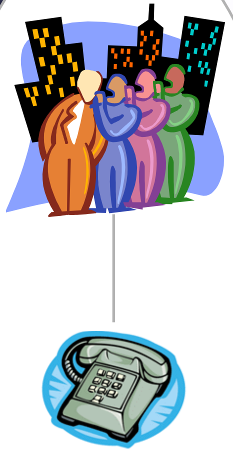
Reducing the Feature Interaction Problem Using an Agent-Based Architecture

Debbie Pinard
Aphrodite Telecom Research,
a Division of Pika Technologies
debbie.pinard@pikatech.com

Introduction

- What are the inherent problems in current PBX architectures?
- Why is this architecture better?
  - Agent-based
  - Data driven
  - Handling of features

The Problem Space



Many people, one phone

Attendant Groups, Help Desks, ACD's

One person, many devices

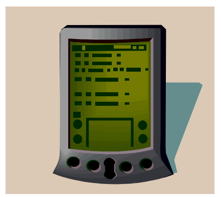
Auto Attendant, Voice Mail

Wireless

Sprint
AT&T
Bell

Different Carriers

FAX

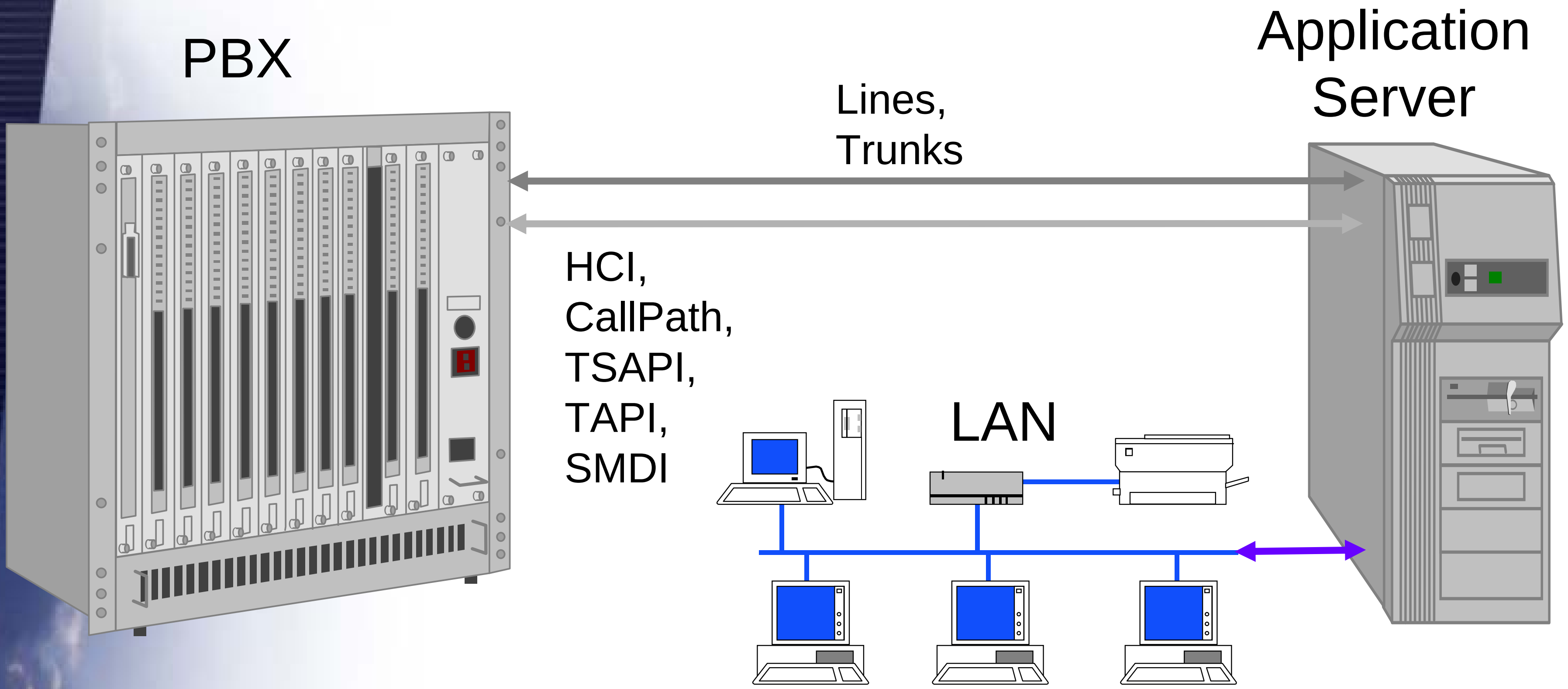


New Device Types

Boss/Secretary

PIKA TECHNOLOGIES INC. the right voice

First Generation Architecture

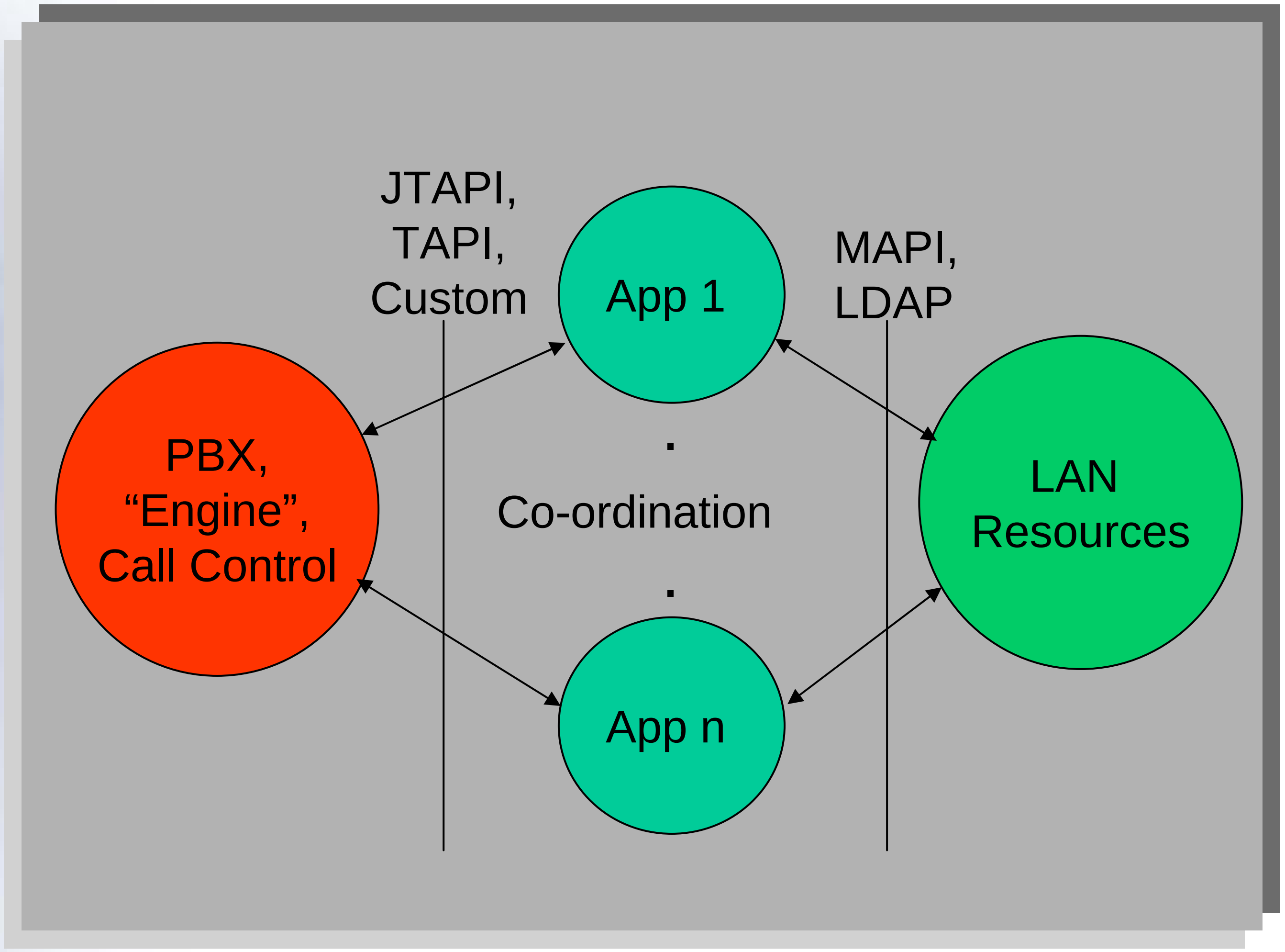


Problems:
- •Device based architecture
- •Large volume of spaghetti code
- •New features “barnacled” on

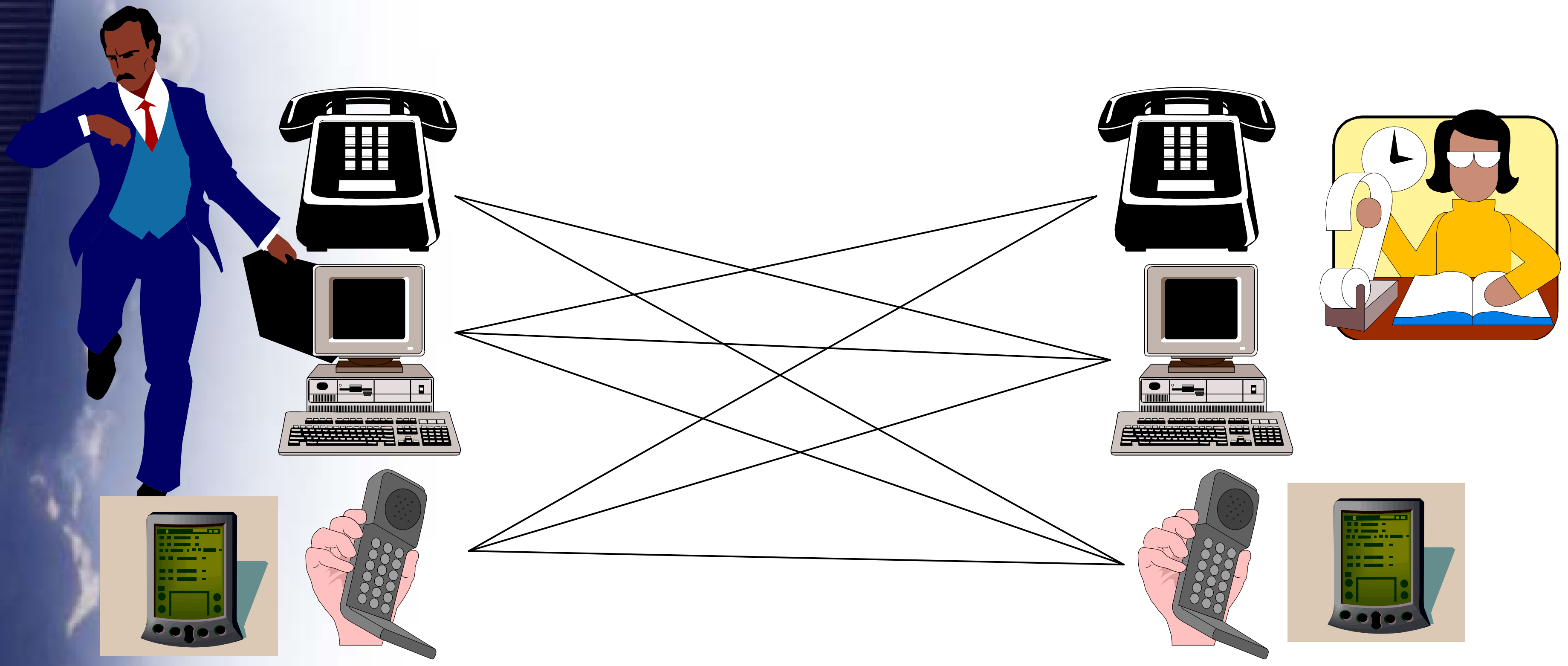
Problems:
- •integration complicated
- •guessing at state of devices
- •glares
- •lack of control over system

the right voice PIKA TECHNOLOGIES INC.

