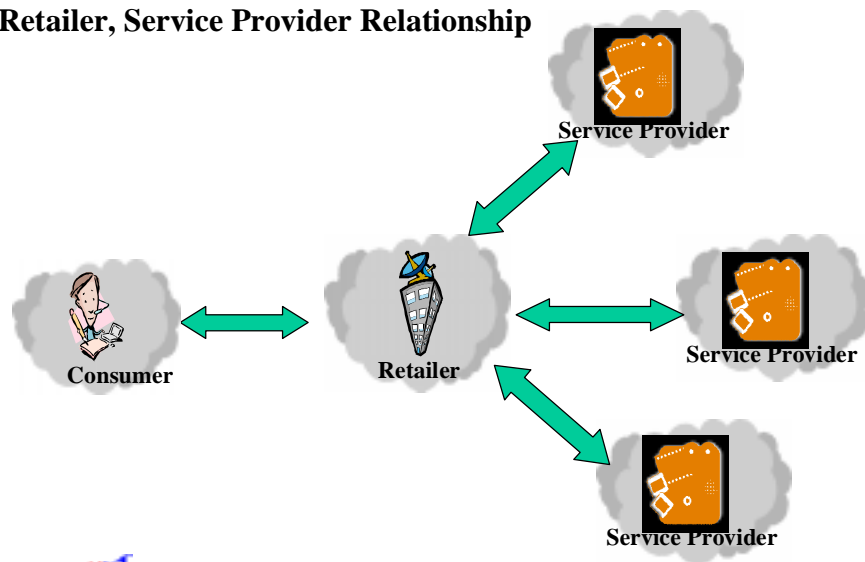
16

### Problems.....

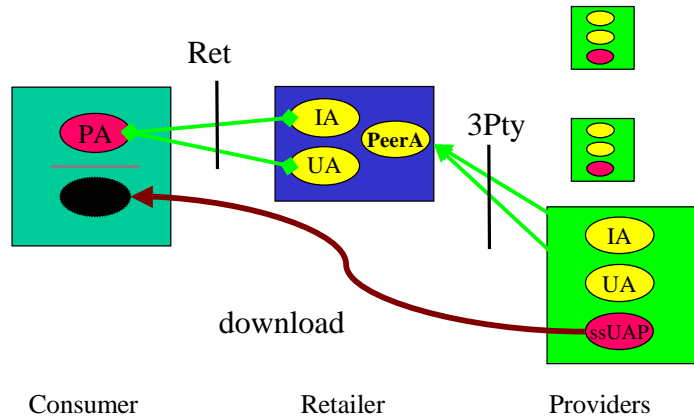
- ORB Interoperability  
mostly OK but:
  - structures in CORBA::Any
  - derived interfaces passed as base
  - rebind following closing of IIOP connection
  - LOCATION\_FORWARD
- Consistency Problems :- Java versions, Java ORBs
- Browser Problems:- Inconsistency with browser Security.



### Retailer, Service Provider Relationship



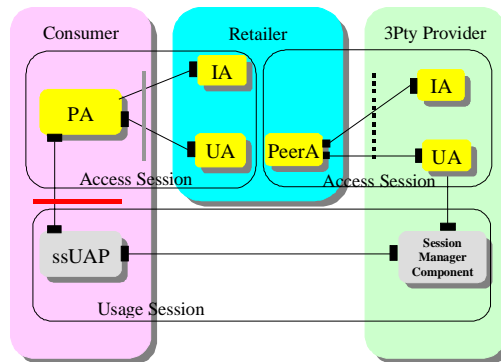
## Test Scenario



Sohail Rana, BT Labs, TINA 99, 12/4/99

19

## Third Party Implementation



PA: Provider Agent, IA: Initial Agent, UA: User Agent ,  
 UAP: User Application PeerA : PA like Component.  
 — : Project (P715) Defined Interfaces.  
 — : TINA Ret, — : TINA Components.  
 ··· : RiR/3Pty (Reuse of Ret).  
 ■ : Standard Interface ■ :- Proprietary Components.



Sohail Rana, BT Labs, TINA 99, 12/4/99

20

## Results

- Third Party Implementation
  - Implemented by two partners (BT and DT)
  - Successfully tested by five partners
- Problems with the Reuse of Ret
  - Identifier problems, e.g. Service ID, Session ID, Invitation ID.
    - Suggestions :- Use of structured strings, i.e. URL, UUID (Universal Unique Identifiers)



## Conclusion

- TINA Ret Specification can be put into practice for a heterogeneous multivendor environment.
- OMG's CORBA and TINA forms a basis to create an open distributed environment for telecom services
- TINA Ret specification - reuse shown.
- Feedback to the SARP working Group and ORB Vendors.
- But... problems with client side ORBs and Browsers but they are evolving.



Web-based access  
to the IN Service Management  
- a TOCTIS implementation -

Oki Electric Industry Co.,Ltd.  
Naoko Nakagawa

Copyright © OKI Electric Industry Co.,Ltd.

