INTELLIGENT NETWORK SIMULATION PLATFORM
INGENUITY

Dr. George C. Atallah (Project Director & Co-PI)
Dr. John Baras (Co-PI)
Dr. Armand Makowski (I)
Students: Shah-An Yang, Kevin Wright
Consultant: Kap Jang
AGENDA

- Hybrid Network (Physical & Logical Views)
- INGENUITY Architecture and Object Oriented Design
- Voice Over IP Implementation Elements
- INGENUITY and Opnet Comparison
- INGENUITY Demonstration
- Statement of Work (Extension of INGENUITY)
- Discussion
PROJECT OBJECTIVE

• Acquire Intelligent Network Simulation Platform based on the following technical requirements:
  – Client/Server Architecture (equal to distributed processing, & open architecture)
  – Web Based Interface
  – Java Programming
  – SS7 Network
  – Payload Network (ATM, ISDN, Satellite)
  – Customized Multimedia Applications and Network Configurations
Hybrid NETWORK LOGICAL VIEW

ATM | Packet Switch Controller

Gatekeeper

IP Network

Gateway

SCP's

SMS's

SCE's

customer's applications

Phone Images Comp

PSTN/ISDN

SS7 Signaling Network

IPs

SSPs

STPs

voice

data

800

Images

customer's applications

Phone

Images

Comp

G C. Atallah  page5  11/15/99
IN SIMULATION WEB BASED CLIENT/SERVER ARCHITECTURE

CLIENTS DOMAIN

end users’ applications, images, 800, emails, ...

INTERNET/WWW DOMAIN

SERVERS DOMAIN

DATABASE DOMAIN
Ingenuity Development Environment

- Software developed using Sun’s Java 2 Software Development Kit (SDK) in a Windows environment.
- Java 2 SDK is available on 3 platforms, Windows, Solaris/SPARC, and Solaris/Intel.
- To run the application, all that is required is a copy of the Java 2 Runtime Environment.
- To run as an applet, the Java Plug-in software must be installed.
- Both the Runtime Environment and the Plug-in are available for Windows and Solaris/SPARC systems.
- Third party (non-Sun) support exists for other platforms.
INGENUITY Key Components

- Graphical User Interface
  - Main Menu
  - Key Panel
  - Work Area
  - GUI Property Object
  - Event Queue
  - GUI Property Object
  - Simulation Elements
  - Extended Simulation Elements
  - Payload Elements
  - SS7 Elements
  - SS7 Payload Elements

- User Interface
  - Input/Output
  - File Access
  - Remote File Access
  - Graph
  - Bar
  - Line
  - Pie
  - Model Saving/Loading
  - Import/Export Data

- Simulation
  - Transactions
  - Workload
  - Network
  - Nodes
  - Links
  - Workload
  - special case of
  - Network
  - consists of
  - Nodes
  - Links
  - Simulation Elements
  - Extended Simulation Elements
  - Payload Elements
  - SS7 Elements
  - SS7 Payload Elements

- Remote File Access
  - consists of
  - special case of
  - File Access
  - supports
  - Import/Export Data

- Graph consists of
  - Bar
  - Line
  - Pie
  - sets of
  - uses

- Model Saving/Loading
  - sets of
  - uses
INGENUITY Objects

Simulation

SS7 Simulation

Event Queue

Payload Simulation

Combined Simulation

Payload Network

Combined Simulation

Abstract Simulation

Payload Simulation

Simulate SS7 Network

Simulate Both Networks

Simulate Payload Network

Network

SS7 Network

Abstract Network

Payload Network

Simulate Payload Network

Node

SSP

STP

SCP

IP

ISTP

Node

ATM

ISDN

Sat.