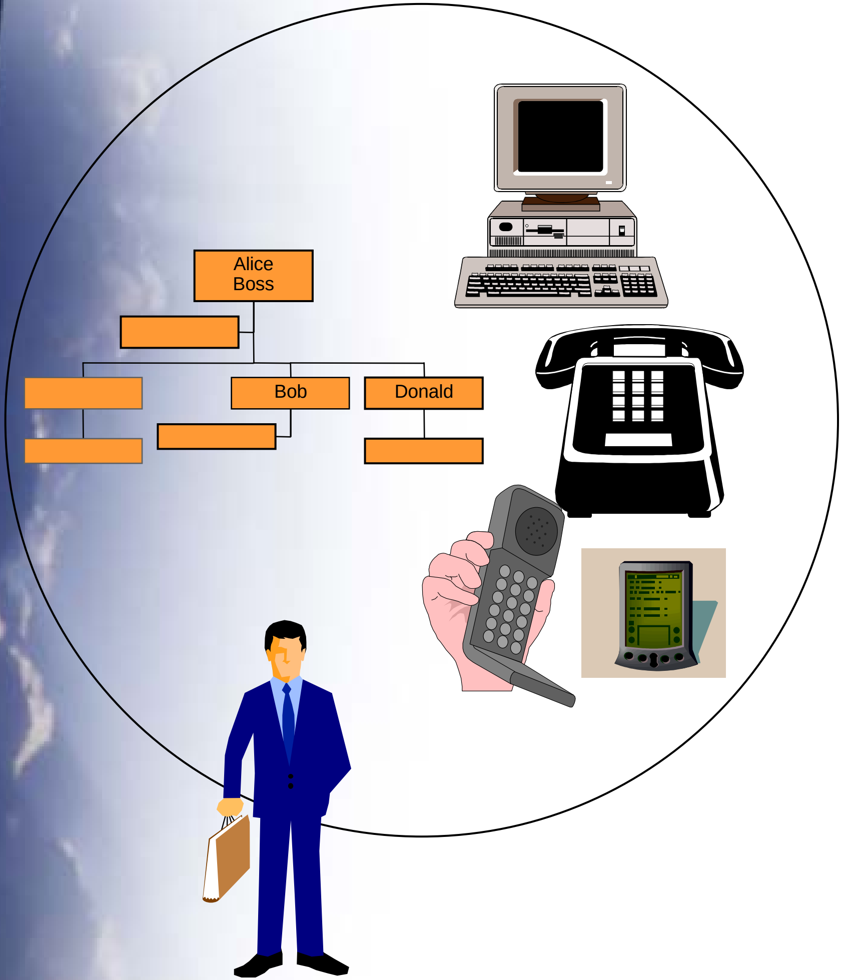
Second Generation Architecture

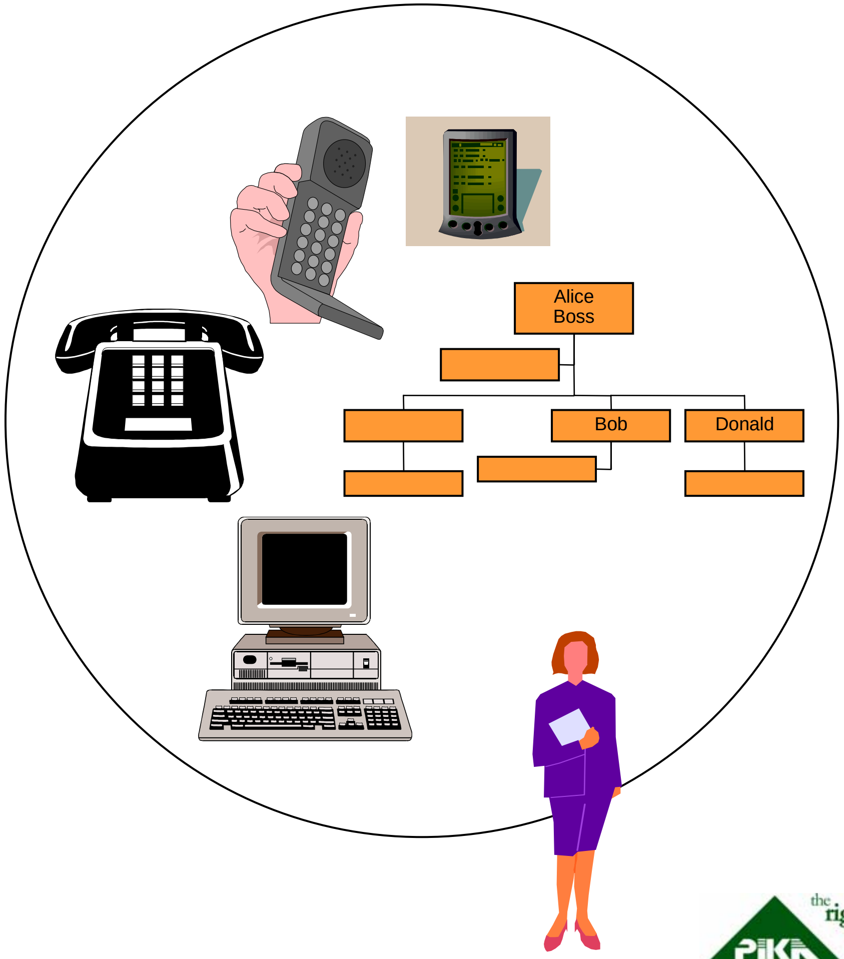


Communication: The Present



The Reality





Agents, Agencies and Organizational Design

Introduction

- Mid 1990’s
- Step past *Object* Oriented design into *Agent* Oriented design
- Agents are autonomous or semi-autonomous software systems that perform tasks
- Autonomy means there is no centralized control
- An *agency* consists of a group of agents which take specific roles within an organizational structure
- The group is more than the sum of the capabilities of its members

Agent Oriented Approach to a Communication System

- Completely Different Way of Thinking!
  - not device based
  - no individual applications
- Communication based on organizational policies
- Each Resource is represented by an agent

the right voice PIKA PIKA TECHNOLOGIES INC.