Contents

- *What is TOCTIS ?*
- *TOCTIS CNM Service Overview*
- *TOCTIS Component Objects*
- *Brief GUI Images*
- *Conclusions*

## 1. What is TOCTIS ?



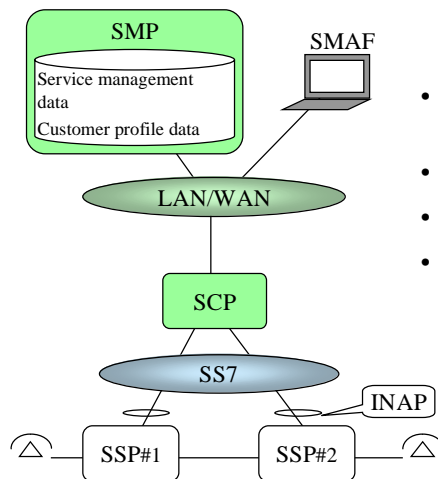
### **TOCTIS :**

OKI's software solution for Intelligent Network.

- Supports SCF,SDF,SMF and SMAF,
- Implements TINA based session models,
- Applies Web and JAVA technologies,
- Integrates with existing systems by wrapping technology.

2

## 2. Service management in IN architecture

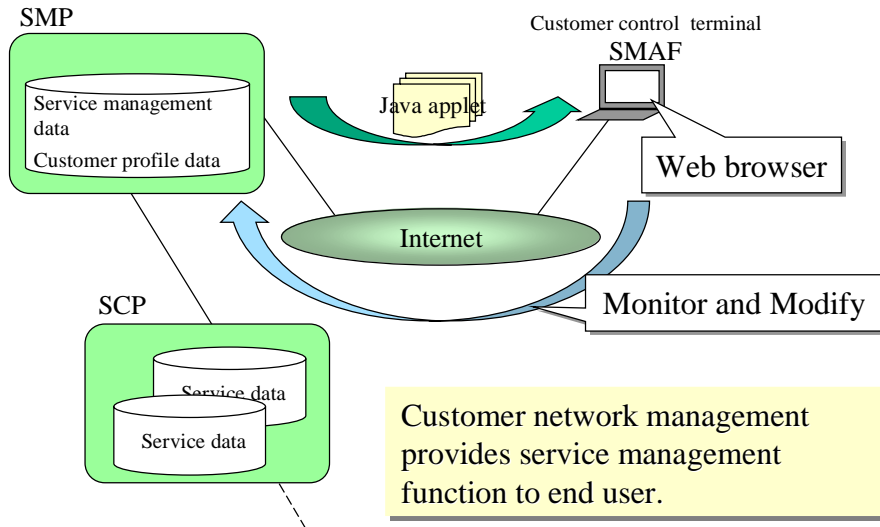


- Customer specific service definition by SMP and customer control terminal.
- Service verification
- SLP creation
- Service simulation

SSP: Service Switching Point  
SCP: Service Control Point  
SMP: Service Management Point  
INAP: IN Application Protocol  
SLP: Service Logic processing Program

3

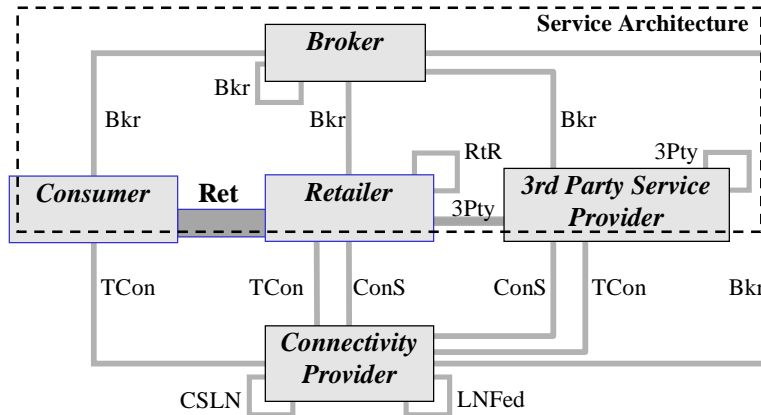
### 3. Customer network management overview in TOCTIS



### 4. Service of TOCTIS customer network management

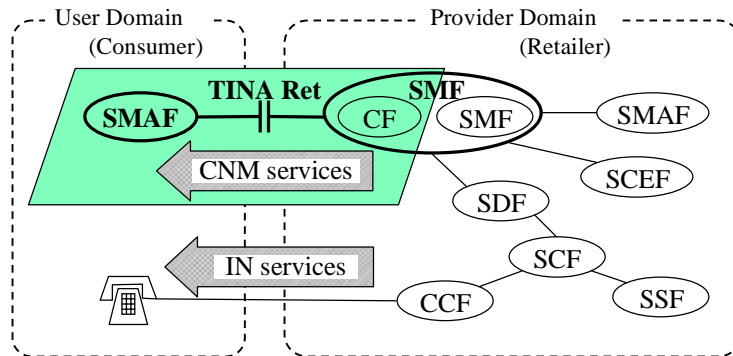
Service	Outline
Monitor and Modify Contractual Data	Customize IN service by referring and modifying contractual data and create new contract.
Monitor Usage Info.	Collect statistical information of IN service calls.
Monitor User Access History	Monitor usage information of customer control service.
Monitor and Modify User Profile	Management of Customer Operator: create or delete a customer operator and change password.

