p. 18

PROGRAMMING FAULT-TOLERANT DISTRIBUTED SYSTEMS IN Ada

Susan J. Voigt

Computer Science And Applications Branch
Langley Research Center

For Presentation At
The Computer Science/Data Systems Technical Symposium
Leesburg, Virginia

April 18, 1985

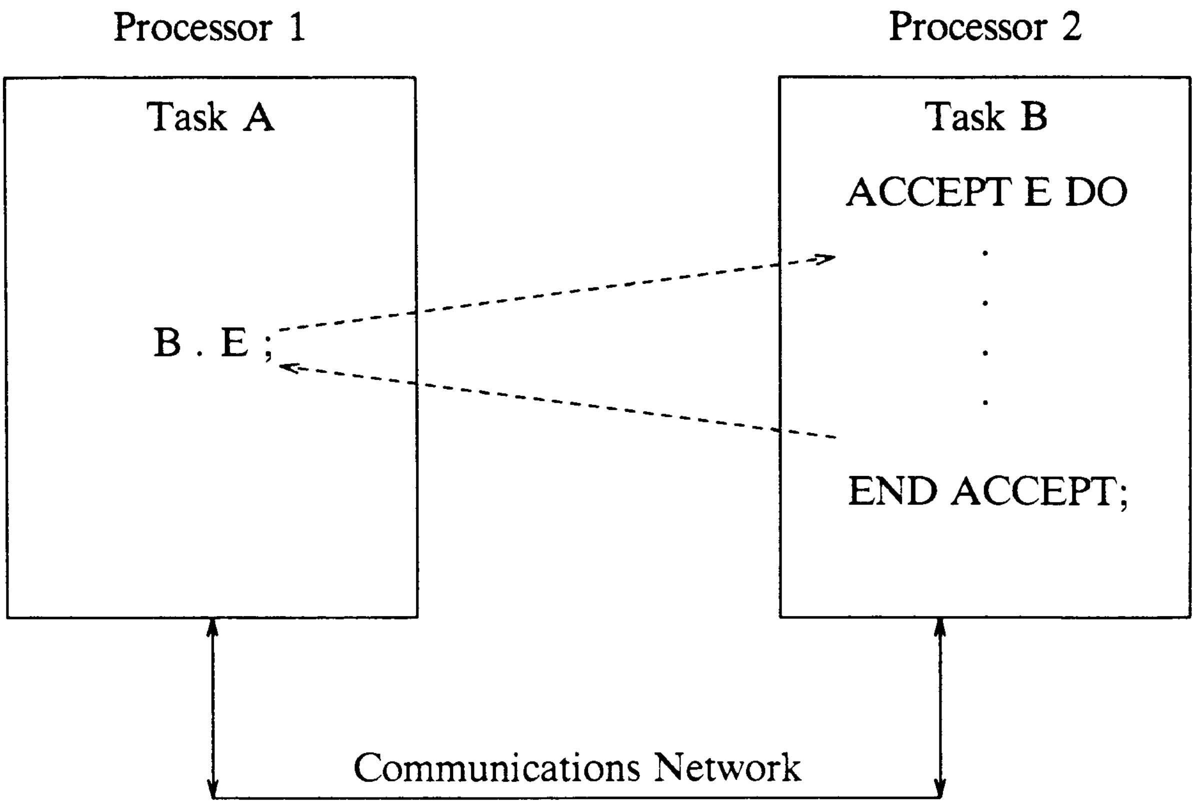
PROJECT GOALS

- Examine Use And Implementation Of Ada On Distributed Systems
- Programming Of Systems With "Fail-Stop" Components
- Analyze Tolerance To Loss Of Processors
- Propose Solutions To Language Inadequacies
- Implement These Solutions
- Perform Validation Experiments
- Suggest Long-Term Changes To Ada

WHY Ada ON DISTRIBUTED SYSTEMS?

