LECTURE 9: Working Together

An Introduction to MultiAgent Systems
http://www.csc.liv.ac.uk/~mjw/pubs/imas

Working Together

- Why and how do agents work together?
- Important to make a distinction between:
  - *benevolent agents*
  - *self-interested agents*

Benevolent Agents

- If we “own” the whole system, we can design agents to help each other whenever asked
- In this case, we can assume agents are *benevolent*: our best interest is their best interest
- Problem-solving in benevolent systems is *cooperative distributed problem solving* (CDPS)
- *Benevolence simplifies the system design task enormously!*

Self-Interested Agents

- If agents represent individuals or organizations, (the more general case), then we cannot make the benevolence assumption
- Agents will be assumed to act to further their own interests, possibly at expense of others
- Potential for *conflict*
- May complicate the design task enormously

Task Sharing and Result Sharing

- Two main modes of cooperative problem solving:
  - *task sharing*:
    components of a task are distributed to component agents
  - *result sharing*:
    information (partial results, etc.) is distributed

The Contract Net

- A well known task-sharing protocol for *task allocation* is the *contract net*:
  1. Recognition
  2. Announcement
  3. Bidding
  4. Awarding
  5. Expediting

Recognition

- In this stage, an agent recognizes it has a problem it wants help with.
  Agent has a goal, and either…
  - realizes it cannot achieve the goal in isolation — does not have capability
  - realizes it would prefer not to achieve the goal in isolation (typically because of solution quality, deadline, etc.)

Announcement

- In this stage, the agent with the task sends out an *announcement* of the task which includes a *specification* of the task to be achieved
- Specification must encode:
  - description of task itself (maybe executable)
  - any constraints (e.g., deadlines, quality constraints)
  - meta-task information (e.g., “bids must be submitted by…”)
- The announcement is then *broadcast*

Bidding

- Agents that receive the announcement decide for themselves whether they wish to *bid* for the task
- Factors:
  - agent must decide whether it is capable of expediting task
  - agent must determine quality constraints & price information (if relevant)
- If they do choose to bid, then they submit a *tender*

Awarding & Expediting

- Agent that sent task announcement must choose between bids & decide who to “award the contract” to
- The result of this process is communicated to agents that submitted a bid
- The successful *contractor* then expedites the task
- May involve generating further manager-contractor relationships: *sub-contracting*

Issues for Implementing Contract Net

- How to…
  - …specify *tasks*?
  - …specify *quality of service*?
  - …select between competing offers?
  - …differentiate between offers based on multiple criteria?

The Contract Net

- An approach to *distributed problem solving*, focusing on task distribution
- Task distribution viewed as a kind of contract negotiation
- "Protocol" specifies *content* of communication, not just form
- Two-way transfer of information is natural extension of transfer of control mechanisms

Cooperative Distributed Problem Solving (CDPS)

- Neither global control nor global data storage — no agent has sufficient information to solve entire problem
- Control and data are distributed

CDPS System Characteristics and Consequences

- Communication is slower than computation
  - loose coupling
  - efficient protocol
  - modular problems
  - problems with large grain size

More CDPS System Characteristics and Consequences

- Any unique node is a potential bottleneck
  - distribute data
  - distribute control
  - organized behavior is hard to guarantee (since no one node has complete picture)

Four Phases to Solution, as Seen in Contract Net

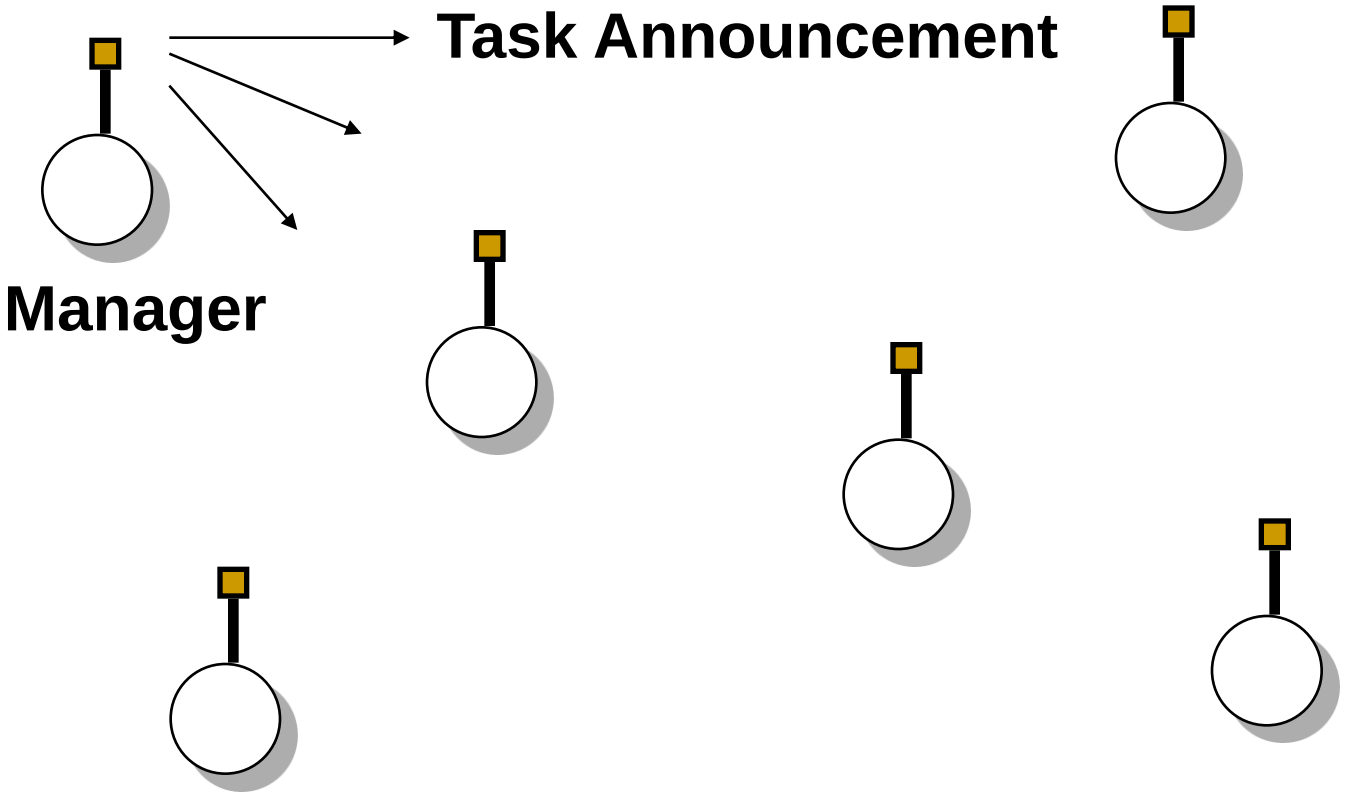
1. Problem Decomposition
2. Sub-problem distribution
3. Sub-problem solution
4. Answer synthesis

The contract net protocol deals with phase 2.

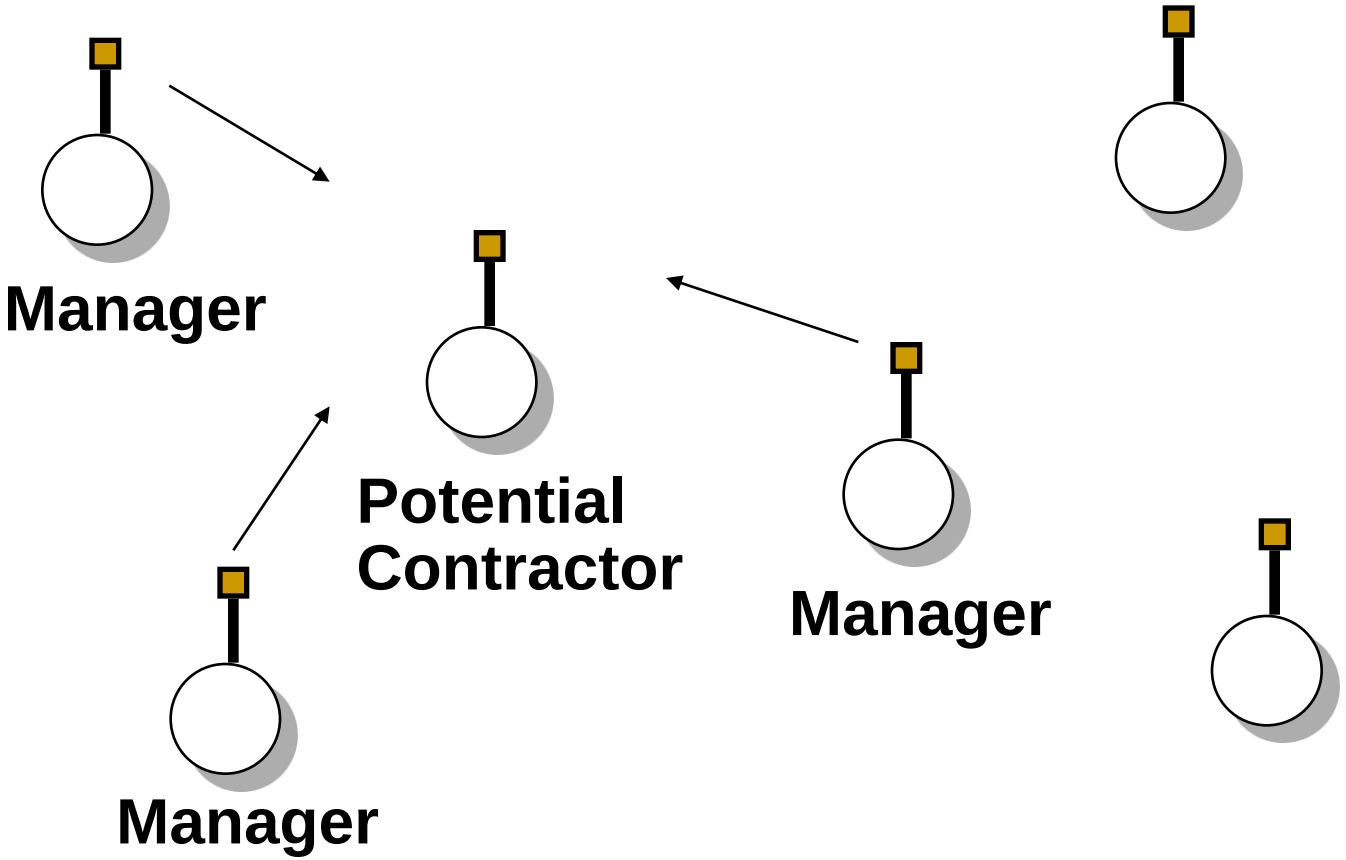
Contract Net

- The collection of nodes is the “contract net”
- Each node on the network can, at different times or for different tasks, be a manager or a contractor
- When a node gets a composite task (or for any reason can’t solve its present task), it breaks it into subtasks (if possible) and announces them (acting as a manager), receives bids from potential contractors, then awards the job (example domain: network resource management, printers, …)

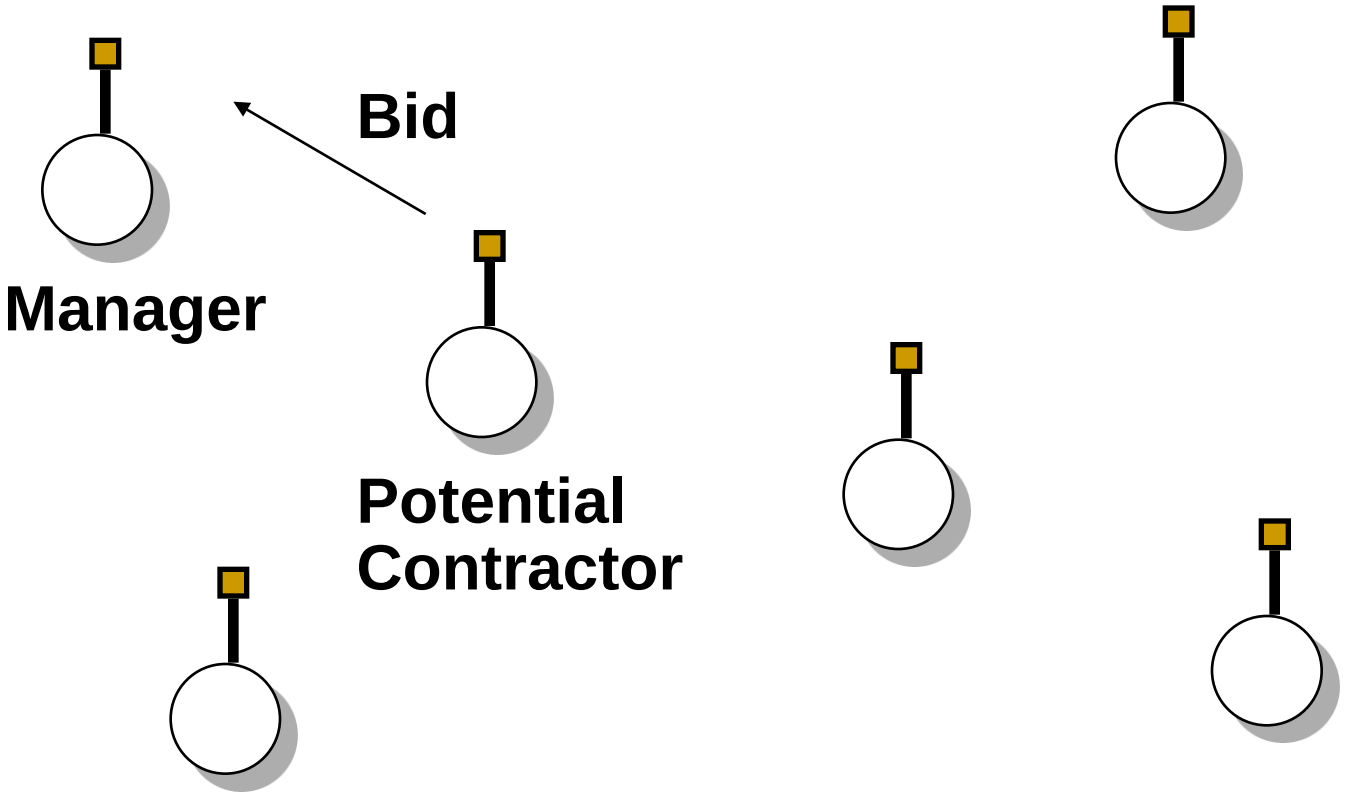
Node Issues Task Announcement



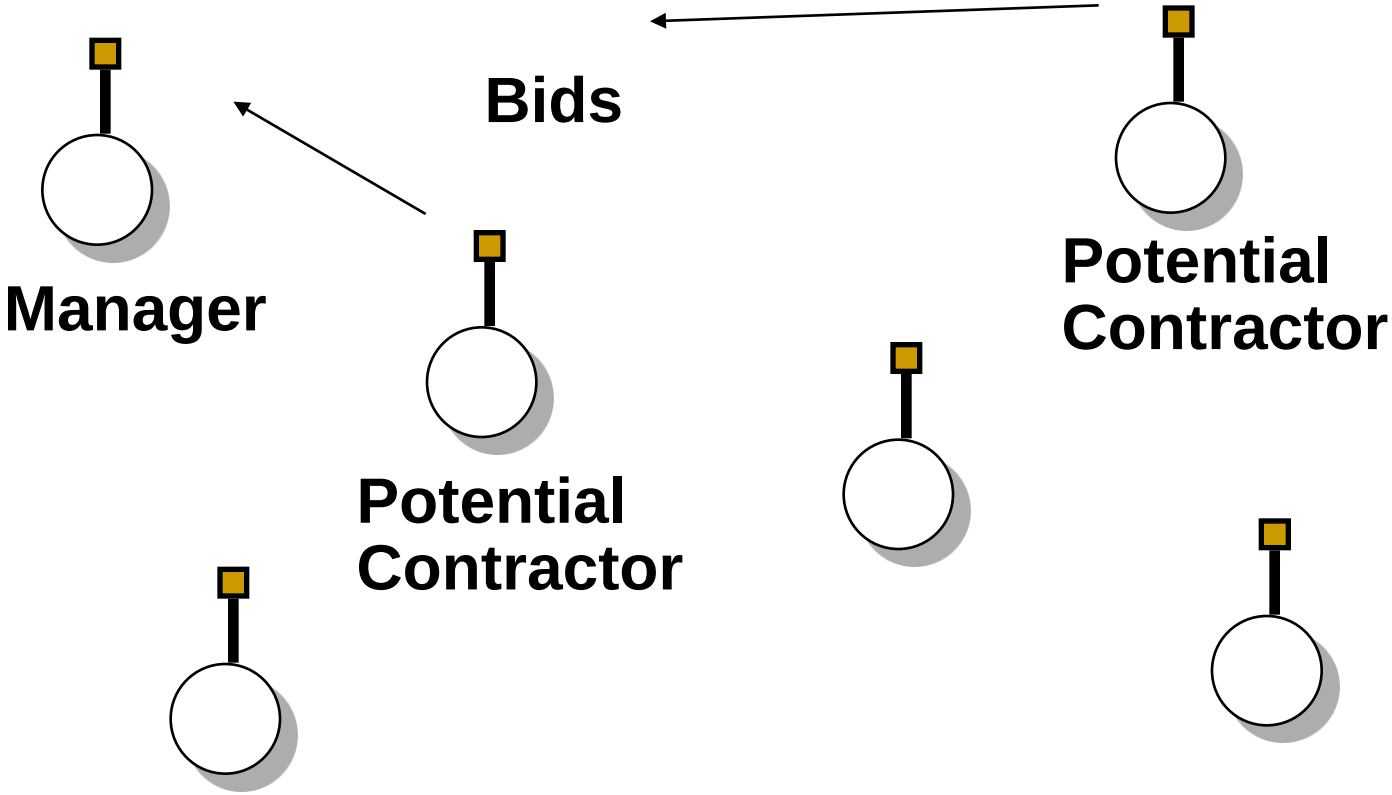
Idle Node Listening to Task Announcements