## 5. TINA business model



6

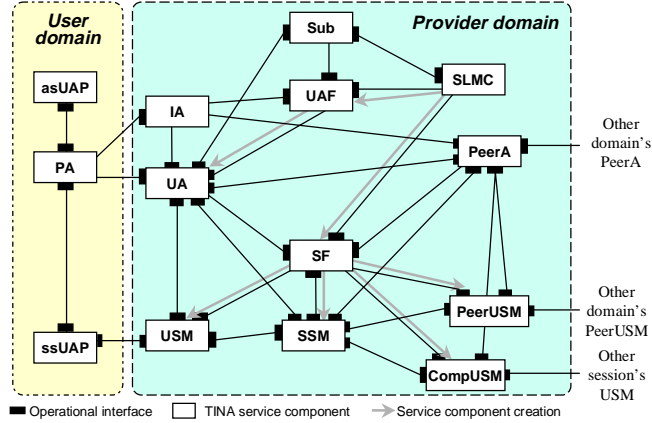
## 6. Application of TINA to the IN CNM service



SMF: Service Mngmt. Func.      SMAF: Service Mngmt Access Func.  
 SCEF: Service Creat. Env. Func.      SDF: Service Data Func.  
 SCF: Service Control Func.      SSF: Service Switching Func.  
 CCF: Call Control Func.      CF: Customer Network Mngmt Func.  
 /: Customer Network Management Service

7

## 7. TINA service components

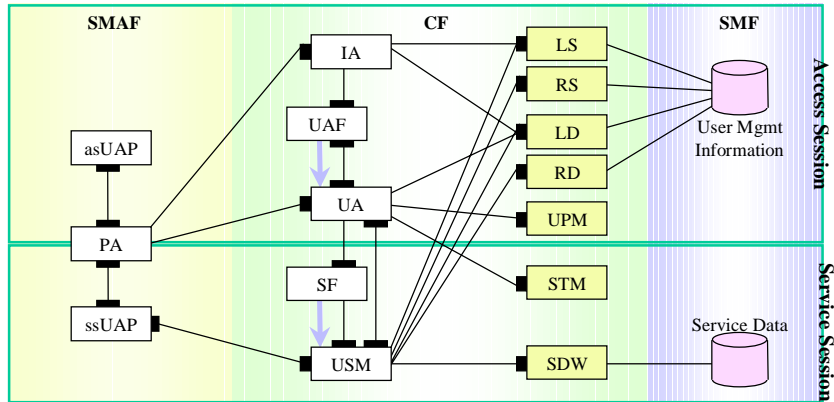


PA: Provider Agent  
 asUAP: Access Session User Application  
 Sub: Subscription Management Component  
 SLMC: Service LifeCycle Management  
 ssUAP: Service Session User Application  
 USM: User Service Session Manager  
 CompUSM: Composer Usage Session Manager

IA: Initial Agent  
 UA: User Agent (named/anon)  
 UAF: User Agents Factory  
 SF: Service Factory  
 SSM: Service Session Manager  
 PeerA: Peer Agent  
 PeerUSM: Peer Usage Session Manager

"Service Component Specification Computational Model and Dynamics," TINA-C, Jan. 19, 1998.

## 8. Computational objects



asUAP: Access Session User Application  
 ssUAP: Service Session User Application  
 PA: Provider Agent

IA: Initial Agent  
 UAF: User Agent Factory  
 UA: User Agent  
 SF: Service Factory  
 USM: User Service Session Manager

LS: Lookup Security  
 RS: Register Security  
 LD: Lookup Distribution  
 RD: Register Distribution  
 UPM: User Profile Manager  
 STM: Service Trading Manager  
 SDW: Service Data Wrapper

## 9. Conclusions



- TINA Retailer Reference Point  
Retailer reference point of TINA business model is adopted to our TOCTIS CNM boundary.
- Web-based technology  
Web-based technology together with Java applet is applied on user interface of TOCTIS CNM.
- Object Wrapping technology  
Object wrapping technology is applied for integration with existing SMF software.

10



```
/* Copyright 1999 Oki Electric Industry Co., Ltd. */
#ifndef TINASessionModel_ih
#define TINASessionModel_ih
#include "TINASessionModel.hh"
class TINASessionModel_i {public: class i_SessionModel_i {
public:
    i_SessionModel_i(char* i_SessionModel_i (ObjectReferenceImpl
*IT_OR) {});
    i_SessionModel_i () : CORBA::Object (1) {} };
DEF_TIE_TINASessionModel_i_SessionModel(i_SessionModel_i);
#endif
```

11