Organizational Design

- What does an organization consist of?
  - Job Descriptions
  - Resources (People and Devices)
  - Procedures
  - Policies
  - Information

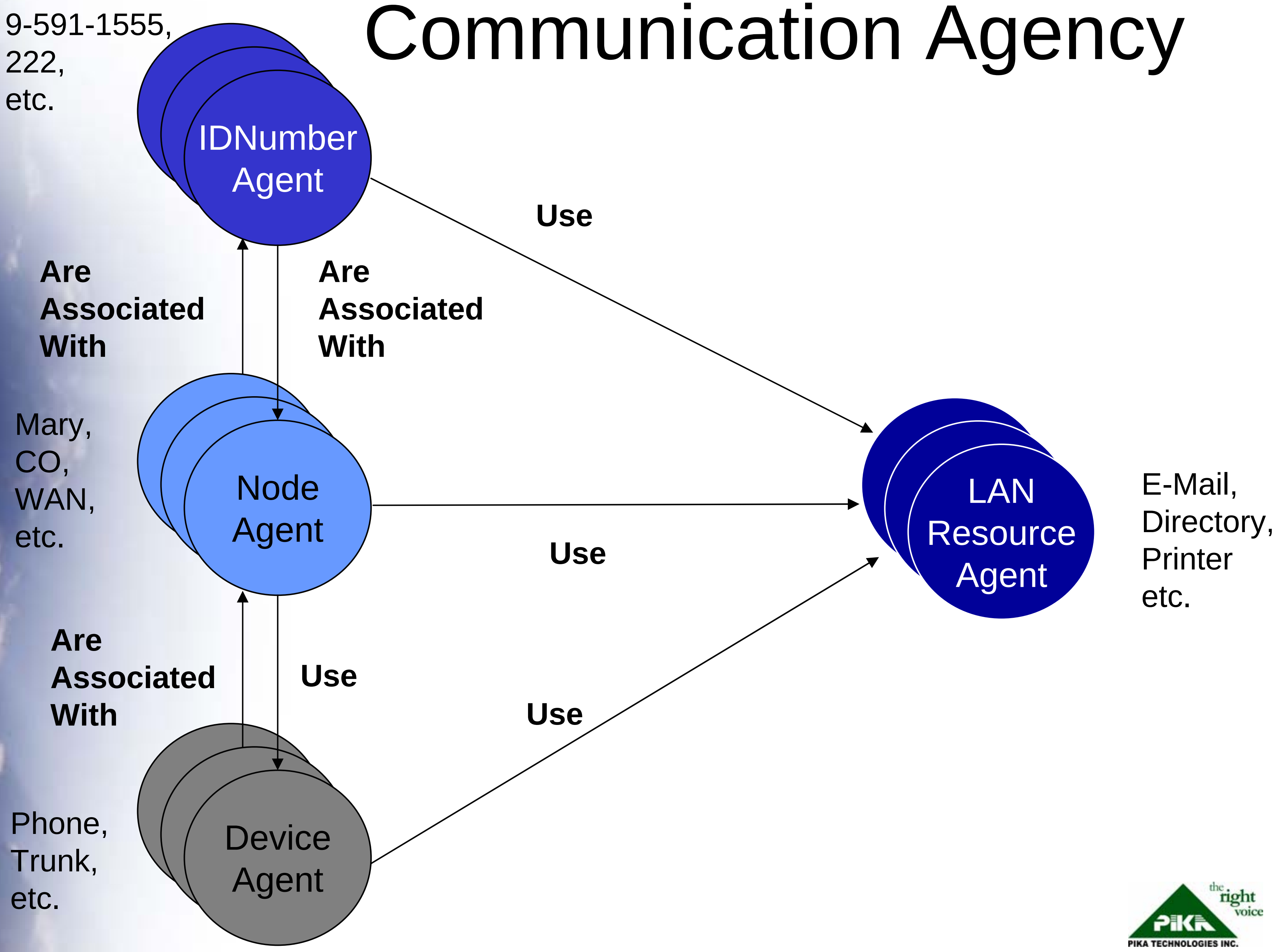
What are the Resources?

How Can Resources Be Grouped?