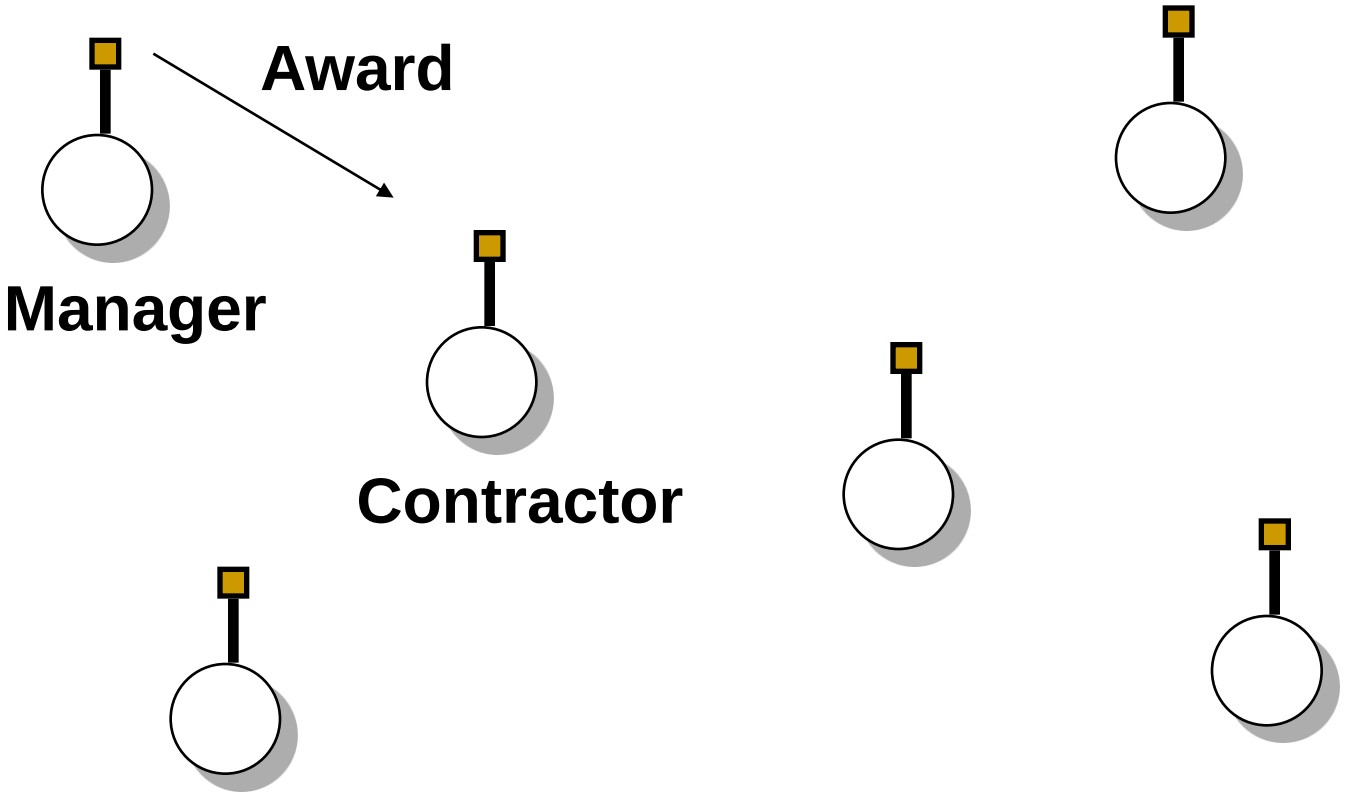
Node Submitting a Bid



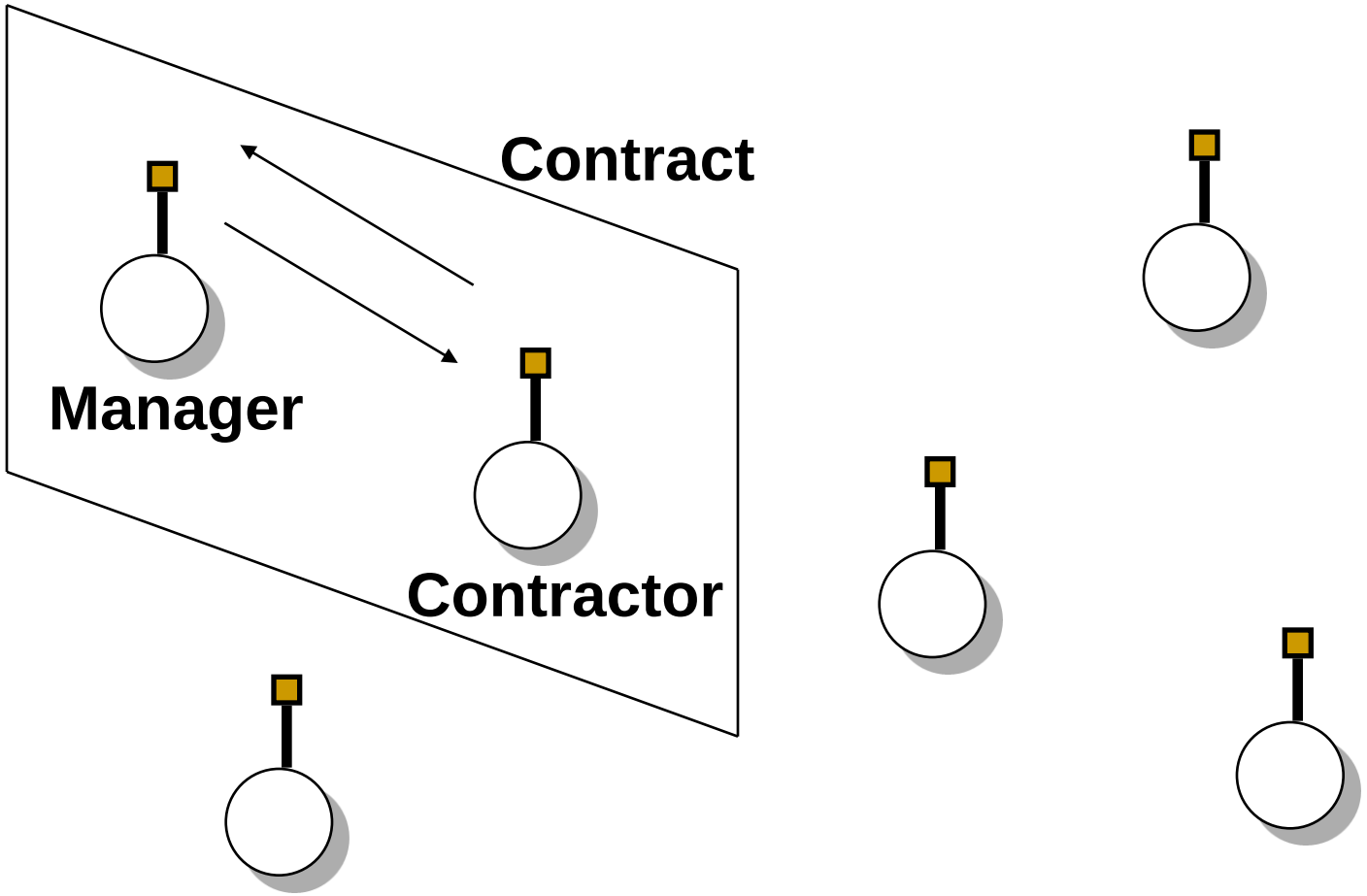
Manager listening to bids



Manager Making an Award



Contract Established



Domain-Specific Evaluation

- Task announcement message prompts potential contractors to use domain specific task evaluation procedures; there is deliberation going on, not just selection — perhaps no tasks are suitable at present
- Manager considers submitted bids using domain specific bid evaluation procedure

Types of Messages

- Task announcement
- Bid
- Award
- Interim report (on progress)
- Final report (including result description)
- Termination message (if manager wants to terminate contract)

Efficiency Modifications

- Focused addressing — when general broadcast isn't required
- Directed contracts — when manager already knows which node is appropriate
- Request-response mechanism — for simple transfer of information without overhead of contracting
- Node-available message — reverses initiative of negotiation process

Message Format

- Task Announcement Slots:
  - Eligibility specification
  - Task abstraction
  - Bid specification
  - Expiration time

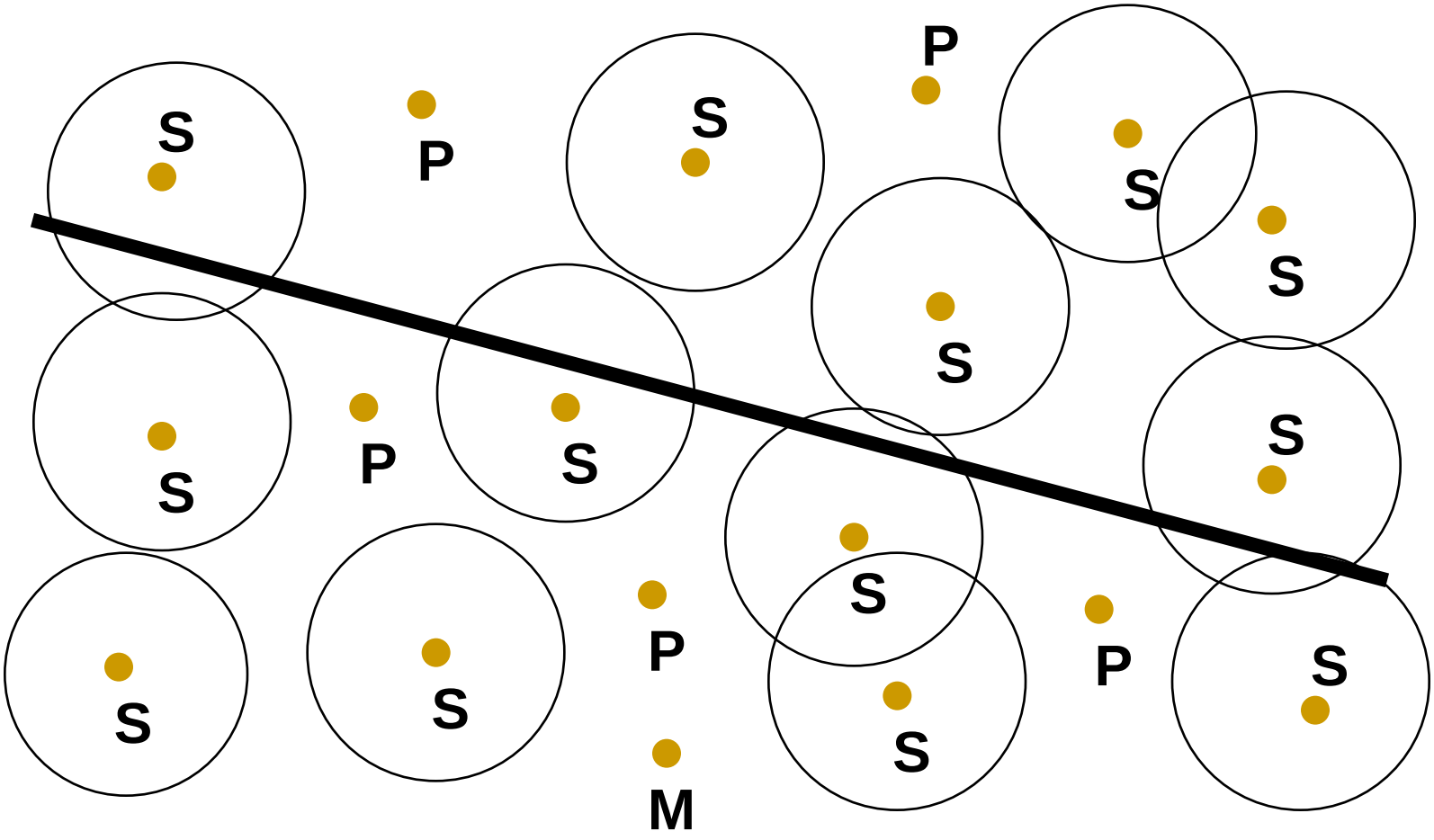
Task Announcement Example

(common internode language)

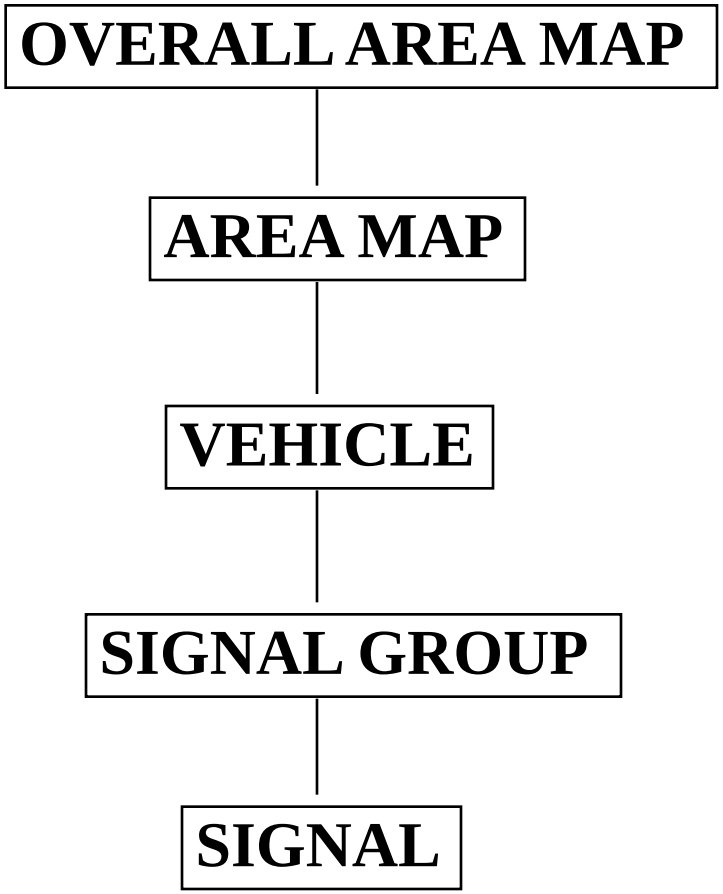
To: *
From: 25
Type: Task Announcement
Contract: 43–6
Eligibility Specification: Must-Have FFTBOX
Task Abstraction:
Task Type Fourier Transform
Number-Points 1024
Node Name 25
Position LAT 64N LONG 10W
Bid Specification: Completion-Time
Expiration Time: 29 1645Z NOV 1980