G C. Atallah  page9  11/15/99
Physical View

- SimulationServer
  - HttpServer
  - FileManager
  - FileTransfer
- SimulationApplication
- JVM
- BrowserClient
- User
- WebBrowser
- SimulationApplet
- JRE
- ApplicationClient
- Manager

1:1 *
INGENUITY High Level Object Relationships

SimulationGUI

EventQueue 1 Simulation

Ss7PayloadSimulation 1 Payload Network

Ss7PayloadWorkload

Ss7PayloadScenario

PayloadNetworkTransaction

Ss7NetworkTransaction

Ss7TransactionInstance

Rx

Sink

Node

Link

TRLink

Simulation

Network

RoutingTable

Ss7Payload

Ss7Network

Ss7NetworkTransaction

PayloadNetworkTransaction

PayloadTransactionInstance

Package Legend

Gui

InteliNet

InteliNet.Ss7

InteliNet.Payload

InteliNet.Ss7Payload
Simulation Class

Simulation
  SimTime

EventQueue
  Fire()

SimEvent

Workload

Network

Ss7 Simulation

Payload Simulation

Ss7Payload Simulation
Workload Class

- **Workload**
  - *Scenario*
  - Transaction
- **Scenario**
  - Transaction
    - **Ss7 Transaction**
      - Ss7Payload Transaction
    - Network Transaction
      - createTransactionInstance()
    - Transaction Instance
- **Transaction**
  - Payload Transaction
    - PayloadNetwork Transaction
• Event-Based Simulation
  – Events generated by network components: Packet generation; transmit, receive, process, and sink data packet
  – Events are time-stamped and drive the simulation forward

• Event Queue
  – Ordered by time stamp
  – Process as follow:

• OO Design
Workload and Network Interaction

**Scenario 1**

**Payload Network**

1) load Payload Transaction
2) Instantiate Ss7 Transaction
3) load Ss7 Transaction
4) Ss7 Transaction
5) Activate Payload Transaction
6) Payload Transaction
7) Instantiate Ss7 Transaction

**Ss7 Network**

1) load Payload Transaction
2) Instantiate Ss7 Transaction
3) load Ss7 Transaction
4) Ss7 Transaction

**Ss7 & Payload Workload**

1) Transaction Event
2) Transaction
3) Payload Transaction
4) Ss7 Transaction

**Transaction 1**

1) Generate Transaction
2) Activate Transaction
Transaction and Node Interaction

Scenario

1) Instantiate Payload Transaction

Ss7Payload Transaction

Payload Transaction

Payload Transaction Instance

Generate Packet

SS7 Transaction

SS7 Transaction Instance

Generate Packet

2) Load Payload Transaction Instance

3) Instantiate SS7 Transaction

4) Generate SS7 Packets

5) Instantiate more SS7 Transactions

6) Activate Payload Transaction Instance

7) Generate Payload Packets

Ss7PayloadNode

Payload Node

SS7 Node

G C. Atallah   page21   11/15/99
Transmission Model

- Assumption 1: The transmitter is a server with a service time of 0.
- Assumption 2: Packets will immediately arrive at the destination after service (0 propagation time).
- To accommodate both assumptions and account for previous packet processing time, the server will take a vacation for a duration equal to the size of the packet divided by the bandwidth allocated for transmission.
Transmitter Model
Buffer and Bandwidth Allocation Management

1. Packets arriving at this transmitter enter the input queue.

2.a) Packets feed from the input queue into the buffer for the transaction they belong to.

2. b) Or if there is no buffer yet for this transaction, bandwidth is allocated based on the transaction requirements and a new buffer is created.

3. Packets are sent down the link with the allocated bandwidth.

Packet Input Queue
Transaction Buffers
Link Bandwidth
Free Bandwidth

These buffers are taken down as soon as they empty.

These buffers remain until a call takedown message is received.

A call setup message is received. If enough bandwidth is free in the transmitter, then it will be set aside to service this call.

Packets that arrive with a previous call setup are put directly into the appropriate buffer for service.
Routing

- Routing is performed using a static table constructed during simulation initialization.
- Djstra’s algorithm used to determine the minimum weight paths.
- Link weights are provided by user as input.
- Each node is given a mapping that takes a destination node and outputs the next node along the shortest path.