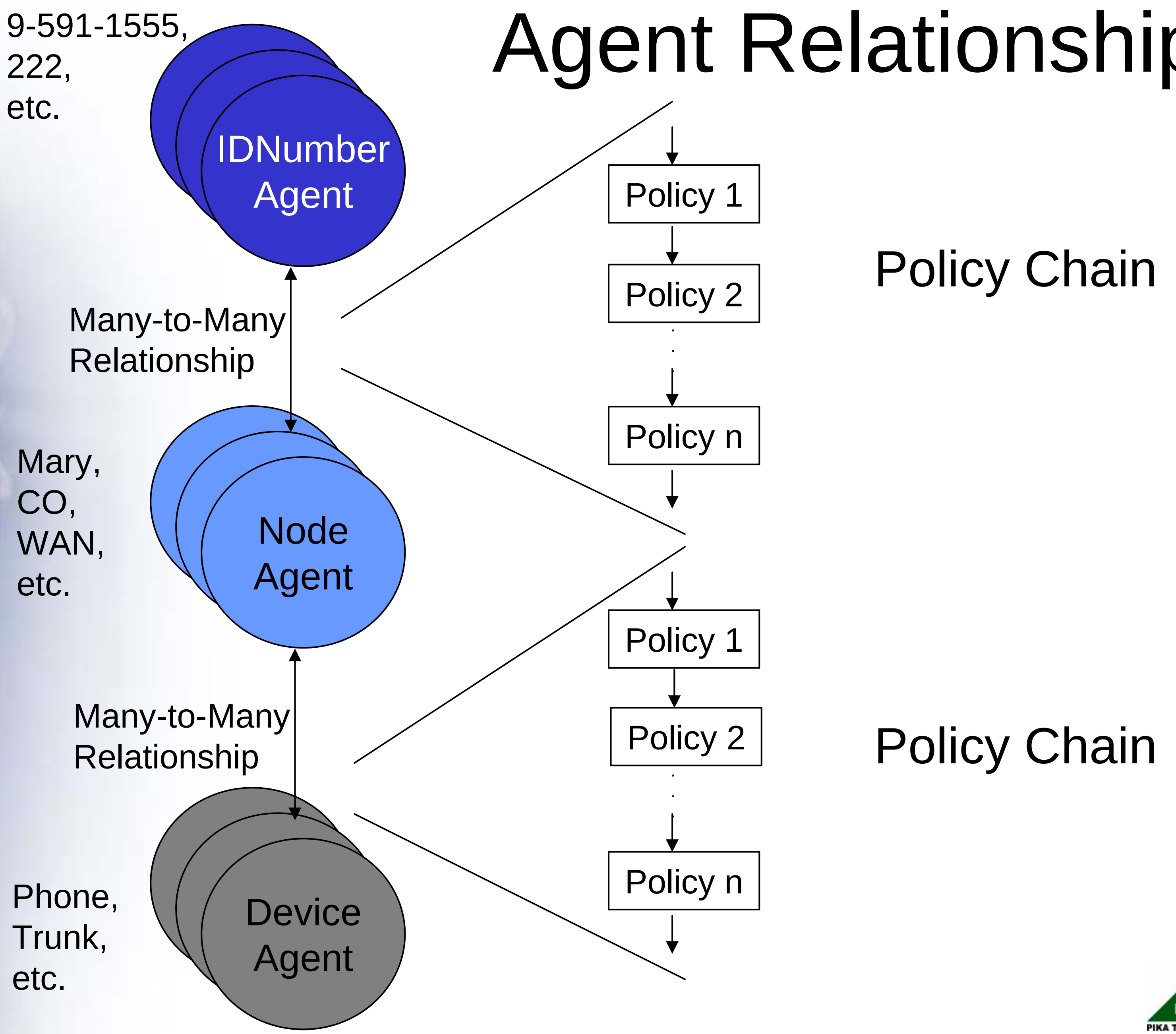
- LAN Resources
  - E-Mail
  - Corporate Directory
  - Corporate Printer/Fax
  - Database
- IDNumbers (Roles)
  - External #s
  - Internal #s
- Nodes
  - CO
  - WAN
  - Known Users
  - Unknown Users
  - Objects (Doors, Lights, etc.)
- Devices
  - DSPs
  - Trunks
  - PCs, Personal Printer
  - Phones
  - Sensors
  - Switches
  - PDAs

PIKA TECHNOLOGIES INC.