The existence of a common internode language allows new nodes to be added to the system modularly, without the need for explicit linking to others in the network (e.g., as needed in standard procedure calling) or object awareness (as in OOP)

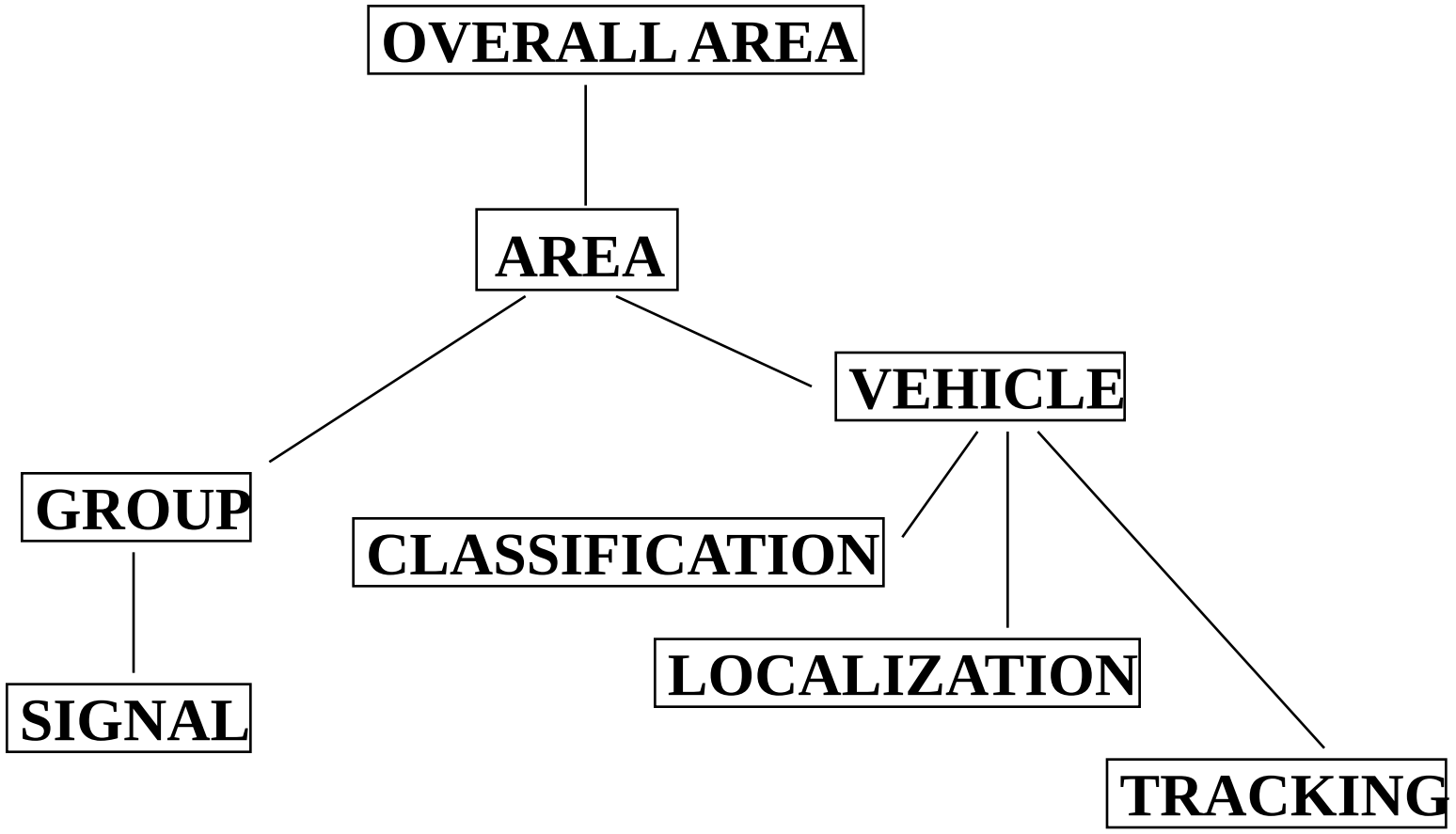
Example: Distributed Sensing System



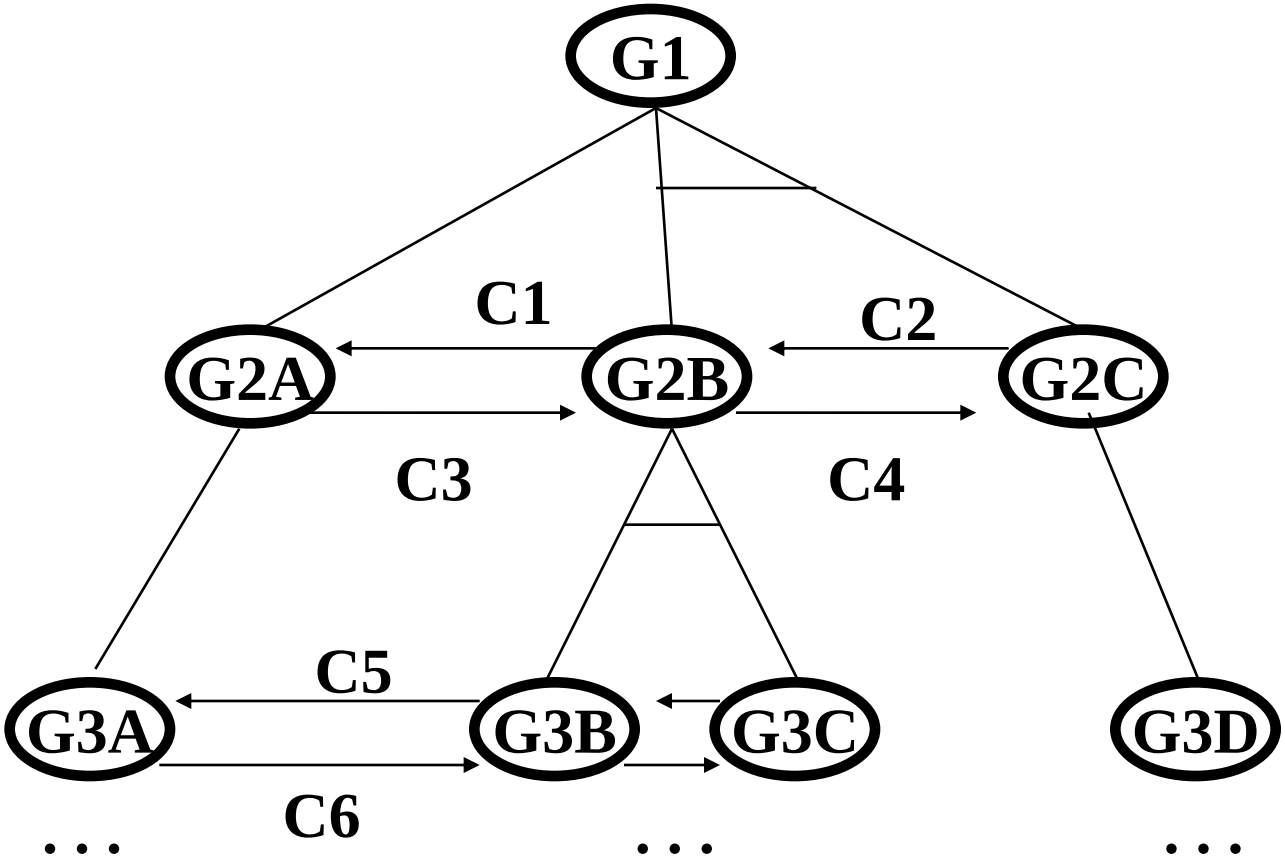
Data Hierarchy



Interpretation Task Hierarchy



Interpretation Problem Structure



Nodes and Their Roles

Nodes are simultaneously workers and supervisors

Monitor Node: integrate area maps into overall map
Area Task Manager: oversee area contractors

Area Contractor: integrate vehicle traffic into area map

Group Task Manager: oversee group contractors

Vehicle Task Manager: oversee vehicle contractors

Group Contractor: assemble signal features into groups

Signal Task Manager: overvsee signal contractors

Vehicle Contractor: Integrate Vehicle Information

Classification/Localization/Tracking Task Manager: overvsee respective contractors

Signal Contractor: provide signal features

Classification Contractor: classify vehicle

Localization Contractor: locate vehicle

Tracking Contractor:track vehicle

Note: Classification and Signal Contractors can also communicate…

Example: Signal Task Announcement

To: *
From: 25
Type: Task Announcement
Contract: 22–3–1
Eligibility Specification:
Must-Have SENSOR
Must-Have Position Area A
Task Abstraction:
Task Type Signal
Position LAT 47N LONG 17E
Area Name A Specification (…)
Bid Specification: Position Lat Long
Every Sensor Name Type
Expiration Time: 28 1730Z FEB 1979

Example: Signal Bid

To: 25
From: 42
Type: BID
Contract: 22–3–1
Node Abstraction:
LAT 47N LONG 17E
Sensor Name S1 Type S
Sensor Name S2 Type S
Sensor Name T1 Type T

Example: Signal Award

To: 42
From: 25
Type: AWARD
Contract: 22–3–1
Task Specification:
Sensor Name S1 Type S
Sensor Name S2 Type S

Features of Protocol

- Two-way transfer of information
- Local Evaluation
- Mutual selection (bidders select from among task announcements, managers select from among bids)
- Ex: Potential contractors select closest managers, managers use number of sensors and distribution of sensor types to select a set of contractors covering each area with a variety of sensors

Relation to other mechanisms for transfer of control

- The contract net views transfer of control as a runtime, symmetric process that involves the transfer of complex information in order to be effective
- Other mechanisms (procedure invocation, production rules, pattern directed invocation, blackboards) are unidirectional, minimally run-time sensitive, and have restricted communication

Suitable Applications

- Hierarchy of Tasks
- Levels of Data Abstraction
- Careful selection of Knowledge Sources is important
- Subtasks are large (and it's worthwhile to expend effort to distribute them wisely)
- Primary concerns are distributed control, achieving reliability, avoiding bottlenecks

Limitations

- Other stages of problem formulation are nontrivial:
  Problem Decomposition
  Solution Synthesis
- Overhead
- Alternative methods for dealing with task announcement broadcast, task evaluation, and bid evaluation

The Unified Blackboard architecture
The Distributed Blackboard architecture

The Hearsay II Speech Understanding System

- Developed at Carnegie-Mellon in the mid-1970's
- Goal was to reliably interpret connected speech involving a large vocabulary
- First example of the blackboard architecture, "a problem-solving organization that can effectively exploit a multi-processor system." (Fennel and Lesser, 1976)

The Motivations

- Real-time speech understanding required more processor power than could be expected of typical machines in 1975 (between 10 and 100 mips); parallelism offered a way of achieving that power
- There are always problems beyond the reach of current computer power—parallelism offers us hope of solving them now
- The complicated structure of the problem (i.e., speech understanding) motivated the search for new ways of organizing problem solving knowledge in computer programs

Result Sharing in Blackboard Systems

- The first scheme for cooperative problem solving: the *blackboard system*
- Results shared via shared data structure (BB)
- Multiple agents (KSs/KAs) can read and write to BB
- Agents write partial solutions to BB
- BB may be structured into hierarchy
- Mutual exclusion over BB required $\Rightarrow$ bottleneck
- Not concurrent activity
- Compare: LINDA tuple spaces, JAVASPACES

Result Sharing in Subscribe/Notify Pattern

- Common design pattern in OO systems: *subscribe/notify*
- An object *subscribes* to another object, saying "tell me when event *e* happens"
- When event *e* happens, original object is notified
- Information pro-actively *shared* between objects
- Objects required to know about the *interests* of other objects $\Rightarrow$ inform objects when relevant information arises

The Blackboard Architecture

1. Multiple, diverse, independent and asynchronously executing knowledge sources (KS's)
2. Cooperating (in terms of *control*) via a generalized form of hypothesize-and-test, involving the data-directed invocation of KS processes
3. Communicating (in terms of *data*) via a shared blackboard-like database

A "Knowledge Source" (KS)

"An agent that embodies the knowledge of a particular aspect of a problem domain," and furthers the solution of a problem from that domain by taking actions based on its knowledge.

In speech understanding, there could be distinct KS's to deal with acoustic, phonetic, lexical, syntactic, and semantic information.

Abstract Model

- The blackboard architecture is a parallel production system (productions: P $\rightarrow$ A)
- Preconditions are satisfied by current state of the (dynamic) blackboard data structure, and trigger their associated Action
- Actions presumably alter the blackboard data structure
- Process halts when no satisfied precondition is found, or when a “stop” operation is executed (failure or solution)

The Blackboard

- Centralized multi-dimensional data structure
- Fundamental data element is called a node (nodes contain data fields)
- Readable and writable by any precondition or KS (production action)
- Preconditions are procedurally oriented and may specify arbitrarily complex tests

The Blackboard (continued)

- Preconditions have “pre-preconditions” that sense primitive conditions on the blackboard, and schedule the real (possibly complex) precondition test
- KS processes are also procedurally oriented, generally hypothesize new data (added to data base) or verify or modify data already in the data base

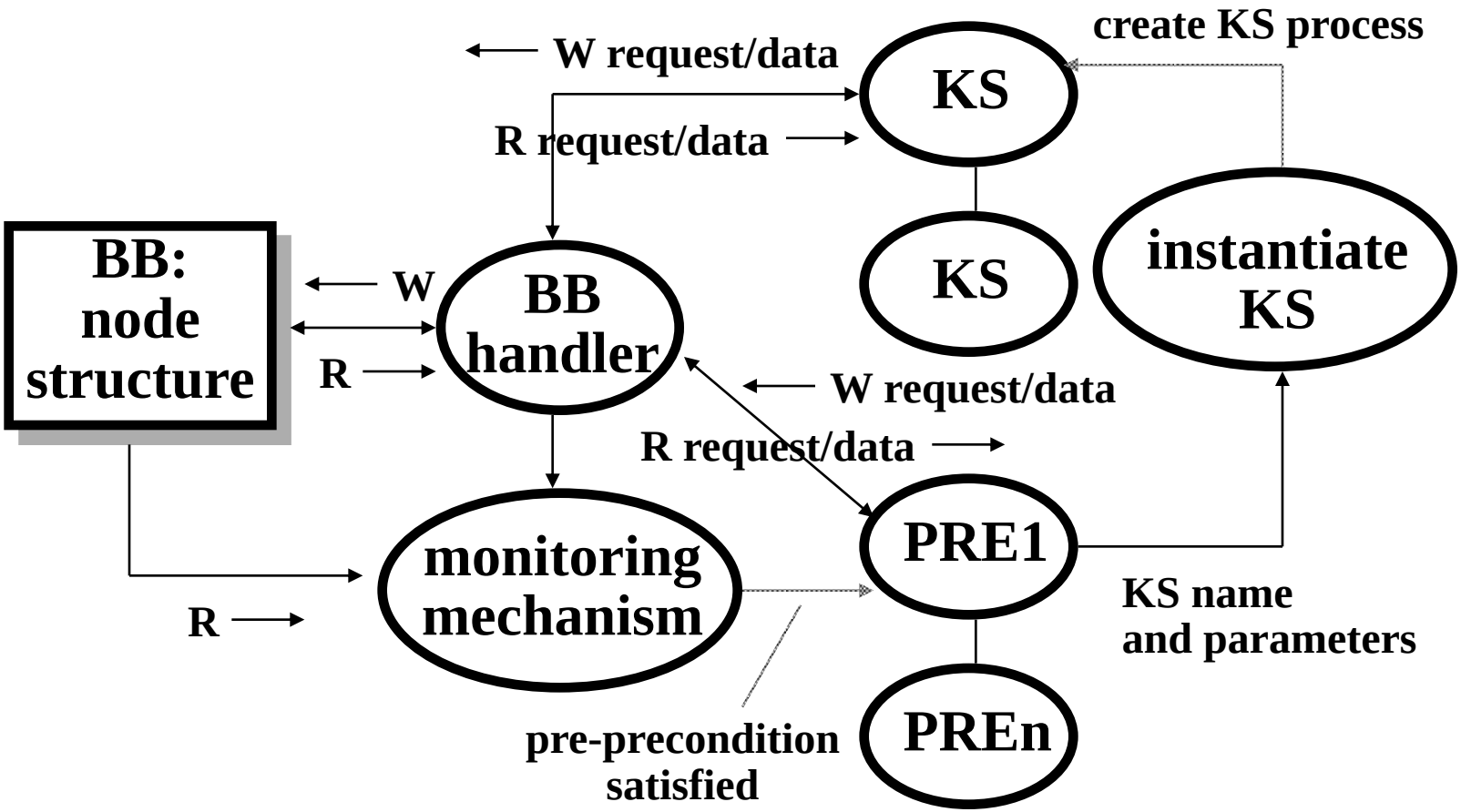
The Blackboard (continued)