- Ada Will Be Used Extensively In Embedded Systems
- Embedded Systems Will Be Distributed
- Many Applications Will Be Crucial
  - Spacecraft Systems
  - Aircraft Systems
  - Industrial Process Control
- Distributed Systems Should Support Graceful Degradation
  - Partial Power Failure
  - Physical Damage
  - Component Failure
- Ada Permits Distributed Targets Explicitly

Ada RENDEZVOUS



- Task A Suspended If Processor 2 Fails During ACCEPT

Ada DIFFICULTIES

- Language Elements That Cause Difficulty:

  — All Forms Of Rendezvous

  -- Shared Variables

  — Task Elaboration And Termination

  — Loss Of Context

  — Distribution Control

  — Processor Loss - Detection And Signaling

- Complete Lack Of Distribution Semantics

- Complete Lack Of Failure Semantics

SOLUTIONS

- Short Term:
  - — Define Simple Distribution Semantics
    - What Can Be Distributed
    - Where It Can Be Distributed
    - Control Of This Distribution
    - Necessary Restrictions On Program Structure
  - — Define Failure Semantics As Equivalent To "ABORT"
- Long Term:
  - — Complete Redefinition Of Tasking Semantics
  - — Partial Replacement Of Tasking Syntax
  - — Language Support For Distributed Systems Using Fail-Stop Components

STATUS OF SOLUTIONS

- Language Review Complete
- Short Term Solution:
  - Distribution Semantics Complete
  - Failure Semantics Complete
  - Implementation Design Complete
  - Implementation Complete And Under Test
  - Realistic NASA Application Required For Evaluation Of All The Ideas
- Long Term:
  - Several Proposals Being Examined
  - Radical Changes To Ada Required
  - U.Va's Ada-2 Design Expected Soon

OTHER ISSUES IN DISTRIBUTED SYSTEMS

- Concurrency Implies Nondeterminism
- Difficulties With "What If" Questions Of Language Semantics
- Difficulties With "What If" Questions Of Fault-Tolerance Strategy
- Require "Complete" Demonstration Of Fault Tolerance
- Systematic, Repeatable Experiments

TESTBED REQUIREMENTS

- Model Arbitrary Physical Architectures
- Represent Any Logical Organization
- Provide Parallel Execution (The Illusion At Least)
- Control Interprocessor Communication
- Control Process Execution
- Fail And Restart Processors At Well-Defined Points
- Maintain Time Correctly
- Provide Monitoring Facilities

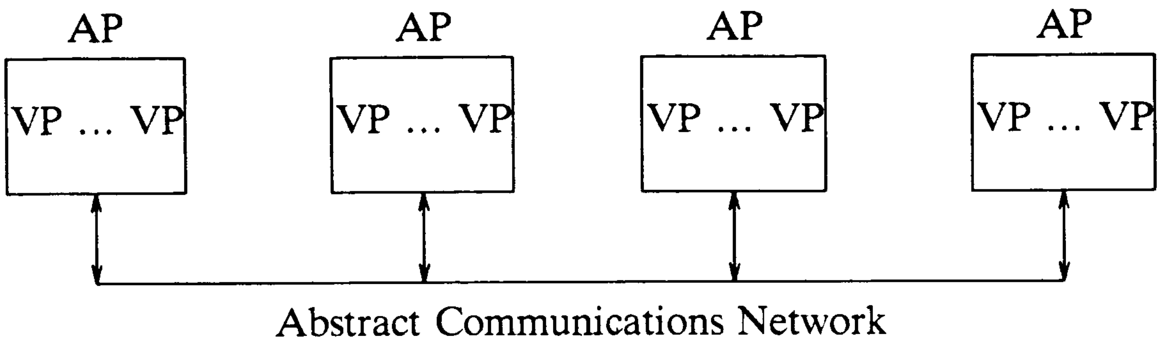
VIRTUAL PROCESSORS

- Key Component Of The Testbed
- Literally Ada "Virtual Machines"
- One Virtual Processor For Each Ada Task In A Program
- Designed To "Implement" Ada Tasking And Exception Handling
- Several Different "Memories" - Whatever Is Convenient
- "Hardware" Implemented Entry Queues
- "Rendezvous" And Similar Instructions
- Implemented By Simulation

ABSTRACT PROCESSORS

- Correspond To "Real" Equipment Required By Experimenter
- Each Abstract Processor Implements Any Number Of Virtual Processors
- Abstract Processors Are "Ideal" Also - E.G. Suspended, Failed, Etc
- Abstract Processors Communicate Via An Abstract Communications System
- Testbed Supports An Arbitrary Number Of Abstract Processors

