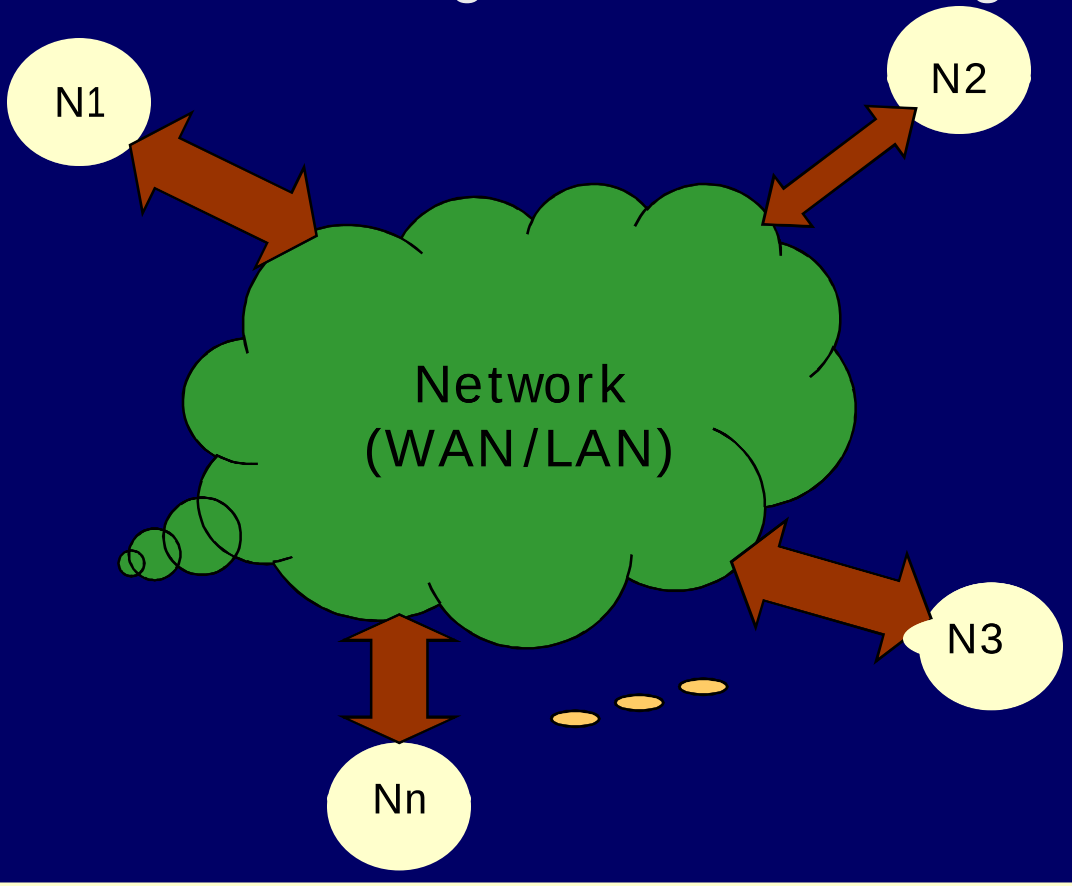
Distributed Real-Time Systems

What is distributed system?

- A set of nodes commun. through a network
- Network could be LAN or WAN
- Nodes could be homogeneous or heterogeneous



Why distributed systems?

- Applications themselves are distributed
  - E.g., command and control, air traffic control
- High performance
  - Better load balancing
- High availability (fault-tolerance)
  - No single point of failure

What are the problems with distributed systems?

- Resource management is difficult
  - No global knowledge on workload
  - No global knowledge on resource allocation
- No synchronized clock (or clocks need to be synchronized)
- Asynchronous nature of the nodes
- Communication related errors
  - Out of order delivery of packets, packet loss, etc.
- Difficult to distinguish network partition from node/link failures

System model

- The application is realized on a distributed system
- Tasks arrive at each node independent of other nodes
- Each node has resource manager for managing the workload at local node and for facilitating migration of workload to remote nodes
- Nodes cooperate among themselves for meeting tasks’ deadlines

Workload assumptions

- Periodic tasks and aperiodic tasks
- Periodic messages and aperiodic messages
- Task may have precedence constraints, resource and FT requirements
- The commn. pattern among two communicating periodic tasks is also periodic
- Two communicating tasks could be scheduled on two different nodes
- Meeting tasks deadlines require bounding and meeting message deadlines

Resource management in Distributed RT systems (Node architecture)

- Local scheduling
  - Resource management within a node
  - Task scheduling, resource reclaiming, etc. (issues discussed in chapters 2-4)
- Global scheduling
  - Balancing load across nodes
  - Transfer policy, selection policy, information policy, and location policy
- Communication resource management
  - QoS routing (channel setup time)
  - Resource reservation (channel setup time)
  - Packet scheduling (run-time)

Global scheduling

- Goal: migrate tasks from a local node (when it is heavily loaded) to a lightly loaded node
- Transfer policy: when tasks are to be migrated from/to local node to/from remote nodes
- Selection policy: which tasks are to be migrated
- Location policy: where tasks are to be migrated
- Information policy: what information is exchanged among nodes to realize task migration

Transfer policy

- Load index: the quantitative measure of node’s load
  - Non-real-time systems: queue length, processor utilization
  - Real-time systems: processor utilization, tasks’ laxity/deadline
- Transfer policy determines whether the current node is suitable to participate in a task migration either as a sender or as a receiver
- Threshold-based load index
  - Two thresholds (L-upper and L-lower) based on which a node’s load is classified as Light, Normal, or Overload
  - Light load implies the node could be a receiver for task migration
  - Heavy load implies the node is a sender for task migration
  - Normal load implies neither sender nor receiver
  - Fixing thresholds is hard

Transfer policy (contd.)

- Relative load index
  - The load of a node in relation to system’s average load
  - If node’s load > SysAvgLoad + delta, the node is overloaded; otherwise it is under-loaded
  - Average load could be misleading

Selection policy

- Once transfer policy determines the current node is the sender of a task migration, selection policy decides which tasks to migrate

- While choosing the tasks, following needs to be considered
  - End-to-end delay: sum of local decision time, migration time, remote decision time, and task's execution time must be less than task's deadline
  - Task's affinity to node – e.g., the required resource must be available at the remote node
  - Task's "value" – it is better meet deadlines of higher value offering tasks

Location policy

- Choosing the receiver node for a task migration

- There are several policies possible

  - Random policy – select the receiver randomly

  - Polling policy – poll the potential receivers of their load in sequential or parallel

  - Information based – based on the information provided by the information policy

Information policy

- Nodes exchange state info so as to obtain global state

- Demand-driven policy
  - A node collects state info from other nodes when it becomes a sender or receiver for task migration
  - Depends on node’s load state change to Light or Heavy

- State-driven policy
  - Whenever node’s load state changes, it informs other nodes
  - Similar to other demand-driven

- Periodic policy
  - Nodes periodically exchange state info irrespective of their states

Application

- in all embedded technology based electronic equipments which is timer based.

Scope of Research

1. Real-Time CORBA
2. Safety of Data in Real-Time Distributed Systems
3. Designing and debugging real-time distributed systems