- Hypothesize-and-test paradigm — hypotheses representing partial problem solutions are generated and then tested for validity
- Neither precondition procedures nor action procedures are assumed to be “indivisible”; activity is occurring concurrently (multiple KS’s, multiple precondition tests…)

Multi-dimensional Blackboard

- For example, in Hearsay-II, the system data base had three dimensions for nodes:
  - informational level (e.g., phonetic, surface-phonemic, syllabic, lexical, and phrasal levels)
  - utterance time (speech time measured from beginning of input)
  - data alternatives (multiple nodes can exist simultaneously at the same informational level and utterance time)

Hearsay-II System Organization



Modularity

- The “KS’s are assumed to be independently developed” and don’t know about the explicit existence of other KS’s — communication must be indirect
- Motivation: the KS’s have been developed by many people working in parallel; it is also useful to check how the system performs using different subsets of KS’s

KS Communication

- Takes two forms:
  - Database monitoring to collect data event information for future use (local contexts and precondition activation)
  - Database monitoring to detect data events that violate prior data assumptions (tags and messages)

Local Contexts

- Each precondition and KS process that needs to remember the history of database changes has its own local database (local context) that keeps track of the global database changes that are relevant to that process
- When a change (data event) occurs on the blackboard, the change is broadcast to all interested local contexts (data node name and field name, with old value of field)
- The blackboard holds only the most current information; local contexts hold the history of changes

Data Integrity

- Because of the concurrency in blackboard access by preconditions and KS's (and the fact that they are not indivisible), there is a need to maintain data integrity:
  - Syntactic (system) integrity: e.g., each element in a list must point to another valid list element
  - Semantic (user) integrity: e.g., values associated with adjacent list elements must be always less than 100 apart

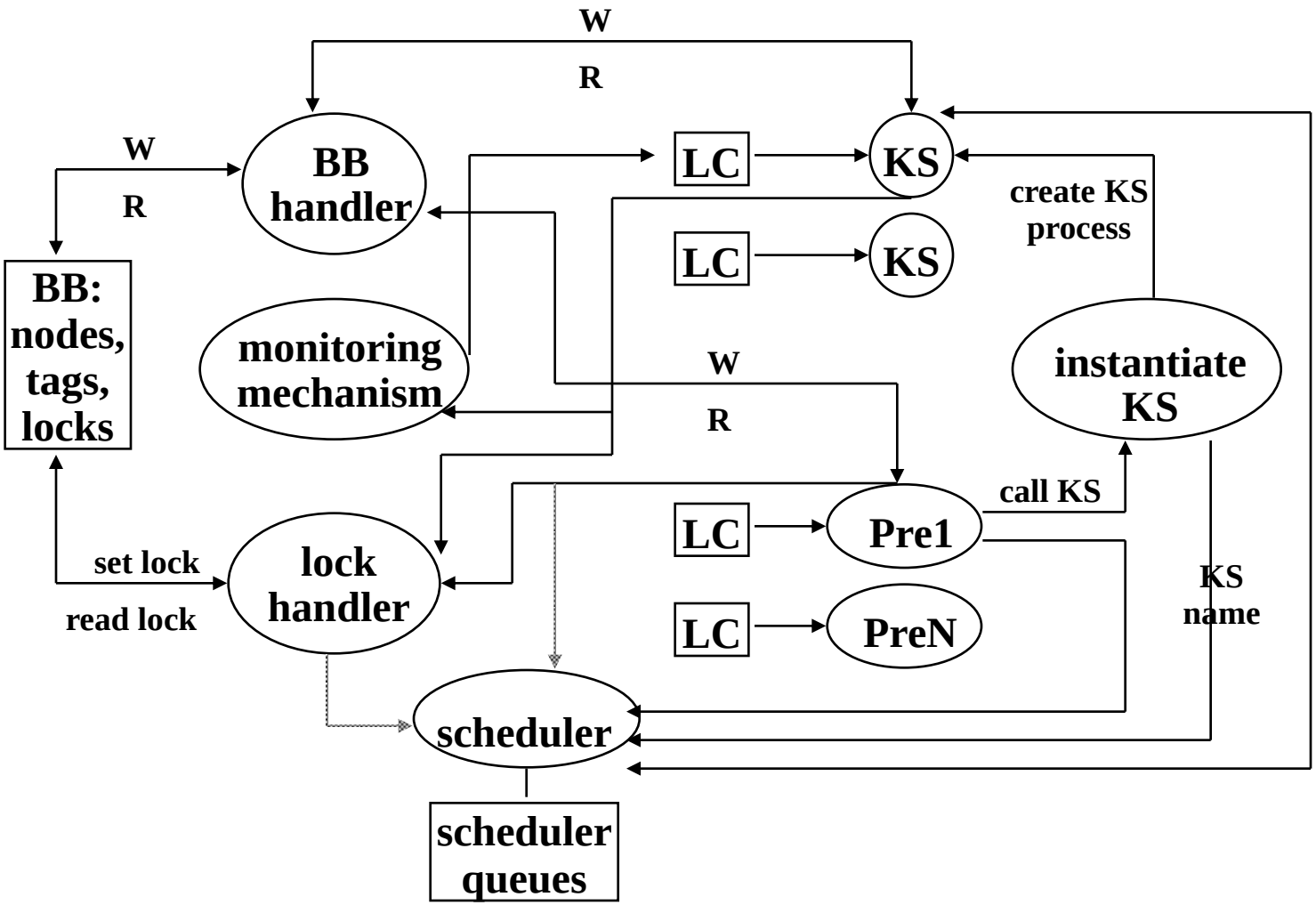
Locks

- Locks allow several ways for a process to acquire exclusive or read-only data access:
  - Node locking (specific node)
  - Region locking (a collection of nodes specified by their characteristics, e.g., information level and time period)
  - Node examining (read-only access to other processes)
  - Region examining (read-only)
  - Super lock (arbitrary group of nodes and regions can be locked)

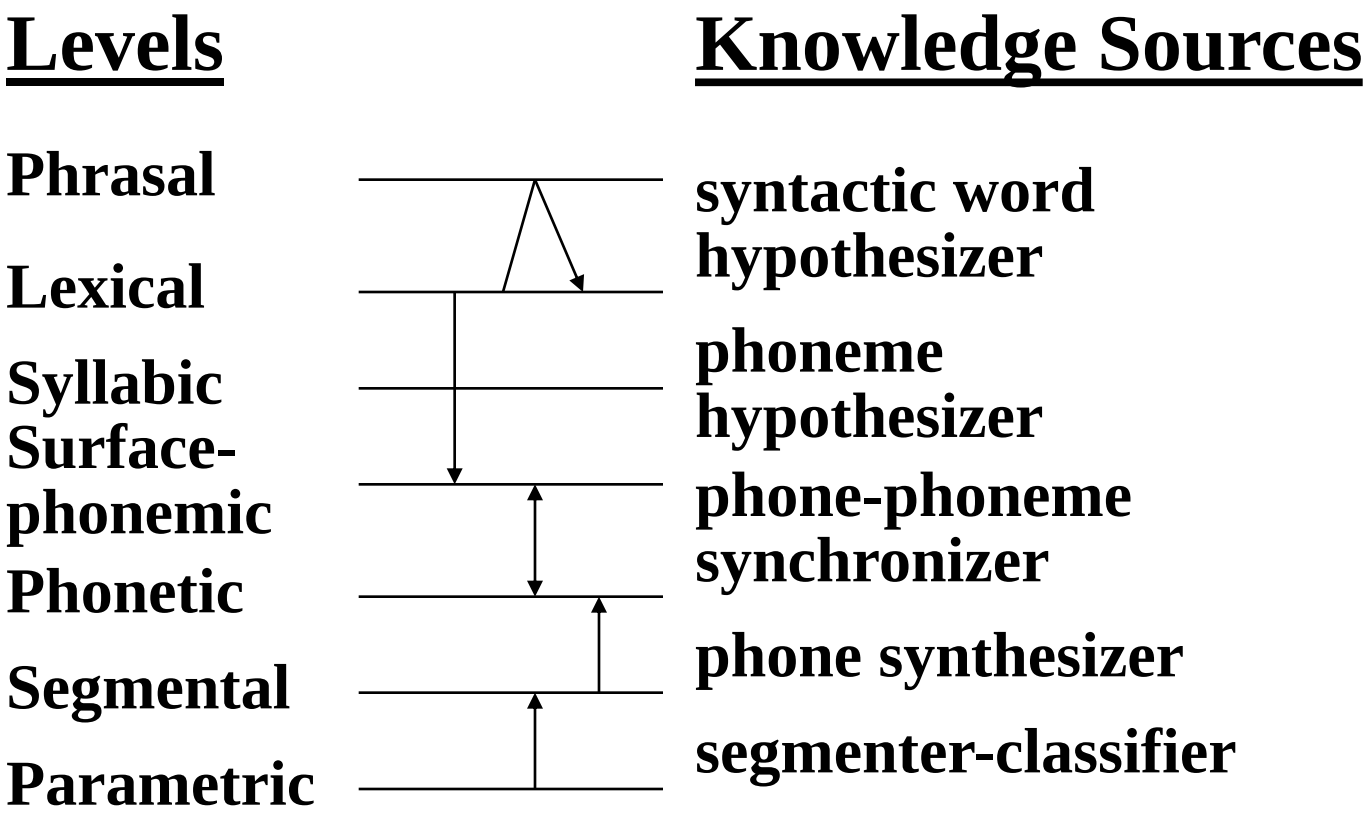
Tagging

- Locking can obviously cut down on system parallelism, so the blackboard architecture allows data-tagging:
  - Data assumptions placed into the database (defining a critical data set); other processes are free to continue reading and writing that area, but if the assumptions are invalidated, warning messages are sent to relevant processes
  - Precondition data can be tagged by the precondition process on behalf of its KS, so that the KS will know if the precondition data has changed before action is taken

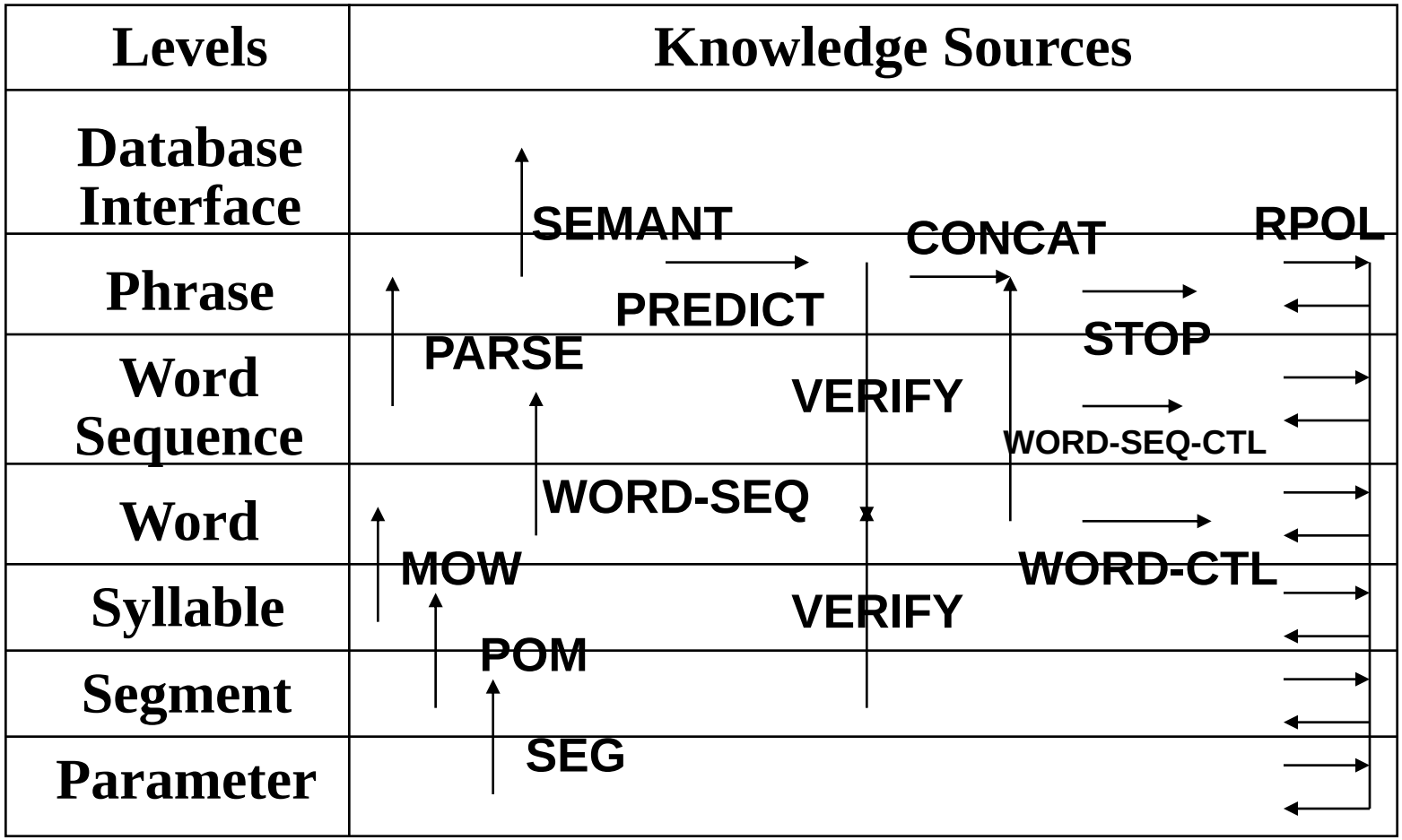
Hearsay II System Organization (partial)



Hearsay II Blackboard Organization (Simplified)



Hearsay II — Another View



The KS's

Signal Acquisition, Parameter Extraction, Segmentation and Labeling:

SEG: Digitizes the signal, measures parameters, produces labeled segmentation

Word Spotting: POM: Creates syllable-class hypothese from segments MOW: Creates word hypotheses from syllable classes

WORD-CTL: Controls the number of word hypotheses that MOW creates

Phrase-Island Generation: WORD-SEQ: Creates word-sequence hypotheses that represent potential phrases, from word hypotheses and weak grammatical knowledge WORD-SEQ-CTL: Control the number of hypotheses that WORD-SEQ creates PARSE: Attempts to parse a word-sequence and, if successful, creates a phrase hypothesis from it