EXPERIMENTAL STRUCTURE

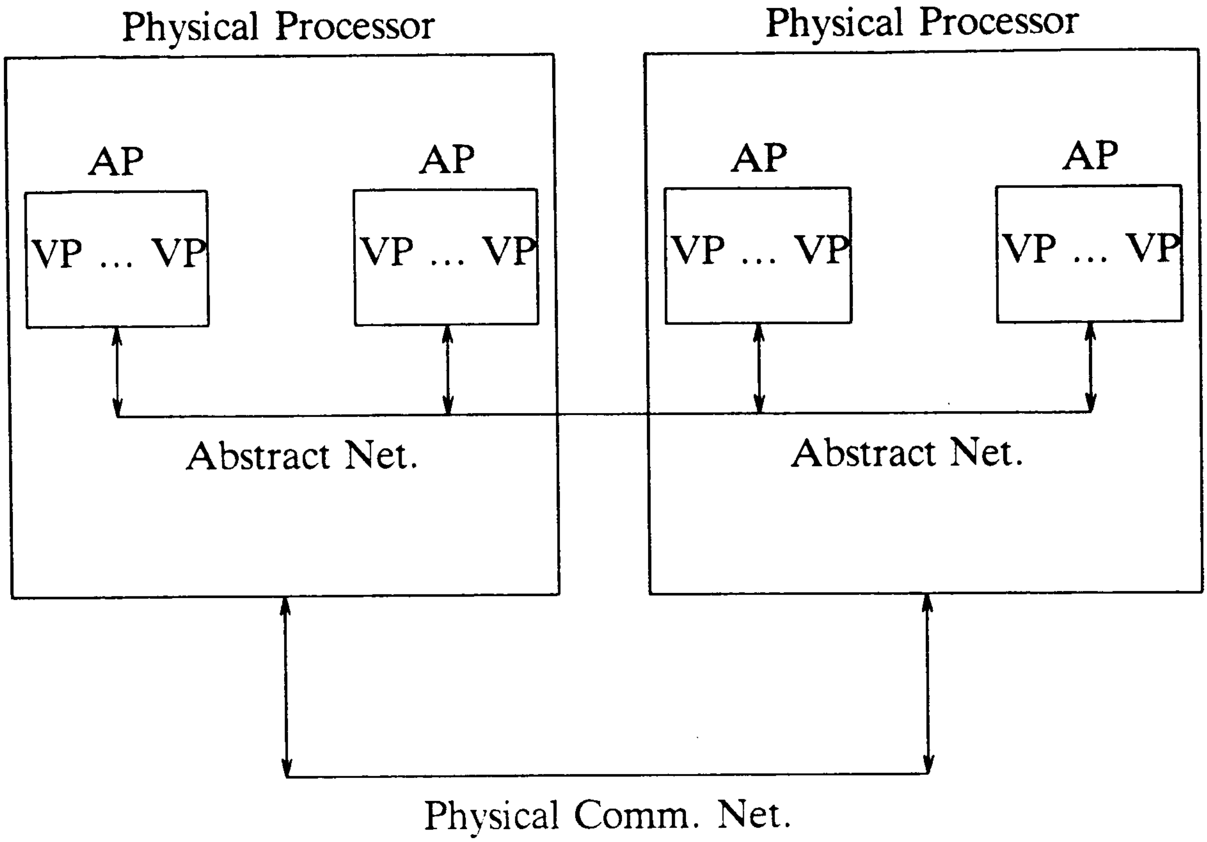


- View Seen By Experimenter
- AP's Represent Ultimate Target He Has In Mind

PHYSICAL PROCESSORS

- Correspond To "Actual" Equipment Available To Experimenter
- Each Physical Processor Implements Any Number Of Abstract Processors
- Physical Processors Are "Real"
- Physical Processors Communicate Via A "Real" Communications System
- Single Abstract Processor Can Run On Each Physical Processor