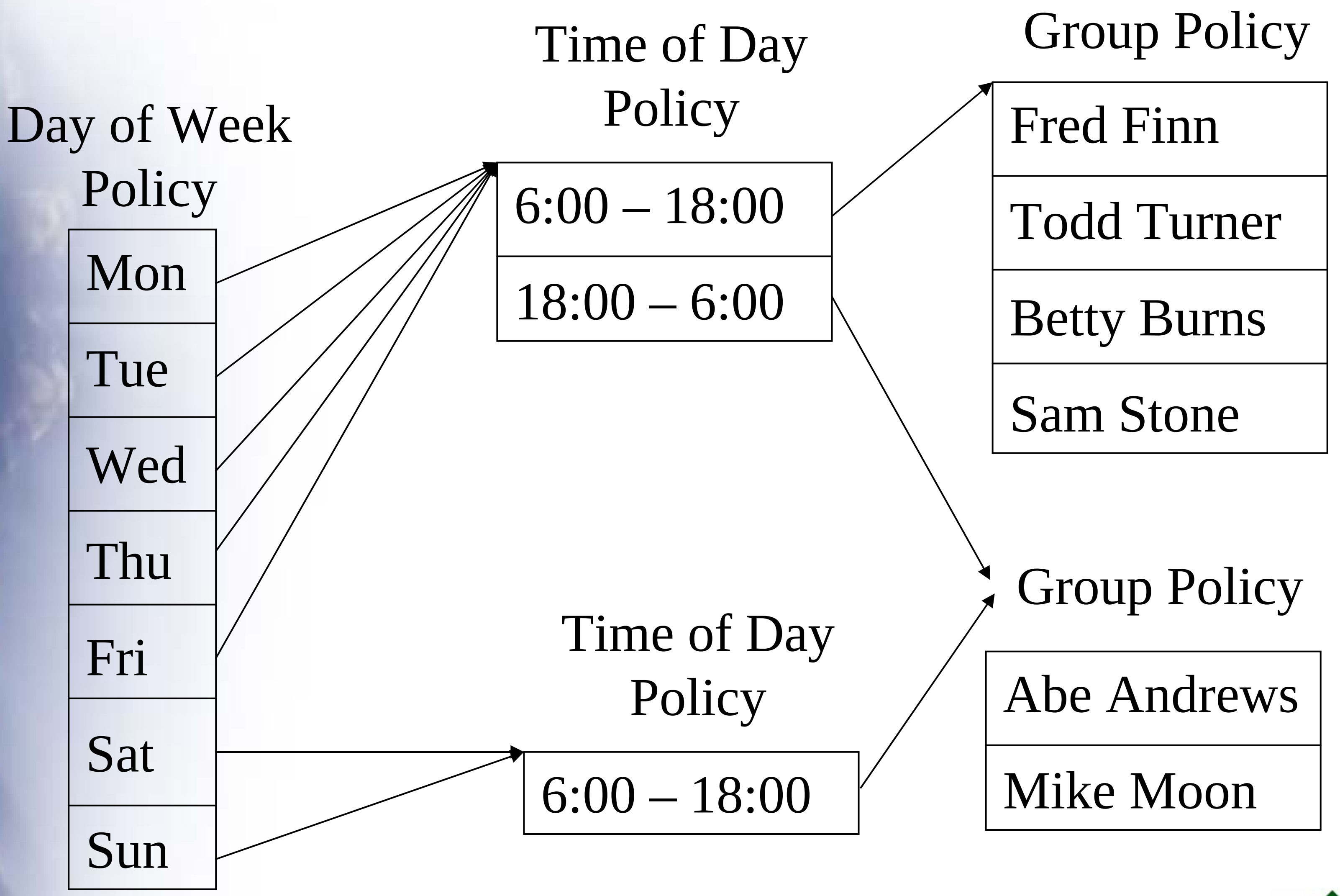
Communication Agency



Agent Relationships



Example of a Policy Chain



Feature Design

Feature Types

Two types:

- Standard (e.g. camp on, call forwarding, transfer, etc.)
- Feedback (e.g. IVR, auto attendant, voice mail, etc.)

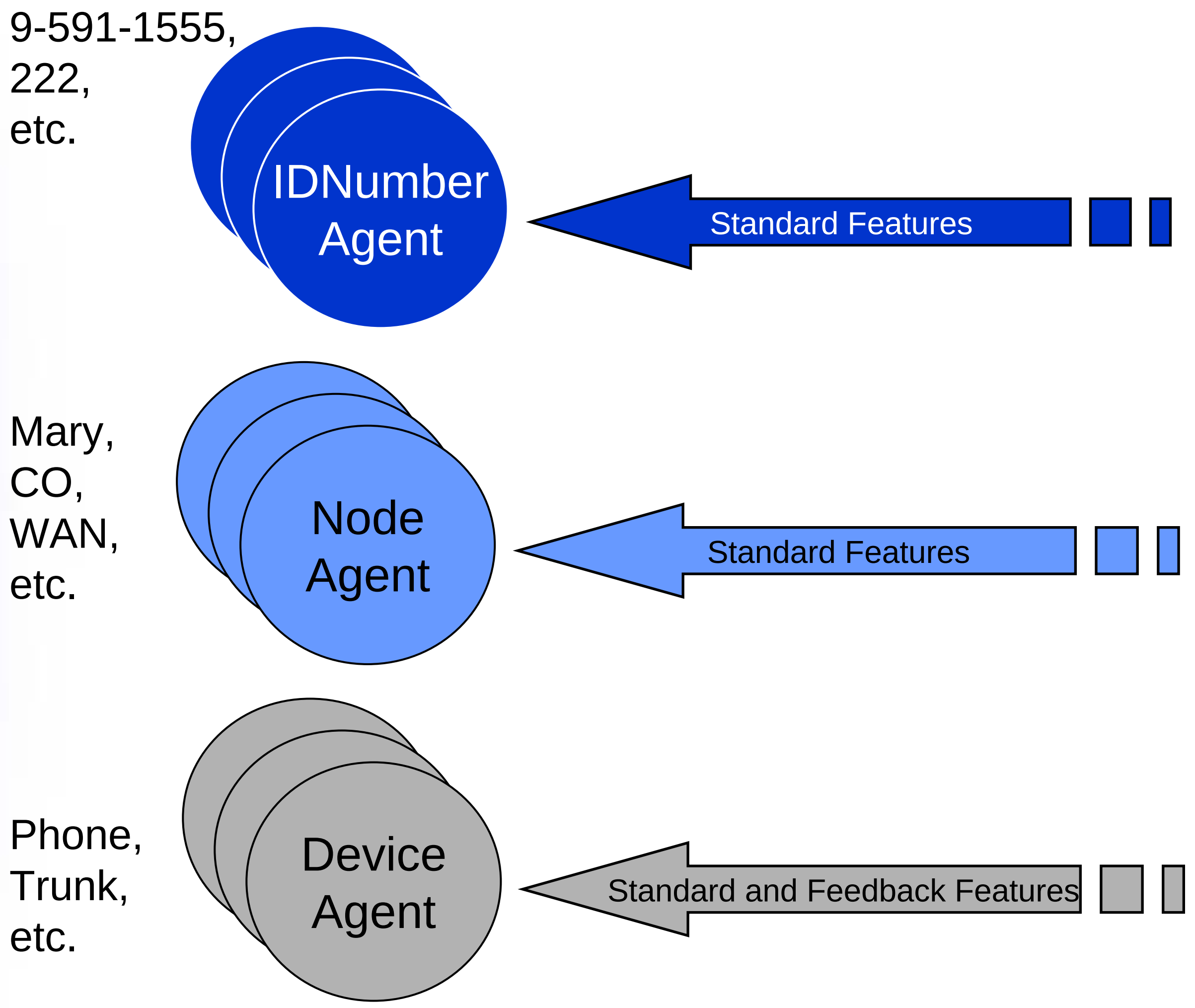
Standard Features

- Temporarily takes over from the basic call flow
- Can be invoked by events (‘off hook’) or access codes (*56)
- Separate from basic call control, managed by a feature manager
- Uses a state/event trigger table, pointing to a policy chain, terminating on a feature object
- Common features are broken into ‘mini’ features (e.g. transfer)

Feedback Features

- Provides options or information to a person instead of giving them a tone
- Uses a tone/reason trigger table, pointing to a policy chain, terminating on an initial Feedback Feature object
- Feedback Feature is made up of a tree of linked objects
- Objects can use DSP resources (like play, record, text-to-speech, etc.), can collect digits, can retrieve data, can invoke a feature, etc.
- Once an object is implemented, it is available to any Feedback Feature

Trigger Tables



Data Driven Design

How is it Done?