Simple Example:

```
<table>
<thead>
<tr>
<th>Source Node</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>X</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>X</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>B</td>
<td>B</td>
<td>X</td>
</tr>
</tbody>
</table>

Global Routing Table

<table>
<thead>
<tr>
<th>Destination Node</th>
<th>Next Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
</tr>
</tbody>
</table>

Route Mapping in Node C
```
Call Model

- Based on Transaction Instance
- Payload Transaction
  - Payload Transaction Instance
- SS7 Transaction
  - HandShake Request Transaction Instance
  - HandShake Response Transaction Instance
  - SspScp Request Transaction Instance
  - SspScp Response Transaction Instance
  - OpenCircuit Notify Transaction Instance
  - ReleaseHand Transaction Instance
Setup Channel

1. Workload
2. SSP/ISDN
3. ISDN
4. SSP/ISDN
5. SSPP/ISDN
6. Workload
7. SSP/ISDN
8. ISDN
9. SSP/ISDN
10. ISDN
11. SSP/ISDN

CurrentNode

HandShake Request

HandShake Response

Allocate Channel

Payload
Setup Channel

- Ssp/ISDN node has a Ssp entity and ISDN entity inside.
- ISDN entity may receive a Payload Transaction Instance from Workload or a HandShake Request Transaction Instance through STP network. (1)
- ISDN entity identifies the NextNode on the path toward destination and initiates handshake to the NextNode (2)
- SSP entity create new HandShake Request Transaction Instance pass to STP network toward the NextNode (3) - (4)
- SSP entity in NextNode receives HandShake Request and pass to ISDN entity. (5)
- ISDN entity test status whether it can allocate channel and initiates HandShake Response Transaction Instance with the status. (6)
- ISDN entity in CurrentNode receives HandShake Response from STP network. (8) - (10)
- ISDN entity allocate channel depending on the return status (11)
SSP-SCP Transaction

1. Workload
2. SSP/ISDN
3. SSP
4. STP
5. SCP

Steps:
1. Workload to SSP/ISDN
2. SSP/ISDN to ISDN
3. SSP
4. STP to SCP
5. SCP to STP
6. STP to STP
7. ISDN to SSP/ISDN
8. SSP/ISDN to Workload

Messages:
- Black arrow: SspScp Request
- Purple arrow: SspScp Response
- Red arrow: HandShake Request
- Gray arrow: Payload
SSP-SCP Transaction

- ISDN entity may receive a Payload Transaction Instance from Workload or a HandShake Request Transaction Instance through STP network. (1)
- If the associated Payload Transaction is for 800 Call, and a SSP/ISDN node can handle 800 call, it initiates SspScp Request to identify destination. (2)
- SSP entity in the SSP/ISDN node creates SspScp Request Transaction Instance toward a SCP and pass through STP network. (3) - (4)
- SCP node receives the Request and initiates SspScp Response Transaction Instance toward the originated SSP/ISDN node with Destination Node information. (5)
- ISDN entity recieves SspScp Response from SCP. (6) - (7)
- ISDN entity initiates HandShake Request toward Destination Node. (8)
• If Destination Node receives HandShake Request and complete allocate channel by initiate OpenCircuit Notify Transaction toward Source
If a Payload Transaction is completed in Destination and the Destination initiates a ReleaseHand Transaction toward the Source to deallocate channels.
Workload Model Parameters

- Simulation duration common to all scenarios.
- Tabbed windows separate the multiple scenarios.
- Each scenario has its own name, arrival rate and number of users.
- Transactions are defined separately for each scenario.
- Tabbed windows separate the multiple transactions.
- Each transaction has its own name and probability.
- Each transaction can be either an 800 call or a direct call model.
Workload Model  Parameters (continued)

- Both SS7 and payload transactions are defined the same way in separate tabbed windows.
- The data size is the size of the entire transaction in bits.
- A transaction can be divided into multiple frames.
- The think time gives the time it takes for the source to completely generate the transaction.
- The source probability pulldown menu contains all the possible sources for this transaction.
- Each source is assigned a probability in the adjacent text field.
- The sink probabilities work similarly.
- The intermediate path allows the selection of any ordered set of nodes that the transaction must pass through.
Transaction Generation