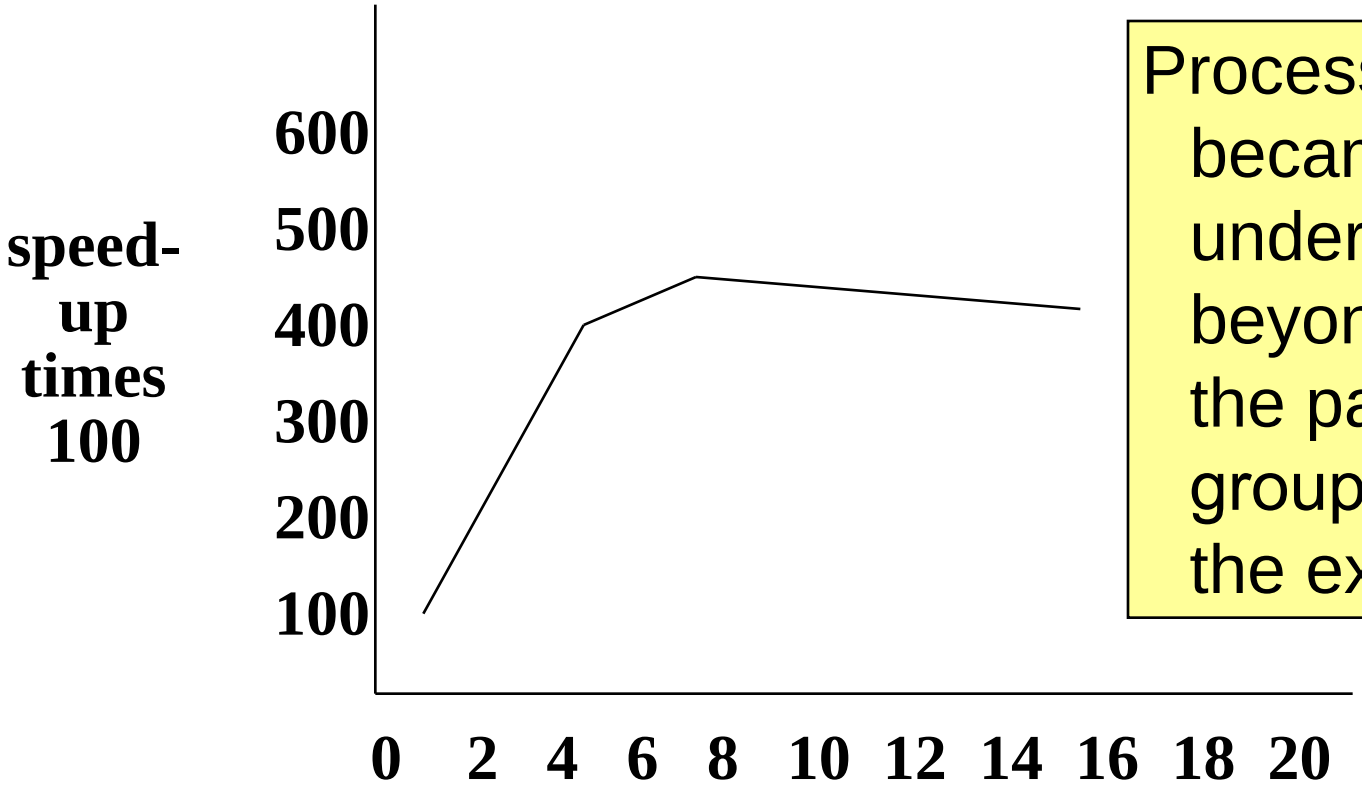
Phrase Extending: PREDICT: Predicts all possible words that might syntactically precede or follow a given phrase VERIFY: Rates the consistency between segment hypotheses and a contiguous word-phrase pair CONCAT: Creates a phrase hypothesis from a verified, contiguous word-phrase pair

Rating, Halting, and Interpretation: RPOL: Rates the credibility of each new or modified hypothesis, using information placed on the hypothesis by other KS's STOP: Decides to halt processing (detects a complete sentence with a sufficiently high rating, or notes the system has exhausted its available resources), and selects the best phrase hypothesis (or a set of complementary phrase hypotheses) as the output SEMANT: Generates an unambiguous interpretation for the information-retrieval system which the user has queried

Timing statistics (non-overlapping)

- Blackboard reading 16%
- Blackboard writing 4%
- Internal computations of processes 34%
  - Local context maintenance 10%
  - Blackboard access synchronization 27%
  - Process handling 9%
  - (i.e., multiprocess overhead almost 50%)

Effective Parallelism According to Processor Utilization



Processors became underutilized beyond 8 — for the particular group of KS's in the experiment

So now we want distributed interpretation…

- Sensor networks (low-power radar, acoustic, or optical detectors, seismometers, hydrophones…)
- Network traffic control
- Inventory control
- Power network grids
- Mobile robots

Distributed Interpretation

- Working Assumption Number 1: Interpretation techniques that search for a solution by the incremental aggregation of partial solutions are especially well-suited to distribution
  - Errors and uncertainty from input data and incomplete or incorrect knowledge are handled as an integral part of the interpretation process
- Working Assumption Number 2: Knowledge-based AI systems can handle the additional uncertainty introduced by a distributed decomposition without extensive modification

Distributed Interpretation

- The early experiments with distributing Hearsay-II across processors were simple; later experiments (e.g., the DVMT) were much more rigorous:
  1. At first, few (only 3) nodes
  2. Few experiments (heavy simulation load)
  3. “There is probably no practical need for distributing a single-speaker speech-understanding system.”

How do we go about distributing?

- Options:
  - Distribute information (the blackboard is multi-dimensional — each KS accesses only a small subspace)
  - Distribute processing (KS modules are largely independent, anonymous, asynchronous)
  - Distribute control (send hypotheses among independent nodes, activating KS's)

Distributed Interpretation

- The multi-processor implementation of Hearsay-II, with explicit synchronization techniques to maintain data integrity, achieved a speed-up factor of six — but the need for any synchronization techniques is a bad idea for a true distributed interpretation architecture

The uni-processor and synchronized multi-processor versions…

1. The scheduler (which requires a global view of pending KS instantiations [scheduling queues] and the focus-of-control database) is centralized
2. The blackboard monitor (updating focus-of-control database and scheduling queues) is centralized
3. Patterns of KS blackboard access overlap, hard to have compartmentalized subspaces

Distributed Interpretation

- In fact, the explicit synchronization techniques could be eliminated, and the speedup factor increased from 6 to 15
- All sorts of internal errors occurred because of the lack of centralized synchronization, but the architecture was robust enough to (eventually) correct these errors

Dimensions of Distribution

Information:

- Distribution of the blackboard:
  - Blackboard is distributed with no duplication of information
  - Blackboard is distributed with possible duplication, synchronization insures consistency
  - Blackboard is distributed with possible duplications and inconsistencies

Dimensions of Distribution

Information (continued):

- Transmission of hypotheses:
  - Hypotheses are not transmitted beyond the local node that generates them
  - Hypotheses may be transmitted directly to a subset of nodes
  - Hypotheses may be transmitted directly to all nodes

Dimensions of Distribution

Processing:

- Distribution of KS's:
  - Each node has only one KS
  - Each node has a subset of KS's
  - Each node has all KS's
- Access to blackboard by KS's:
  - A KS can access only the local blackboard
  - A KS can access a subset of nodes' blackboards
  - A KS can access any blackboard in the network

Dimensions of Distribution

Control:

- Distribution of KS activation:
  - Hypothesis change activates only local node's KS's
  - Change activates subset of nodes' KS's
  - Change activates KS's in any node
- Distribution of scheduling/focus-of-control:
  - Each node does its own scheduling, using local information
  - Each subset of nodes has a scheduler
  - A single, distributed database is used for scheduling

Two ways of viewing the distribution of dynamic information

1. There is a virtual global database; local nodes have partial, perhaps inconsistent views of the global database
2. Each node has its own database; the union of these across all nodes, with any inconsistencies, represents the total system interpretation — not a system that's been distributed, but a network of cooperating systems

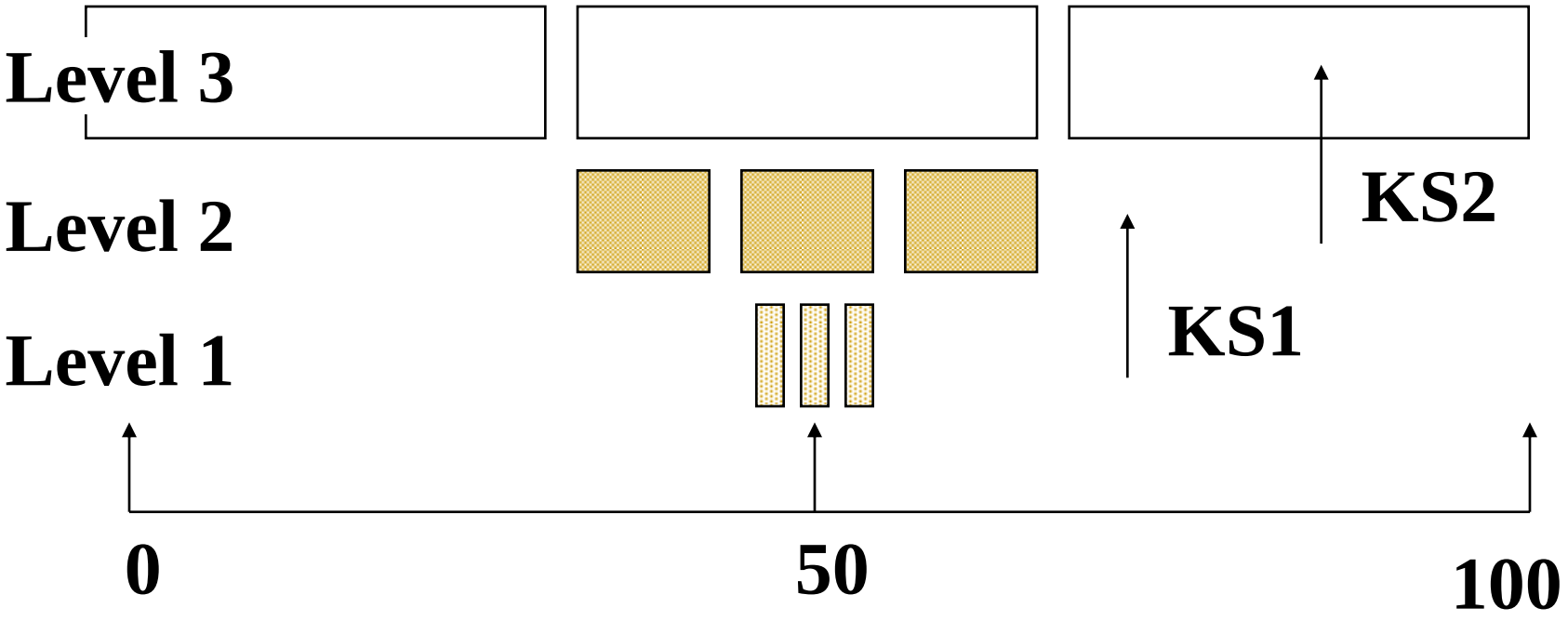
Focusing the nodes

- The blackboard is multi-dimensional: one dimension might be the information level
- Other dimensions, orthogonal to the information level, fix the location of the event which the hypothesis describes:
  - *signal interpretation*: physical location
  - *speech understanding*: time
  - *image understanding*: 2 or 3 dimensional space
  - *radar tracking*: 3 dimensional space

Focusing the nodes

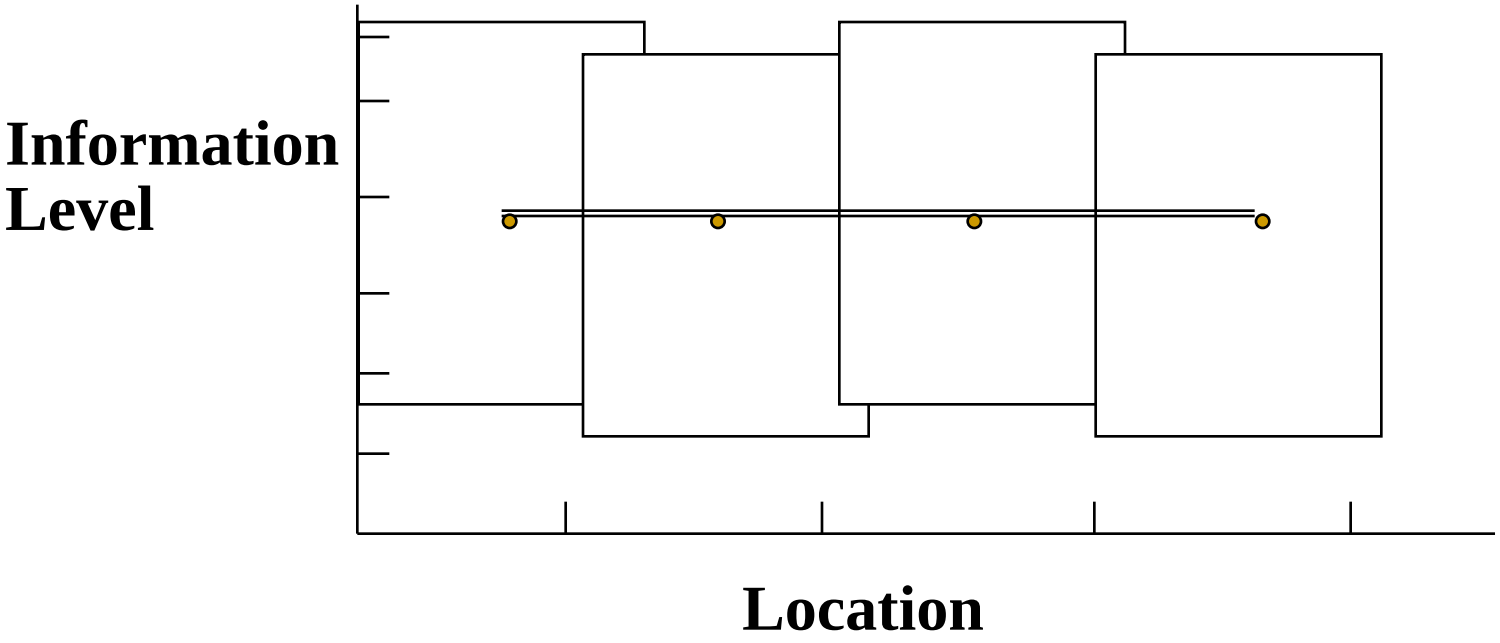
- All levels of the system, together with the full extent of the location dimension(s), define the largest possible scope of a node
- The area of interest of a node is the portion of this maximum scope representable in the node's local blackboard
- The location segment extends beyond the range of the local sensor (to allow the node to acquire context information from other nodes)
- At higher levels, the location dimension tends to get larger