- A record in a table represents each agent
- Linked policy records represent the relationship between agents, as well as the path between a feature trigger and a feature
- Features are triggered based on data in a trigger table record
- Feedback Features are made up of a tree of linked records
- The running system is built from the database, each record is represented by an object, some of which are linked together

Why is it Important?

- Call flow is determined by a system administrator
- Users can also be given the ability to change data
- Each running system is unique, and tailored to an organizations preferences and policies
- Feedback to users can be programmed to give more information rather than just a tone
- It is much more open to adding some form of AI in to change data based on different criteria

The ‘Feature Interaction’ Problem: How the Architecture Helps

Conflicting Goals

Different features triggered by same state/event

- Trigger table allows for ordering of features
- Example: conference and swap
- Can also be handled by introducing a feedback feature and letting the caller choose
- Example: call wait and voice mail
- Implementation eliminates interaction
- Example: call forward busy and call wait

Type of Call

Feature invoked based on type of call

- Example: call forwarding vs. hunt group call
- *Eliminates* this form of feature interaction, since the type of call is inherent in the path it is following

Competition for Resources

Specific set of resources available

- Co-ordination needed to partition resources among applications
- Needs to take into account enterprise and group policies
- Example: 911 trunk access
- New hunt policy which takes over a trunk or line if the group is busy, based on the call path

Changing Assumptions on Services

What was true in the past can change with the advent of new technology and services

- Example: What does ‘Busy’ mean?
- A Device agent can specify what busy means for different devices
- A Node agent representing a person can specify what busy means for that person, regardless of the device used

Policy Replacing Many Features

Implementing one data-driven policy replaces many features

- Example: call restriction, toll control, interconnect rights, call screening
- Create a list of people, numbers, devices
- At each agent level, a policy can be invoked which restricts a call based on a list
- Can let the call through or block it
- Note: Tom’s availability

Policy Available to All Call Types

Call type had a feature that was only available to it

- Example: toll control on trunks
- Chaining of a day of week policy followed by a time of day policy between IDNumber and Node agents and Node agents and Device agents provides ultimate flexibility to all types of calls with no extra code

Reverting to Basic Call ASAP

To avoid some feature interactions, features terminate as quickly as possible and revert back to a basic call state

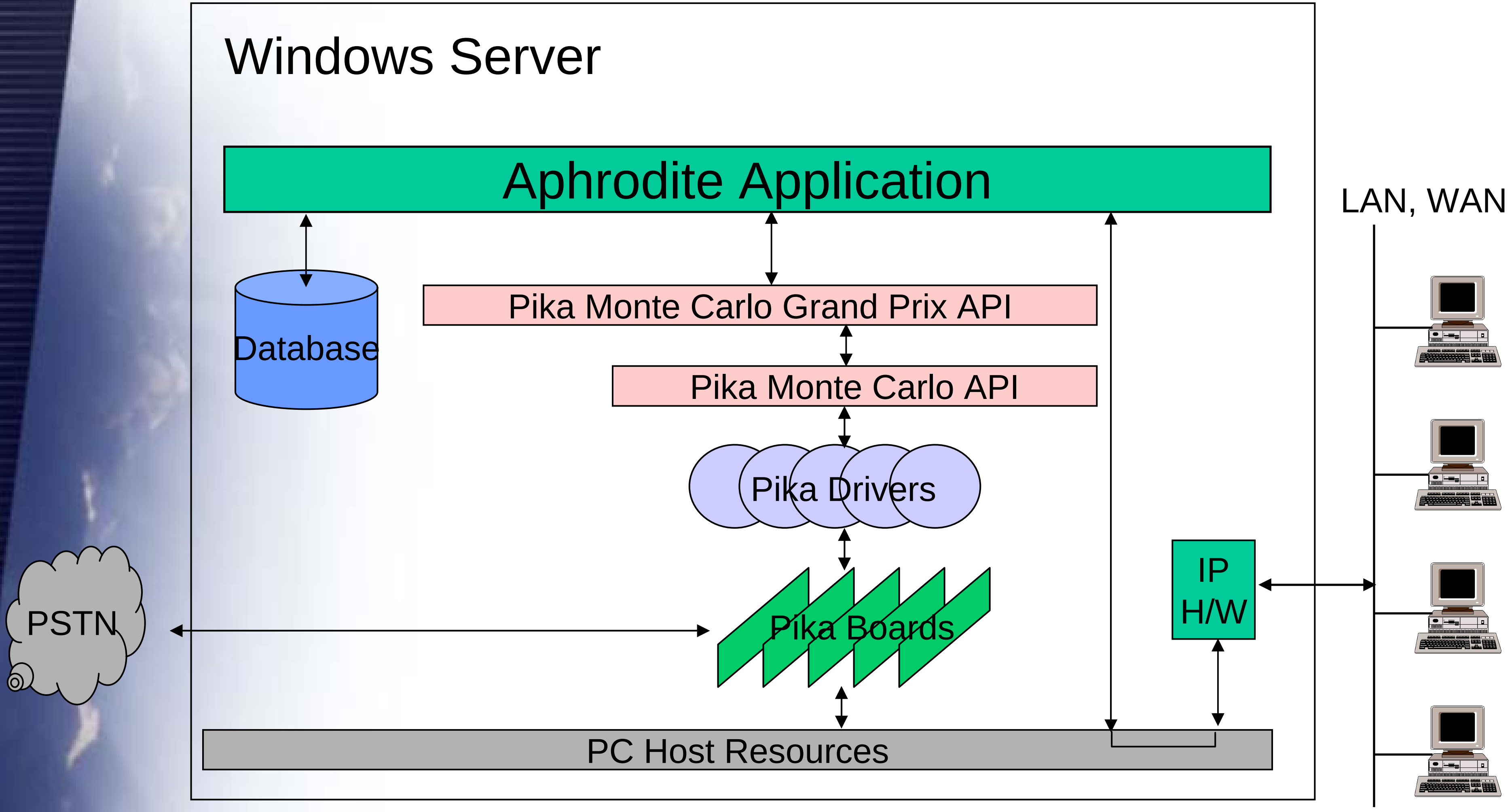
- Example: call hold, queuing
- Held or queued person reverts back to the Waiting for Termination basic call state, not a ‘special’ state
- All features triggered off of events in the waiting for termination state are still available