- Arrival Rate gives the transaction arrival rate for the per user.
- The No. of Users multiplied by the Arrival Rate gives the actual transaction arrival rate for the entire scenario.
- The probability multiplied by the overall transaction arrival rate for the scenario gives the arrival rate of this particular transaction.
- The transactions arrive with an exponential distribution with parameter $\lambda$ given by the above computed arrival rate.
- In this particular example, Transaction 1 will have an arrival rate of $0.5 \times 1 \times 0.3 = 0.15$.
- Transaction 1’s arrival will be modeled with an exponential distribution with parameter $\lambda = 0.15$. 

<table>
<thead>
<tr>
<th>Scenario 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td><strong>Arrival Rate</strong></td>
</tr>
<tr>
<td><strong>No. of Users</strong></td>
</tr>
<tr>
<td><strong>Transaction 1</strong></td>
</tr>
<tr>
<td><strong>Type Name</strong></td>
</tr>
<tr>
<td><strong>Probability</strong></td>
</tr>
<tr>
<td>Scenario 1</td>
</tr>
</tbody>
</table>
Packet Generation

• Once a transaction arrives, the frames are generated in the following manner.
• The think time gives the average amount of time it takes to completely generate the transaction.
• The frames arrive with an inter-arrival distribution modeled as an exponential random variable with parameter $\lambda = \frac{(\text{Number of Frames} / \text{Think Time})}{\text{Number of Frames}}$. 

<table>
<thead>
<tr>
<th>SS7 Payload</th>
<th>Data Size</th>
<th>Num Of Frames</th>
<th>Think Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>50000</td>
<td>10</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Voice Over IP

TCP/IP Protocol

<table>
<thead>
<tr>
<th>Layer</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Flow control and data integrity</td>
</tr>
<tr>
<td>Transport (TCP)</td>
<td>Management of the end to end flow of traffic</td>
</tr>
<tr>
<td>Network (IP)</td>
<td>Management of the link between nodes</td>
</tr>
<tr>
<td>Data Link</td>
<td>Encoding to 1’s and 0’s</td>
</tr>
<tr>
<td>Physical</td>
<td></td>
</tr>
</tbody>
</table>

Challenges

- Bandwidth (latency and quality of voice or video)
- Locating the party being called (directory service)
- Call signaling (initialization and termination of a call)
- Interaction with telephony network
Voice Over IP
Implementation Elements

- Terminals, Gatekeeper, Gateway, and Multipoint Control Unit (MCU).
- Gatekeeper performs Admissions control, Bandwidth allocation, and Call control.
- Gateway performs as a standard telephone device in both world of telephony and data network.
  - VOIP Gateway connects to the voice network (using a standard telephony connection media such as coaxial cable or ISDN line) to map the dial number to appropriate IP Gateway using Directory Function.
  - VOIP Gateway connects to the IP network to convert the regular voice traffic to highly compressed digital format.
- MCU used as bridge to provide conferencing capability between multi parties.
Voice Over IP
Latency Solution

• Prioritizing Traffic using priority Queuing logic
• Shaping Traffic using Leaky Bucket Algorithm (Token Rate R, Bucket Depth B) in front of queue
• Fragmenting the data packets to same size as voice packets.
Voice Over IP
Call Signaling Example

1. DNS lookup
2. Newton IP address
3. INVITE Shahan@ voicegate.edu
4. Voicegate.edu
5. Shahan@ eng
6. INVITE shahan@ eng
7. Ok response
8. Ok response
9. ACK shahan@ voicegate.edu
10. ACK shahan@ eng
Voice Over IP
Call Signaling Example

1. George clicks on Shahan name in his e-mail list and clicks Call
2. George’s UAC (User Agent Client) does a DNS lookup and find Shahan’s SIP proxy server, Newton, of the domain voicegate.edu
3. George’s UAC sends an INVITE request to Newton
4. Newton looks up Shahan in the Location Server for a precise host
5. The Location Server returns Shahan@eng
6. Newton forwards the INVITE request to Shahan@eng
7. The UAS at Shahan’s PC pops a message on screen: George is calling, do you want to accept the call? Shahan clicks yes, and the UAS responds with a OK response to Newton
8. Netwon forwards the OK response to George@ISR.edu
9. The UAC at George’
Definition of Performance Measures

Response Time

- Response time for each transaction type is accounted for separately.
- Response time is computed at the sink as the job finishes.

\[ RT = \text{Finish Time} - \text{Birth Time} \]

- The average RT is equal to the average RT over all the jobs of a transaction type.