Example of areas of interest

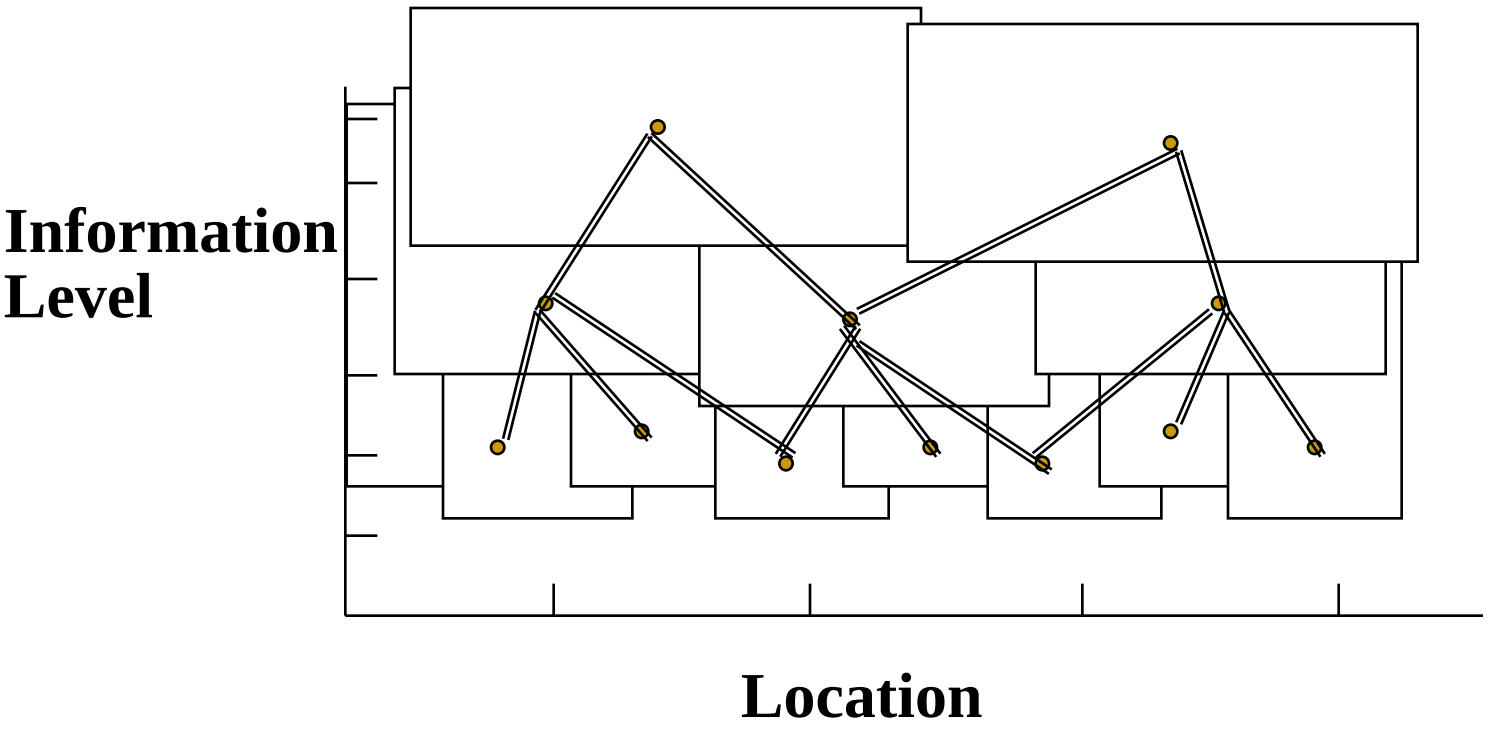


Network Configurations

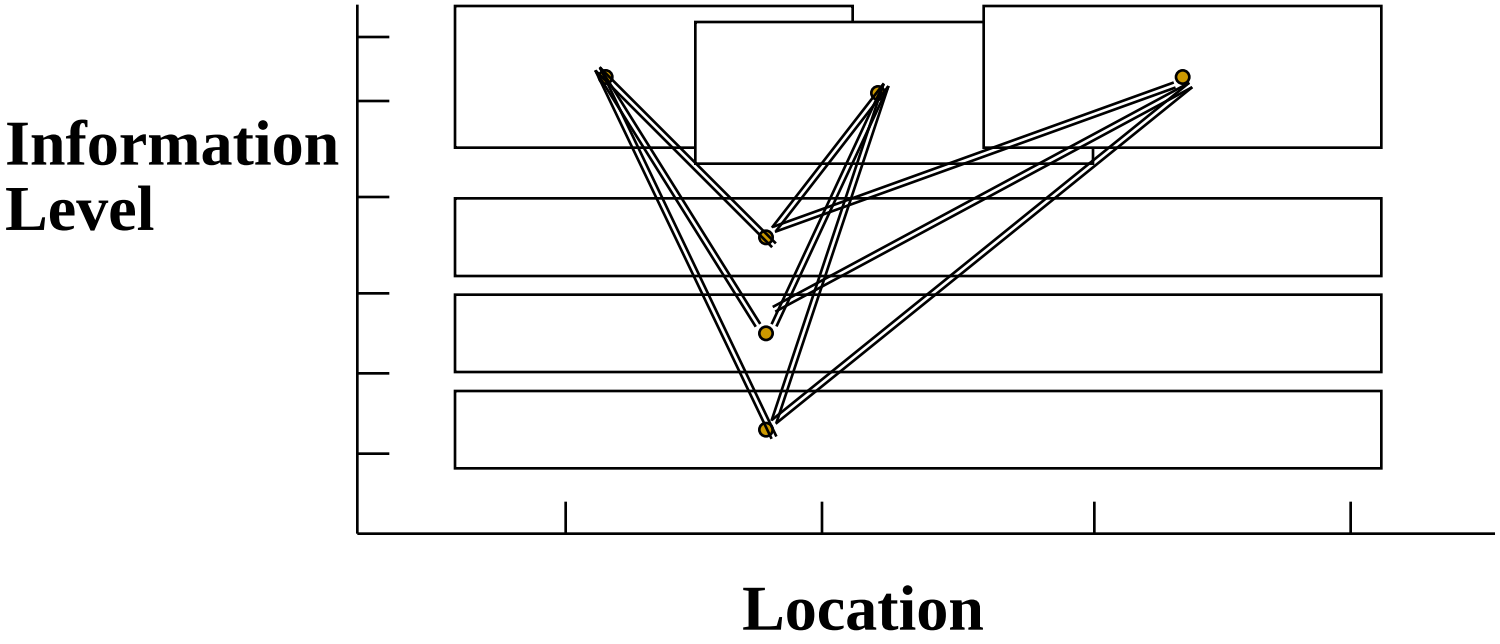
All nodes contain the same set of KS's and levels — the configuration is flat:



Overlapping hierarchical organization:



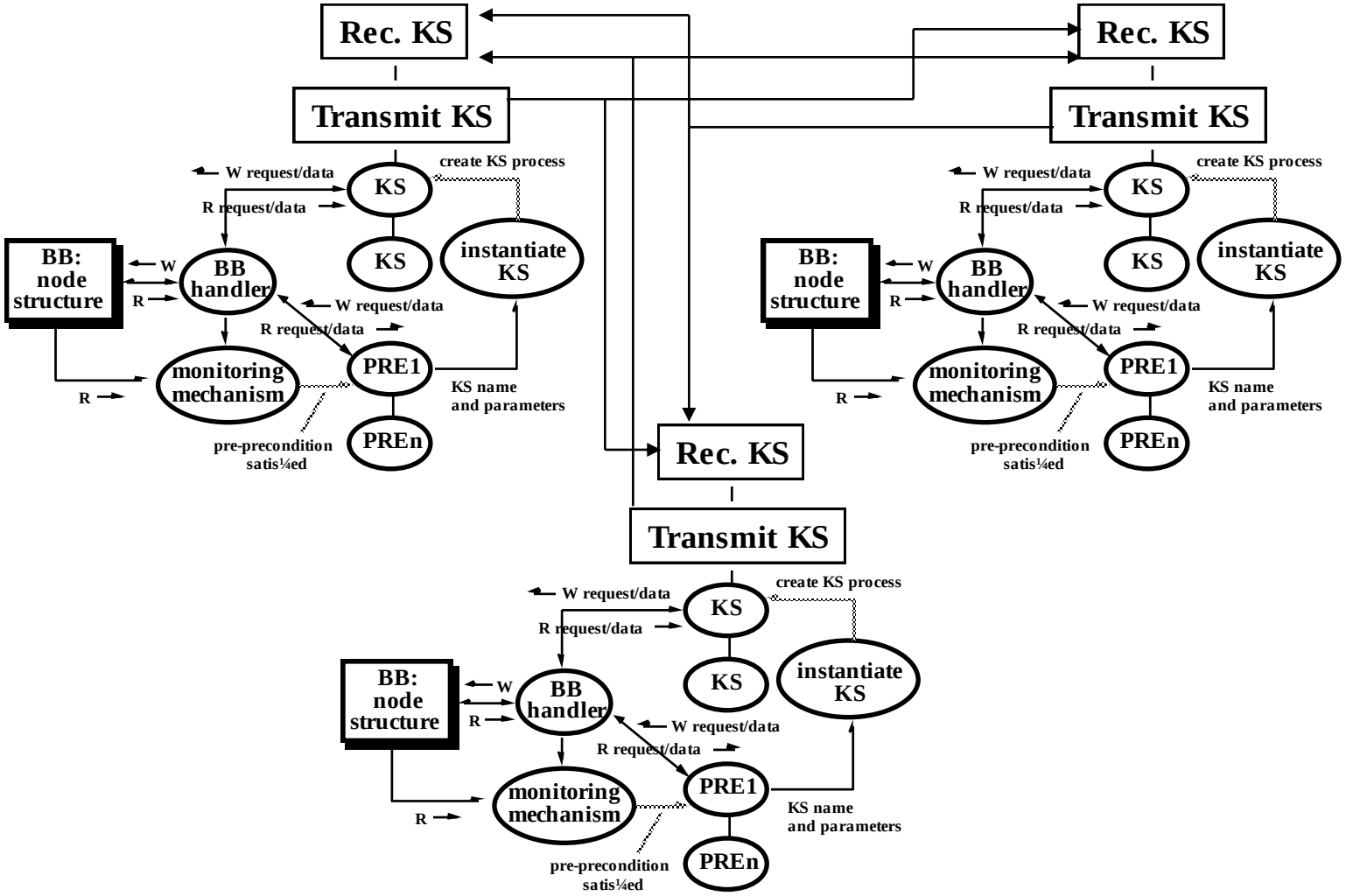
Matrix configuration (each of a set of general-purpose nodes at the higher level makes use of information from lower level specialists):



Internode Communication

- In Hearsay-II, all inter-KS communication is handled by the creation, modification, and inspection of hypotheses on the blackboard
- In the distributed Hearsay-II architecture, inter-node communication is handled the same way
- Added to the local node's KS's is a RECEIVE KS and a TRANSMIT KS

Network of Hearsay-II Systems



Internode Communication

- In general, communication occurs to “nearby” nodes, based on the location dimensions and overlapping areas of interest
- As a heuristic this makes sense: close nodes are likely to be most interested in your information (and have interesting information for you)
- Those are also the nodes with whom it is cheapest to communicate

Communication Policy

- Nodes can deal with the transmission and receipt of information in different ways
- Basic Policy:
  - Accept any information within the area of interest and integrate it as if it had been generated locally
  - Select for transmission hypotheses whose estimated impact is highest and haven't been transmitted yet
  - Broadcast them to all nodes that can receive them directly

Communication Policy

- The key point here is that there is an incremental transmission mechanism (with processing at each step)
- A limited subset of a node's information is transmitted, and only to a limited subset of nodes

Variants

- The “locally complete” strategy: transmit only those hypotheses for which the node has exhausted all possible local processing and which then have a high-impact measure
- Good if most hypotheses of small scope are incorrect and if most small-scope hypotheses can be refuted by additional processing in the creating node

Advantages of Locally Complete Strategy

1. Cut down on communication (fewer hypotheses are sent)
2. Reduce processing requirements of receiving nodes (they get fewer hypotheses)
3. Avoid redundant communication (when areas of interest overlap)
4. Increase the relevance of transmitted hypotheses

- Disadvantage of locally complete strategy:

1. Loss of timeliness (earlier transmission might have cut down on search)

Areas of Interest

- Sometimes, nodes that have overlapping areas of interest are the only ones to communicate — but sometimes this might not be sufficient (if there are discontinuities)
- The transmission of input/output characteristics by a node, i.e., its area of interest, can inform other nodes of the kinds of information it needs and the kinds it produces
  - This is the transmission of meta-information, an expansion of a node's area of interest sufficient to get the information it needs)

The Experiments…

- Described in “Distributed Interpretation: A Model and Experiment,” V. R. Lesser and L. D. Erman, in Readings in Distributed Artificial Intelligence.
- One important issue here, expanded later in the DVMT, was the issue of distraction caused by the receipt of incorrect information — and how a node can protect itself from being distracted

Overview

- *Mechanism 1*: Opportunistic nature of information gathering
  - *Impact 1*: Reduced need for synchronization
- *Mechanism 2*: Use of abstract information
  - *Impact 2*: Reduced internode communication
- *Mechanism 3*: Incremental aggregation
  - *Impact 3*: Automatic error detection

Overview (continued)

- *Mechanism 4*: Problem solving as a search process
  - *Impact 4*: Internode parallelism
- *Mechanism 5*: Functionally-accurate definition of solution
  - *Impact 5*: Self-correcting

The Distributed Vehicle Monitoring Testbed

- Coherent Cooperation
- Partial Global Plans

Functionally Accurate/ Cooperative (FA/C) Systems

- A network Problem Solving Structure:
  1. Functionally accurate: “the generation of acceptably accurate solutions without the requirement that all shared intermediate results be correct and consistent”
  2. Cooperative: an “iterative, coroutine style of node interaction in the network”

Hoped-for Advantages of FA/C systems

- Less communication will be required to communicate high-level, tentative results (rather than communicating raw data and processing results)
- Synchronization can be reduced or eliminated, resulting in more parallelism
- More robust behavior (error from hardware failure are dealt with like error resulting from incomplete or inconsistent information)

Need for a Testbed

- The early Hearsay-II experiments had demonstrated the basic viability of the FA/C network architecture, but had also raised questions that could not be adequately answered:
  - Wasted effort (node produces good solution, and having no way to redirect itself to new problems, generated alternative, worse, solutions)

Need for a Testbed

- The impact of distracting information: a node with noisy data would quickly generate an innaccurate solution, then transmit this bad solution to other nodes (that were working on better data) — and distract those other nodes, causing significant delays

Direction of the Research, after the Hearsay-II Phase:

- "We believe that development of appropriate network coordination policies (the lack of which resulted in diminished network performance for even a small network) will be crucial to the effective construction of large distributed problem solving networks containing tens to hundreds of processing nodes."

Why not continue using the Hearsay-II domain?

- Time-consuming to run the simulation, since the underlying system was large and slow
- The speech task didn't naturally extend to larger numbers of nodes (partly because the speech understanding problem has one-dimensional [time] sensory data)

Why not continue using the Hearsay-II domain?

- Hearsay-II had been tuned, for efficiency reasons, so that there was a “tight-coupling among knowledge sources and the elimination of data-directed control at lower blackboard levels” — in direct contradiction of the overall system philosophy! Tight coupling causes problems with experimentation (e.g., eliminating certain KS’s)
- The KS code was large and complex, so difficult to modify

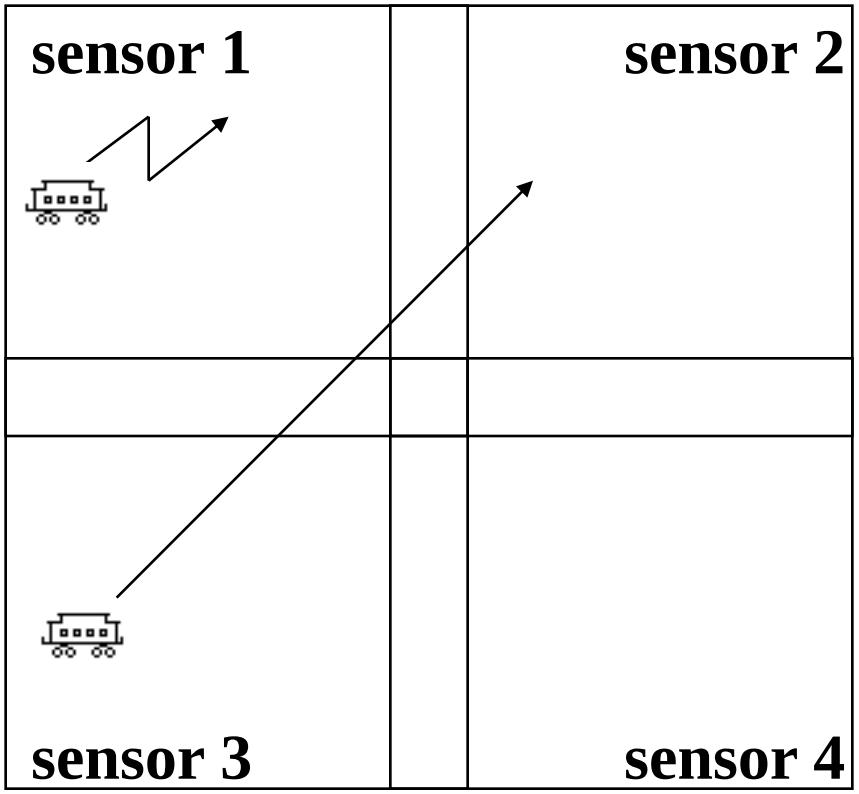
Why not continue using the Hearsay-II domain?

