1.1.2 END-TO-END JOBS AND TASKS

In practice, a system function is often provided by a set of related jobs, that is, a task. The jobs in each task may have precedence constraints.

A task in real time of a monitor system consists of three jobs: sampling, encoding, and processing the reading of a sensor on a field processors; sending the sensor data by a communication processor to the central control processor; and correlating and displaying the data with other sensor data on the control processor.

1.1.2.1 JOB SHOPS AND FLOW SHOPS

Concurrency in a multiprocessor system arises as jobs of different tasks sequencing through different processors in a pipeline manner. The classical jobs shop and flow shop model captures this types of concurrency. According to the job shop model, each task $T_1$is a chain of n(i) jobs denoted by $J_{i,k}$ for k = 1, 2, 3, . . . n(i) for all $1 \leq k \leq n(i)$. The adjacent jobs $J_{i,k}$ and $J_{i,k+1}$ on the chain execute on different processors $J_{i,k+1}$ become ready for execution only when $J_{i,k}$ completes. For example, the real-time monitor task is a chain of three jobs.

We can specify the processors on which n(i) jobs in each task T_i execute by the visit sequence $V_i = (V_{i,1}, V_{i,2}, \ldots, V_{i,n(i)})$ of the task. The k^{th} entry $V_{i,k}$ in this sequence gives the name of the processors on which the job $J_{i,k}$ executes. Therefore, the visit sequence of the real time monitor task is (field processor, communication processor, and control processor).

For example, visit sequence $V_i = (P_1, P_2)$, $V_2 = (P_2, P_3, P_4, P_3, P_1)$ of tasks T_1 and T_2 in a system means that T_1 has two jobs $J_{1,1}$ on P_1 followed by $J_{1,2}$ on P_2. T_2 has five jobs and they execute in turn on P_2 then on P_3, P_4, P_3 and finally on P_1.

A flow shop is a special job shop in which all the task have the same visit sequence. For example, suppose that all the tasks in a real-time monitor system are similar to the monitor task mentioned above. Each task samples, processes, and displays the reading of a different sensor. The visit sequences of all the monitor tasks are (field processor, communication processor, control processor) if the system has only one processor of each the three type.

1.1.2.2 END TO END TIMING CONSTRAINTS

The job shop model of multiprocessor systems we adopt here differs from the classical model in two ways. The first difference is a substantive one. The classical job shop model assumes that all tasks are ready for execution at the same time. Typically, the objective of classical job shop scheduling is to maximize the throughput (i.e. the number of tasks completed per unit time) of the system or to minimize the average response time of the tasks. In contrast, our tasks have arbitrary release times and deadlines, and some tasks have hard deadlines. Meeting hard deadlines is always our primary objective.

The timing constraints that can be derived directly from the high-level requirement of the applications are typically end to end in nature. They give the release time and deadline of each task as a whole.

Formally, we let the release time r_i of a task T_i in a job shop be the release time of the first job $J_{i,1}$ in the task. The deadline d_i of the task is the deadline of its last job $J_{i,n(i)}$. As long as the last job completes by the task's deadline, it is not important when the other jobs in the task complete. The executions of these jobs are constrained only by the dependencies between them and by the fact that they must complete sufficiently early to allow the on-time completion of the last job. Because the timing constraints of such a task are imposed on the jobs at the two ends of the task, we call them end-

Here, J2 is directly blocked by J4 when J2 requests dotted at time 4. J1 is directly blocked by J3 when J1 requests black at time 3. The global critical section of J2 on P2 is delayed by the global critical section of the lower priority J1 in the time interval (7, 11]. The preemption delay thus suffered by J2 must be taken into account when the schedulability of J2 is to be determined. This delay is treated as J2's blocking time.

At time 11, J1 exits from its critical section. Its priority is lower than the priority of the global critical section of J2. As a result, J2 preempts J1 on P2 in the interval (11, 12]. The total delay experienced by a job as a result of preemption by global critical sections of lower priority jobs is also a factor in the total blocking time of the job.