Queue Length (QL)

- Queue Length is measured for each payload node.
- Queue length is the number of jobs in the system at any given time.

\[ QL[t] = \# \text{Jobs in System}[t] = \# \text{Jobs Finished}[t] - \# \text{Jobs Started}[t] \]

- Average QL defined as time average of \( QL[t] \)

\[ QL = \frac{1}{n} \sum_{t=t_1}^{t_n} QL[t] \]
Definition of Performance Measures (continued)

Utilization

• Measured for each payload node.
• Defined as the ratio of the busy time to the total time available.

\[ Utilization = \frac{Busy \ Time}{Total \ Time} \]

• For N channels, utilization is the average utilization of the N channels.

\[ Utilization = \frac{1}{N} \sum_{i=1}^{N} Utilization_i \]

Here Utilization\(_i\) means the utilization of channel \(_i\).

Throughput

• Throughput is measured for each payload node.
• It is defined as the number of jobs finished per second.

\[ Throughput = \frac{\# \ Jobs \ Finished}{Total \ Elapsed \ Time} \]

• Here \# Jobs Finished can be a fraction based on how much of a job is done.
Object Structure for Data Export / Performance Parameters

Simulation
  \( \rightarrow \)

Workload
  \( \rightarrow \)

Scenario
  \( \rightarrow \)

Transaction
  \( \rightarrow \)

Network
  \( \rightarrow \)

Links
  \( \rightarrow \)

Nodes
  \( \rightarrow \)

Sinks
  \( \rightarrow \)

Names of Scenarios

• Arrival Rate
• Number of Users

• Probability

• Weight
• Utilization

• Queue Length
• Throughput
• Utilization
• Other measures depending on type

• Number of Packets
• Average Packet Delay
• Other measures depending on type

SS7 Transaction

Payload Transaction

• # Packets Arrived
• Transactions Setup
• Transactions Terminated
• Average Packet Delay
• Setup Time
• Termination Time
• Data Size
• Number of Frames
• Think Time

• # Packets Arrived
• # Transactions Completed
• Average Packet Delay
• Average Response Time
• Data Size
• Number of Frames
• Think Time

Sink Records

Other measures depending on type
Graphing Capability Configuration & Data Flow

Simulation
(Data stored in cache)

File

getExportString()

Export Function

File

String Parser

Gui.Graph.StringParser
StringParser(File)
StringParser(String)

· Reads data from a file or String and extracts the networks components, their respective measurements and corresponding data.
· Saves this data in a format that other classes can use.

Graph Wizard

Gui.Graph.GraphWizardGUI
GraphWizardGUI(StringParser)

· Takes data from the StringParser and allows the user to choose what type of graph to make and what data to use for axes and measurements.
· Saves this data in a GraphModel.

Graph Viewer

Gui.Graph.GraphDialog
GraphDialog(GraphModel, typeOfGraph)

· Displays the data in a GraphModel in various formats, such as Line Graph, Bar Graph and Pie Chart.
Simulation Scenario
Sample Output Graph

Utilization vs Number of Users

Style

Utilization

Number of Users

Scenario 1
Scenario 2
Scenario 3
Scenario 4
Scenario 5
### Ingenuity / Opnet Comparison

<table>
<thead>
<tr>
<th>Ingenuity</th>
<th>Opnet</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Internet/Intranet-based computing capability (collaboration, accessible)</td>
<td>• Not Available</td>
</tr>
<tr>
<td>• Java client/server object-oriented design and programming</td>
<td>• C object-oriented programming, and it supports C++</td>
</tr>
<tr>
<td>• Platform-independent operational environment</td>
<td>• WS/Unix environment, and it supports Window</td>
</tr>
<tr>
<td>• Easy to learn and use (short learning curve)</td>
<td>• Longer learning curve</td>
</tr>
<tr>
<td>• Menu-driven workload model provides better understanding, organization and quantification of input parameters</td>
<td>• Menu-driven and flat file (suitable for experienced analysts)</td>
</tr>
</tbody>
</table>
## Ingenuity / Opnet Comparison (cont.)