Adding New Device Types

- Just needs a new device agent which manages the device interface, but talks to the node agent in exactly the same way as any other device
- Example: I/O port device
- Opening a door can make a phone call, or making a phone call can turn on a light
- Uses the same trigger table as all other devices
- Can be part of a group

Implementation Details

Hardware/Middleware



Software

- Written in VB using the Pika APIs
- ~ 60 Standard Features, ~20 Feedback Features
- Only 32,400 lines of code
- Ratios:
  - Forms 44%
  - Basic Call 32% (includes hunting)
  - Standard Features 12%
  - Feedback Features 12%
- Supports analog and digital trunks, POTs and I/O ports

Single Threading

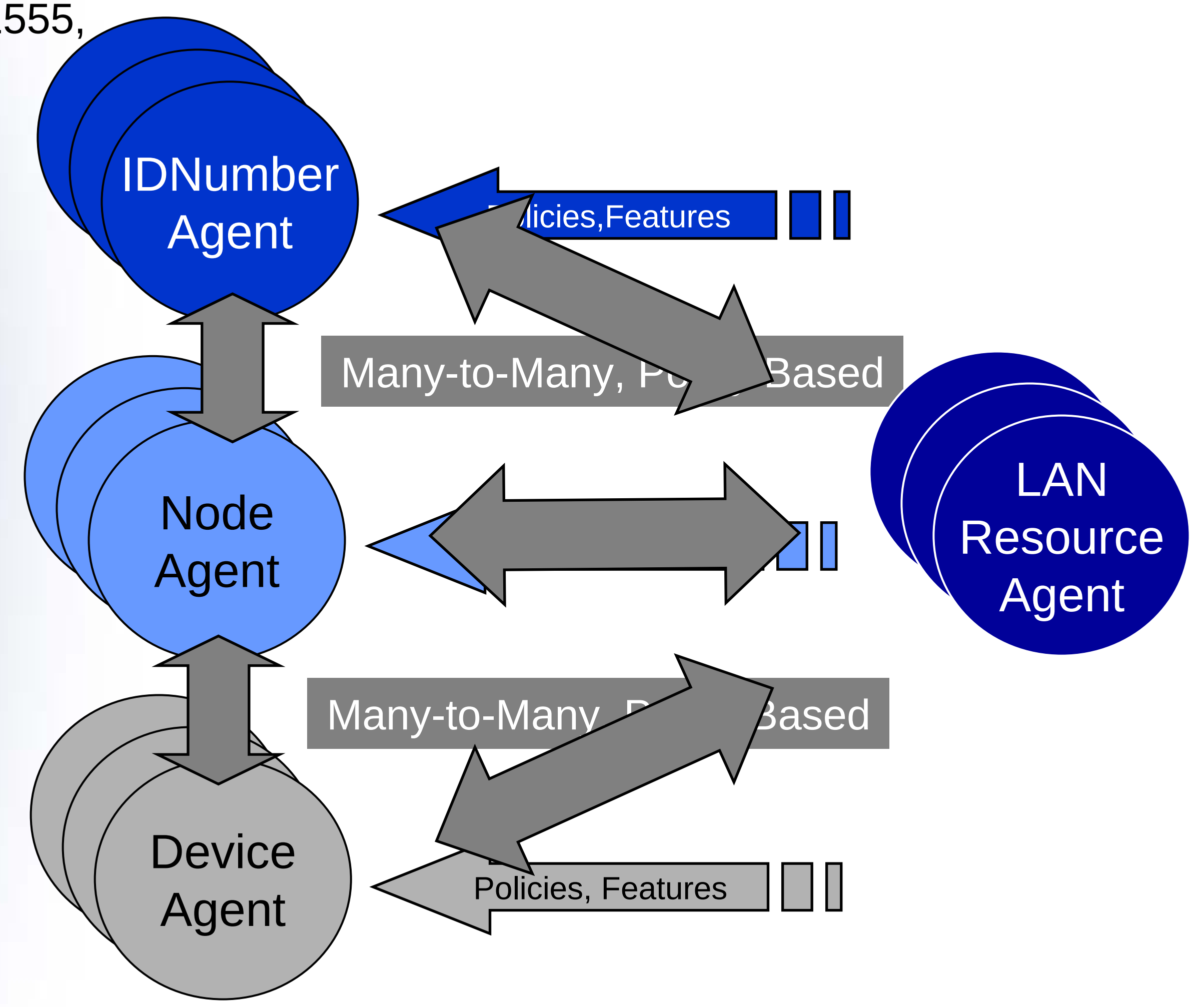
- Cuts down on feature interactions and glare situations
- Only one feature can be active at a time, and runs to completion
- Very few features need more than one event
- Those that do only involve one device
- Multi-threading is done where it makes sense, in the drivers and middleware
- All timing is done in the middleware

Summary

9-591-1555,
222,
etc.

Mary,
CO,
WAN,
etc.

Phone,
Trunk,
etc.



E-Mail,
Directory,
Printer
etc.

Conclusion

The Aphrodite platform proves that a data driven, Agent-based architecture is a superior approach to developing communication systems, and can reduce or even eliminate some forms of feature interactions.

Questions?