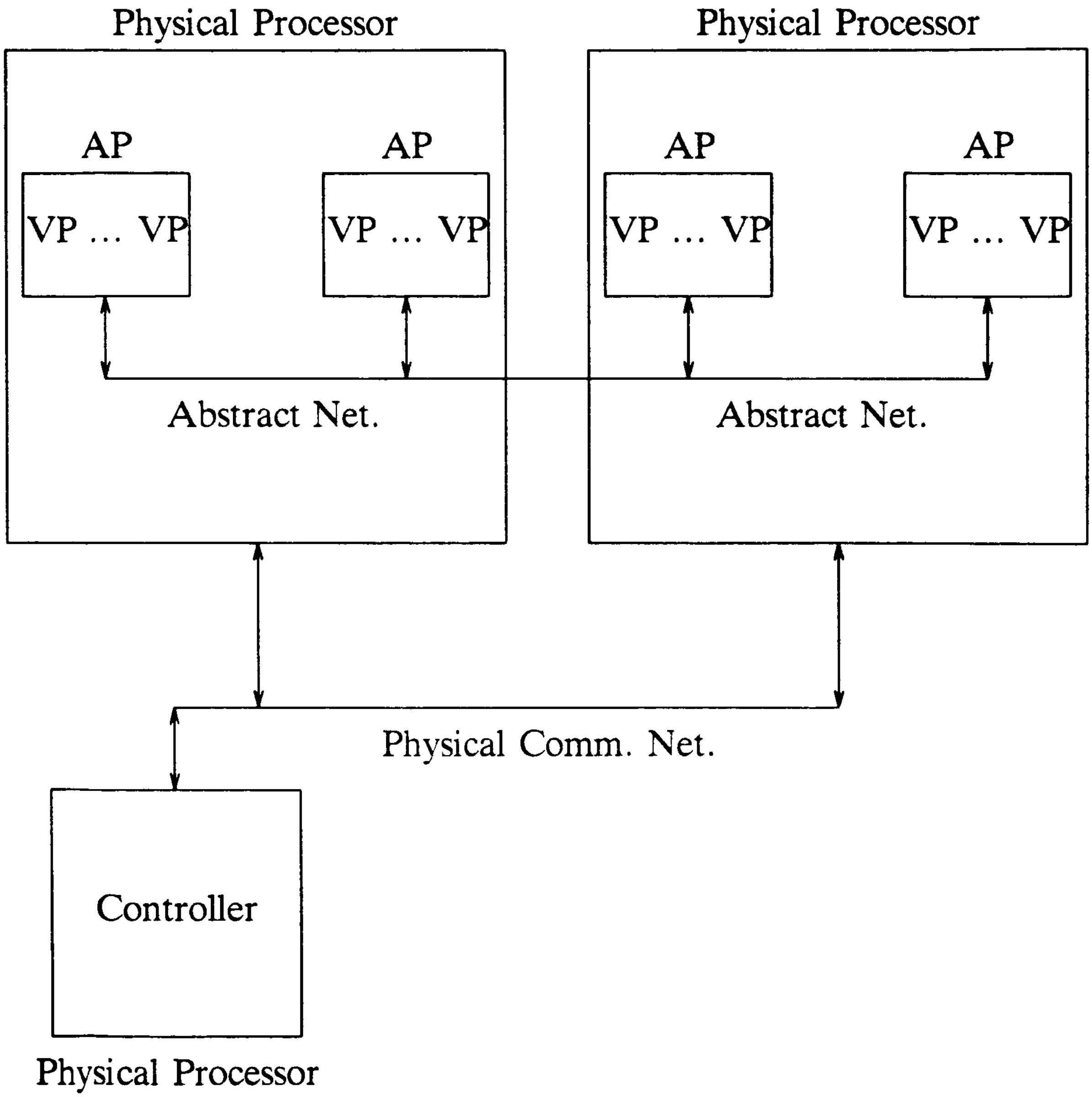
TESTBED STRUCTURE



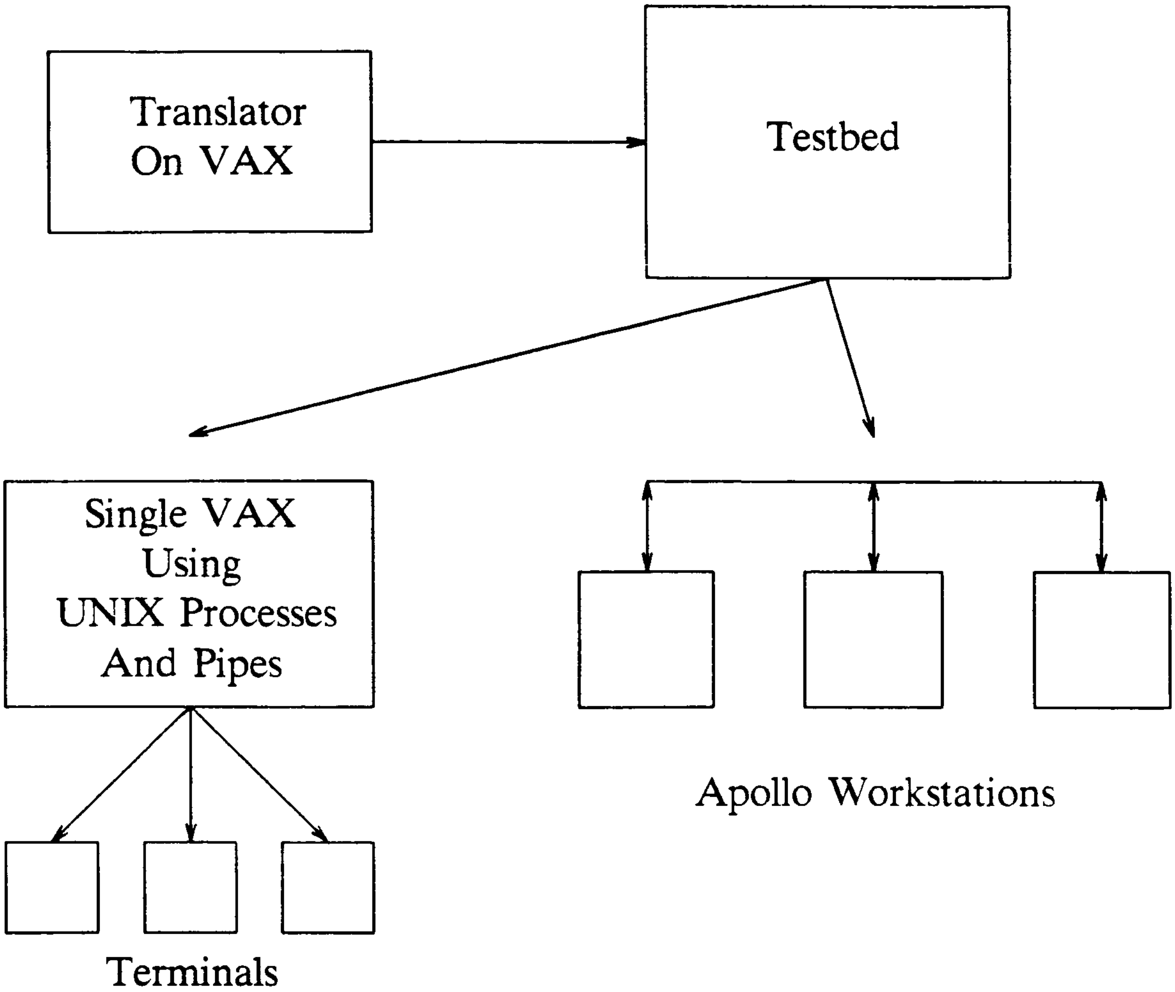
CONTROLLER

- Start, Stop, Single-Step Ada Tasks
- Control Communication At The Message Level
- Manage Breakpoints On A Per-Task Basis
- Arrange Failure Of Abstract Processors
- Collect And Display Information For Experimenter

TESTBED CONTROL



IMPLEMENTATION



Translator On VAX

Testbed

Single VAX Using UNIX Processes And Pipes

Terminals

Apollo Workstations

CONCLUSIONS

- ANSI Ada Does Not Support Processor Failures Well
- ANSI Ada Can Cope Given:
  - — Minor Semantic Enhancements
  - — No Syntax Changes
  - — Major Additions To Execution-Time System
- Demonstration Implementation Being Developed At The University Of Virginia
  - — A Feasibility Study
  - — Not Suitable For Production Use
- Realistic Application Needed For Evaluation Purposes
- Ideal Solution Requires Extensive Language Changes