- "…the flexibility of the Hearsay-II speech understanding system (in its final configuration) was sufficient to perform the pilot experiments, but was not appropriate for more extensive experimentation. Getting a large knowledge based system to turn over and perform creditably requires a flexible initial design but, paradoxically, this flexibility is often engineered out as the system is tuned for high performance" — making it inappropriate for extensive experimentation.

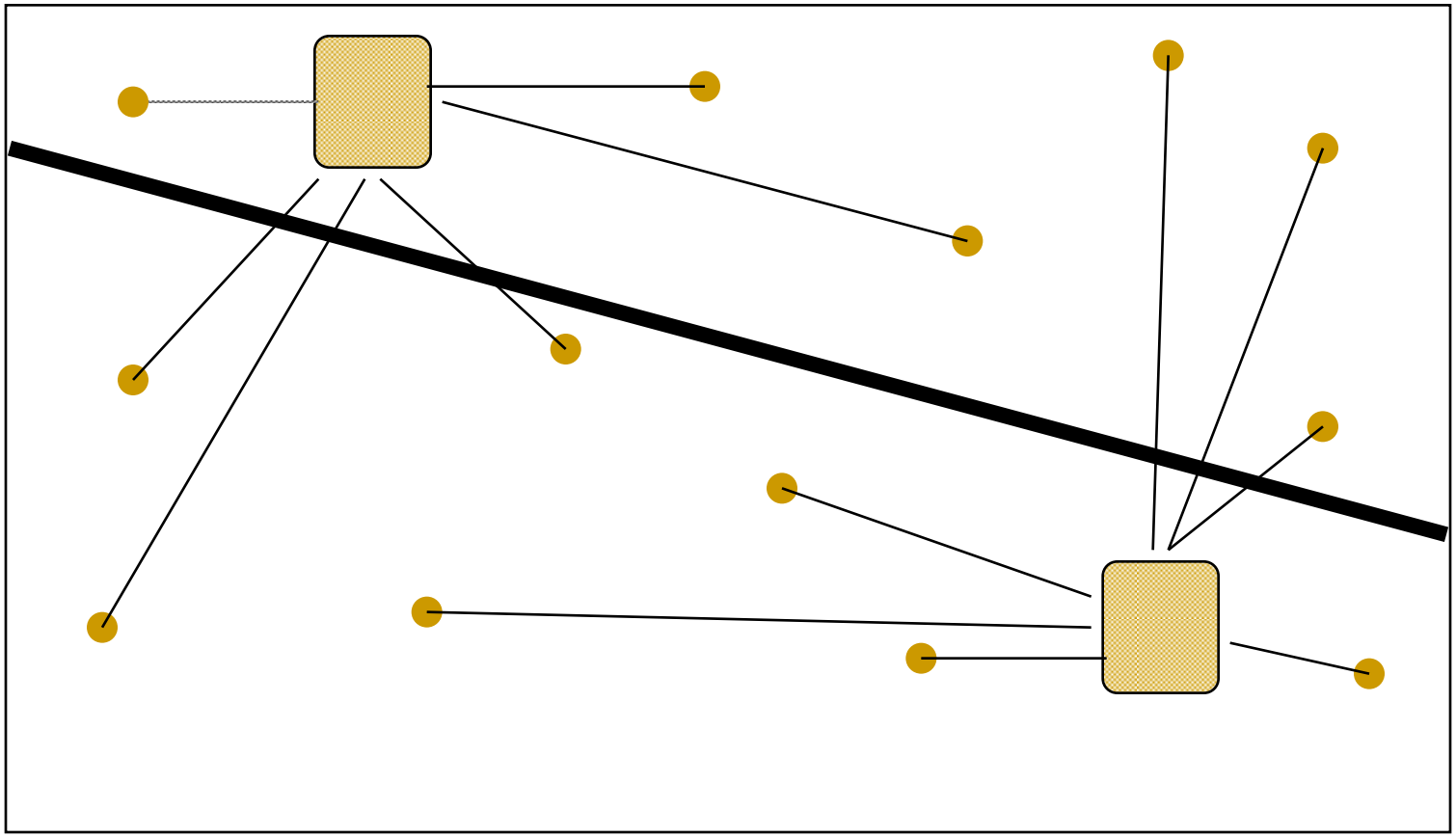
Approaches to Analysis

- On one side: Develop a clean analytic model (intuitions are lacking, however)
- On the opposite extreme: Examine a fully realistic problem domain (unsuited for experimentation, however)
- In the middle, a compromise: Abstract the task and simplify the knowledge (KS's), but still perform a detailed simulation of network problem solving

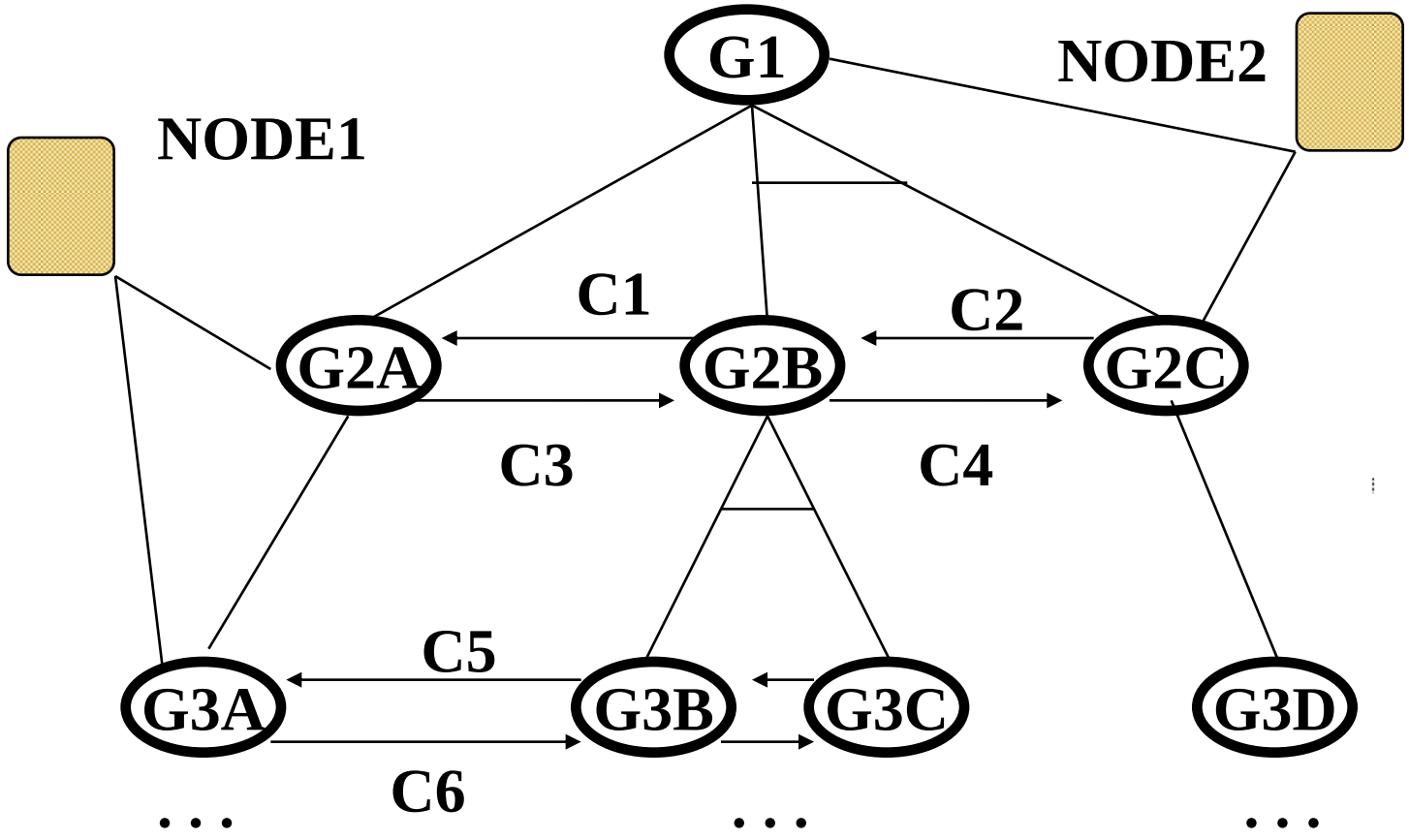
Distributed Vehicle Monitoring



Distributed Interpretation



Distributing the Problem Structure



Why this Domain?

1. A natural for distributed problem solving: geographic distribution of incoming data, large amounts of data (that argues for parallelism)
2. Information is incrementally aggregated to generate the answer map — the generation is "commutative" (actions that are possible remain permanently possible, and the state resulting from actions is invariant under permutations of those actions), making the job easier

Why this Domain?

3. The complexity of the task can be easily varied (increasing density of vehicles, increasing similarity of vehicles, decreasing constraints on known vehicle movement possibilities, increasing the amount of error in sensory data,…)
4. Hierarchical task processing levels, together with spatial and temporal dimensions, allow a wide variety of spatial, temporal, and functional network decompositions

Major Task Simplifications (partial)

- Monitoring area is a two-dimensional square grid, with a discrete spatial resolution
- The environment is sensed discretely (time frame) rather than continuously
- Frequency is discrete (represented as a small number of frequency classes)
- Communication from sensor to node uses different channel than node-to-node communication
- Internode communication is subject to random loss, but received messages are received without error
- Sensor to node communication errors are treated as sensor errors

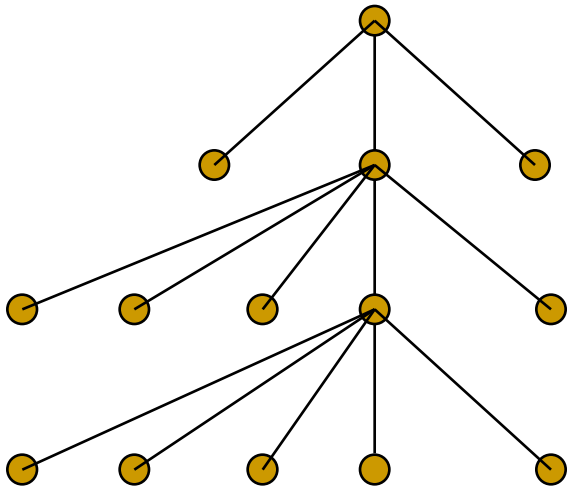
Parameterized Testbed

- The built-in capability to alter:
  - which KS's are available at each node
  - the accuracy of individual KS's
  - vehicle and sensor characteristics
  - node configurations and communication channel characteristics
  - problem solving and communication responsibilities of each node
  - the authority relationships among nodes

Node Architecture in DVMT

- Each node is an architecturally complete Hearsay-II system (with KS's appropriate for vehicle monitoring), capable of solving entire problem were it given all the data and used all its knowledge
- Each node also has several extensions:
  - communication KS's
  - a goal blackboard
  - a planning module
  - a meta-level control blackboard

Task Processing Levels



vehicle patterns

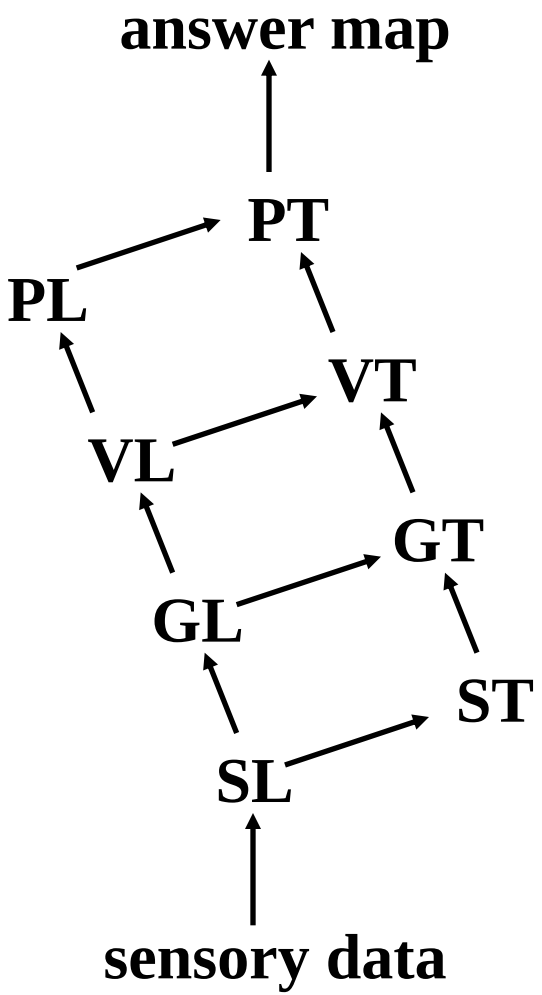
vehicles

signal groups

signals

Each of these 4 groups is further subdivided into two levels, one with *location hypotheses* (representing a single event at a particular time frame), and one with *track hypotheses* (representing a connected sequence of events over contiguous time frames).

Blackboard Levels in the Testbed



pattern track
pattern location
vehicle track
vehicle location
group track
group location
signal track
signal location

Goal Processing

- Goal-directed control added to the pure data-directed control of Hearsay-II, through the use of a goal blackboard and a planner:
  - Goal blackboard: basic data units are goals, each representing an intention to create or extend a hypothesis on the data blackboard
  - Created by the blackboard monitor in response to changes on the data blackboard, or received from another node
  - Can bias the node toward developing the solution in a particular way

The Planner

- The planner responds to the insertion of goals on the goal blackboard by developing plans for their achievement and instantiating knowledge sources to carry out those plans
- The scheduler uses the relationships between the knowledge source instantiations and the goals on the goal blackboard to help decide how to use limited processing and communication resources of the node

Communication KS's

- Hypothesis Send
- Hypothesis Receive
- Goal Send
- Goal Help
- Goal Receive
- Goal Reply

How to organize the work?

- “We believe that development of appropriate network coordination policies (the lack of which resulted in diminished network performance for even a small network) will be crucial to the effective construction of large distributed problem solving networks containing tens to hundreds of processing nodes.”
- So…how does one get “coherent cooperation”?