### Ingenuity

- Object design for call signaling and workload flow (call models)
- Simple and customized routing
- Provides flexible programming environment to accommodate the modeling of wide range of complex networks
- Limited list of queuing logics and performance indices due to its infancy
- Execution time is limited by the Java development environment and by the event based simulation engine
- Output/Export capabilities

### Opnet

- State diagrams
- Simple and customized routing
- Complex/packaged design makes it inefficient to accommodate the modeling of wide range of complex networks
- Longer list of queuing logics and performance indices (due to many years of continuous development)
- Execution time is limited by the GUI capability and by the event based simulation engine
- Output/Export capabilities
### Ingenuity / Opnet Comparison (cont.)

<table>
<thead>
<tr>
<th><strong>Ingenuity</strong></th>
<th><strong>Opnet</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Complete ownership and control of source code, which provides 100% simulation advancement capability</td>
<td>• Licensed Runtime:</td>
</tr>
<tr>
<td>• Geared toward the evolution of network signaling and payload. It addresses network capacity planning and management, and quantifies multimedia workload / applications sizing, hosting, and warehousing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Rigid frame work (simulation engine, library functions, etc) can lead to adaptability and scalability problems</td>
</tr>
<tr>
<td></td>
<td>• General purpose network simulation environment, suitable for low level conceptual network applications and hardware design</td>
</tr>
</tbody>
</table>
Simulation Demonstration

- Simulation Demonstration
Statement of Work
Extension of INGENUITY

• 1.0 Telephony Optimization
  – Extend current INGENUITY software capabilities to meet and to further simulate NSA specific applications, and to analyze NSA evolving telephony network requirements
  – Demonstrate INGENUITY by applying hypothetical intelligent networks specified by NSA such as ArbiNet (documented by NSA)

• 2.0 Simulation Extension
  – Extend INGENUITY to allow simulation of various network types implementing the following technologies:
    – ATM and IP based data networks (including quality of service)
    – Broadband ISDN Access and Backbone networking (SS7 based)
    – User and terminal mobility (digital cellular, PCS, wireless IP)
    – User, Application, and Network Security
• **2.0 Simulation Extension (cont)**
  - Interworking Functions (interfacing network paradigms, such as ss7 and IP) and interoperability problems
  - Network Gateways
  - Link and Node Modeling application programming interface
  - Network management
  - Adaptive networking
  - Dynamic Routing
  - Data Warehousing
  - Software service hosting
3.0 Research
   - One of the essential objectives to develop INGenuity is to help analysts conduct research into specific network technology challenges. Initial research problems will include
     - Voice over IP Telephony Integration
     - Third Generation Cellular Implementations
     - Continuous workload based on analytical models
     - Hierarchical network modeling
     - Adaptive Networking, dynamic routing, data warehousing, and software service hosting

4.0 Training
   - UMD will develop and deliver training packages on the use of INGenuity as applied to each selected technology set. This will include an introductory course and at least three advanced courses through the duration of the project.
Code Distribution

Lines of Code by Package
24k total lines
References


• IEEE proceeding of the 7th International Symposium on MASCOTS’99

• Gamma, E., Helm, R., Johnson, R., Vlissides, J. 1995 Design Patterns. Addison-Wesley, Reading.

• Global Knowledge, Revision 3, 1999 Introduction to Voice Over IP


• Vlissides, J. 1998 *Pattern Hatching*. Addison-Wesley, Reading.


• Warmer, J., Kleppe A. 1999 *The Object Constraint Language*. Addison-Wesley, Reading.