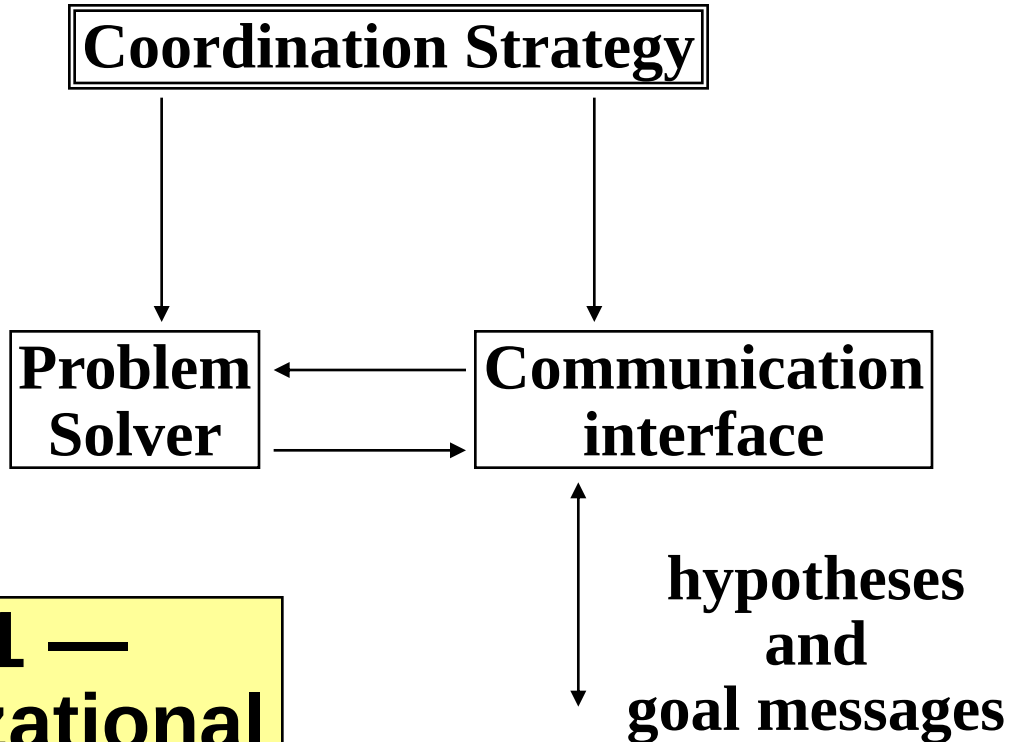
Coherence

- Node activity should make sense given overall network goals
- Nodes:
  - should avoid unnecessary duplication of work
  - should not sit idle while others are burdened with work
  - should transmit information that improves system performance (and not transmit information that would degrade overall system performance)
    - since nodes have local views, their contribution to global coherence depends on good local views of what's going on

Overlapping nodes

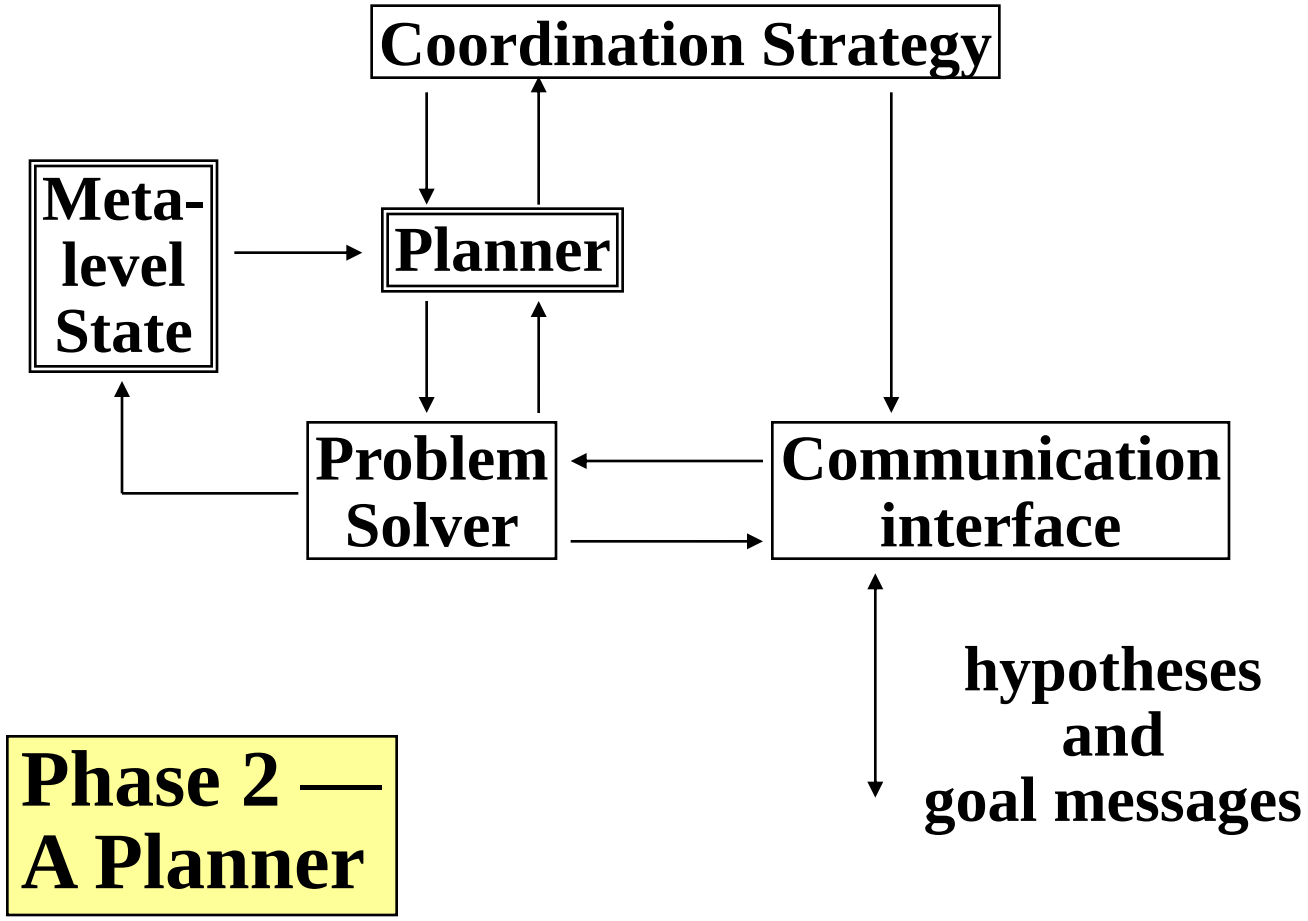
- Nodes often have overlapping views of a problem (intentionally, so that solutions can be derived even when some nodes fail) — but overlapping nodes should work together to cover the overlapped area and not duplicate each other's work
- Issues:
  - precedence among tasks (ordering)
  - redundancy among tasks (to be avoided)
  - timing of tasks (timely exchange of information can help prune search space)

Increasingly sophisticated local control

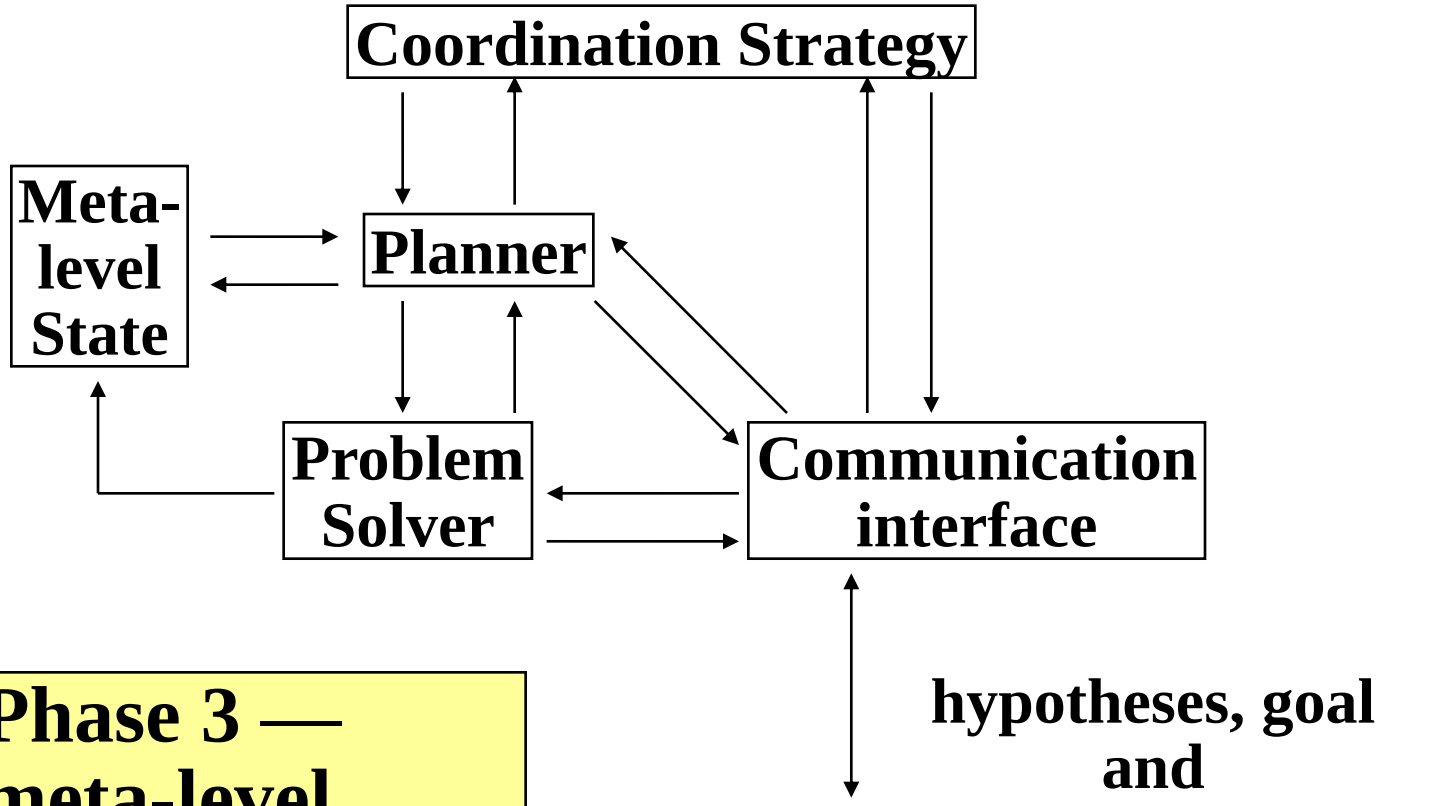


Phase 1 — organizational structure

Increasingly sophisticated local control



Increasingly sophisticated local control



Phase 3 — meta-level communication

Three mechanisms to improve network coherence:

1. Organizational structure, provides long-term framework for network coordination
2. Planner at each node develops sequences of problem solving activities
3. Meta-level communication about the state of local problem solving enables nodes to dynamically refine the organization

Organization

- Options (examples):
  1. Nodes responsible for own low-level processing, exchange only high-level partial results (e.g., vehicle tracks)
  2. Unbiased (treat locally formed and received tracks equally)
  3. Locally biased (prefer locally formed hypotheses)
  4. Externally biased (prefer received hypotheses)

Organization (continued)

5. Roles of nodes (integrator, specialist, middle manager)
6. Authority relationships between nodes
7. Potential problem solving paths in the network
8. Implemented in the DVMT by organizing the interest area data structures

Planning

- Given a low-level hypothesis, a node may execute a sequence of KS's to drive up the data and extend the hypothesis
- The sequence of KS's is never on the queue at the same time, however, since each KS's precondition has only been satisfied by the previous KS in the sequence
- Instead, a structure called a plan explicitly represents the KS sequence

A Plan

- A representation of some sequence of related (and sequential) activities; indicates the specific role the node plays in the organization over a certain time interval
- To identify plans, the node needs to recognize high-level goals — this is done by having an abstracted blackboard (smoothed view of data blackboard), and a situation recognizer that passes along high-level goals to the planner

Meta-level communication

- Information in hypothesis and goal messages improves problem-solving performance of the nodes, but does not improve coordination between them
- Messages containing general information about the current and planned problem solving activities of the nodes could help coordination among nodes. More than domain-level communication is needed…

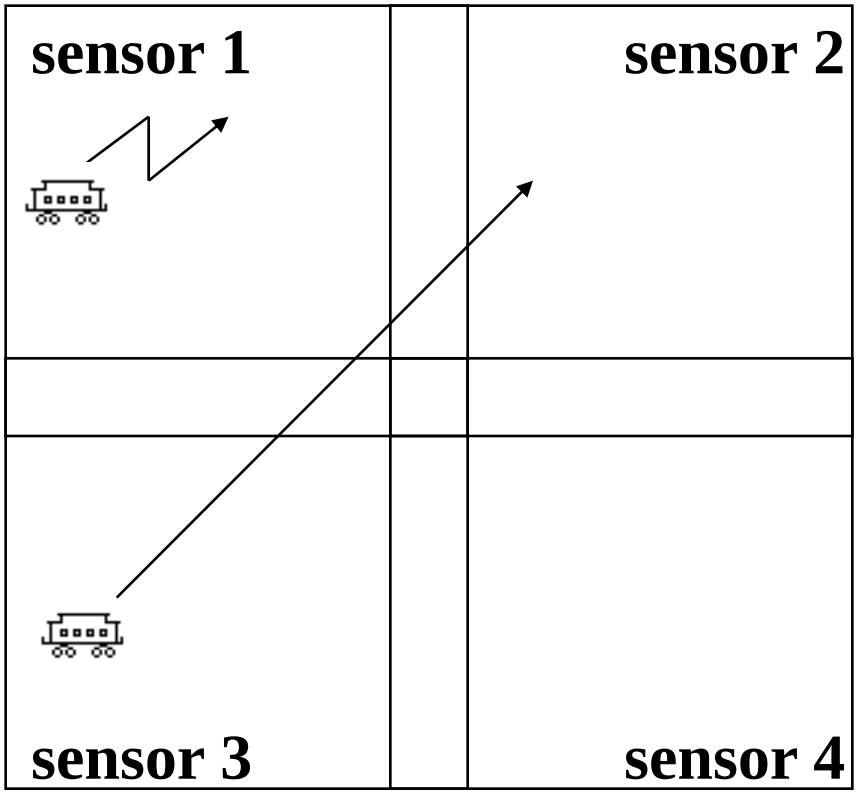
Partial Global Plans (PGP)

- A data structure that allows groups of nodes to specify effective, coordinated actions
- Problem solvers summarize their local plans into node-plans that they selectively exchange to dynamically model network activity and to develop partial global plans
- They enable many different styles of cooperation

How nodes work together

- Sometimes nodes should channel all of their information to coordinating nodes that generate and distribute multi-agent plans
- Sometimes should work independently, communicating high-level hypotheses (FA/C)
- Sometimes nodes should negotiate in small groups to contract out tasks in the network
- PGP is a broad enough framework to encompass all these kinds of cooperation

Distributed Vehicle Monitoring



Node Plans

- The node has local plans based on its own knowledge and local view
- The node's planner summarizes each local plan into a node plan that specifies the goals of the plan, the long-term order of the planned activities, and an estimate of how long each activity will take
- This, in turn, gives rise to a local activity map

Node Plans

- Node plans are simplified versions of local plans and can be cheaply transmitted
- The node's planner scans its network model (based on node plans that it has been receiving) to recognize partial global goals (like several nodes trying to track the same vehicle)
- For each PGG, the planner generates a Partial Global Plan that represents the concurrent activities and intentions of all the nodes that are working in parallel on different parts of the same problem (to potentially solve it faster) — also generates a solution construction graph showing how partial results should be integrated

Three types of plans

1. Local plan: representation maintained by the node pursuing the plan; contains information about the plan's objective, the order of major steps, how long each will take, detailed KS list
2. Node plan: representation that nodes communicate about; details about short-term actions are not represented, otherwise includes local plan data
3. PGP: representation of how several nodes are working toward a larger goal
   - Contains information about the larger goal, the major plan steps occurring concurrently, and how the partial solutions formed by the nodes should be integrated together

Authority

- A higher-authority node can send a PGP to lower-authority ones to get them to guide their actions in a certain way
- Two equal authority nodes can exchange PGP's to negotiate about (converge on) a consistent view of coordination
- A node receiving a node-plan or a PGP considers the sending node's credibility when deciding how (or whether) to incorporate the new information into its network model

A Node's Planner will…

1. Receive network information
2. Find the next problem solving action using the network model:
    1. update local abstract view with new data
    2. update network model, including PGP's, using changed local and received information (factoring in credibility based on source of information)
    3. map through the PGP's whose local plans are active, for each i) construct the activity map, considering other PGP's, ii) find the best reordered activity map for the PGP, iii) if permitted, update the PGP and its solution construction graph, iv) update the affected node plans
    4. find the current-PGP (this node's current activity)
    5. find next action for node based on local plan of current-PGP
    6. if no next action go to 2.2, else schedule next action
3. Transmit any new and modified network information