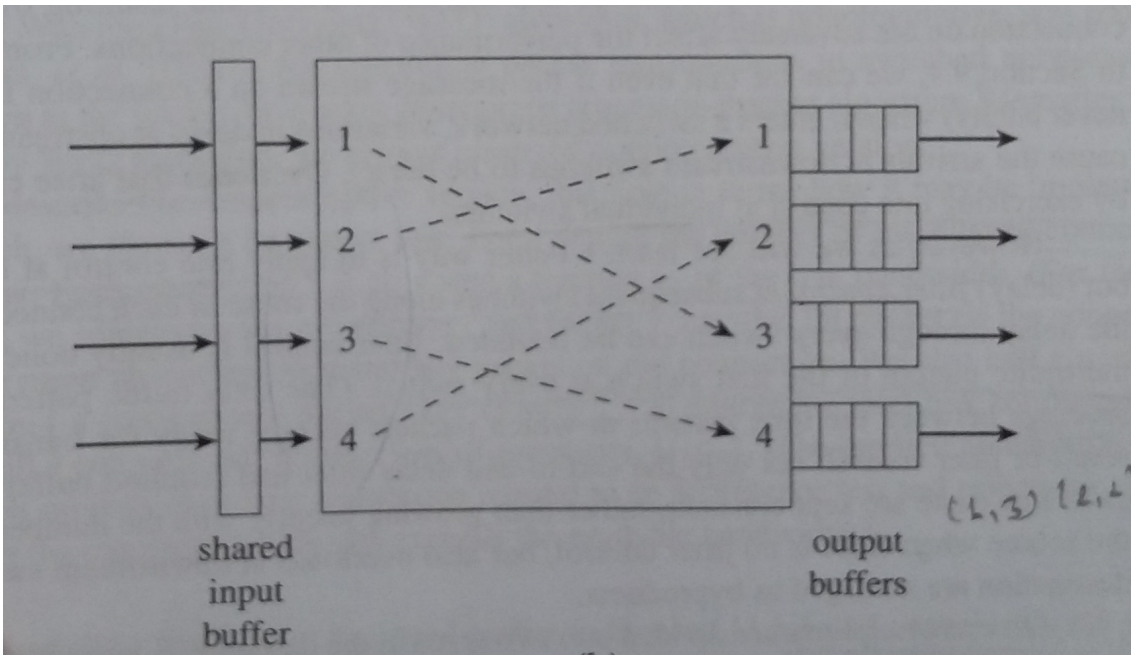
Finally, a job can be delayed by a local higher priority job whose execution is suspended on the local processor when the priority job executes on a remote processor. This delay is another factor of the blocking time of the job. Here in this example, J2 is suspended at time 7. As a result, it is still not completed in (13, 14] and J5 released at time 13.2 cannot start until time 14.

1.4 Elements of Scheduling Algorithms for End-to-End Periodic Tasks

End to end scheduling has two essential components:

1. Protocols for synchronizing the execution of siblings subtasks on different processors so that precedence constraints among subtask are maintained.
2. Algorithms for scheduling subtasks on each processor.

According to the end to end scheduling approach, the fact that no subtasks in the system ever requires remote resource makes it possible for the scheduler on each processor to use any on uniprocessor scheduling algorithms and resource access protocols to schedule subtasks on the processor and control



Service Discipline

The combination of an acceptance test and admission control protocol, and synchronization protocol and a scheduling algorithm used for the purpose of rate control, jitter control and scheduling of packets transmission is called a service discipline. In essence, rate control and jitter control serve the purpose of flow-control for real-time traffic. Flow control is applied at endpoints and within the communication network to prevent an entity upstream from overloading entities down-stream. (Up- and downstream follow the convention of river flow. The direction of traffic flow away from the source is downstream.) Traditional flow-control schemes, such as the sliding-window protocol are inappropriate for real-time traffic for many reasons. For example, they can introduce large variations in the flow rate and delay. Rate and jitter control mechanisms for real-time traffic are integral parts of a service discipline, rather than being separated from scheduling and buffer management functions. They differ fundamentally from the traditional flow-control mechanisms for data traffic.

Specifically, the term rate control is used in the communications literature to mean load management; the purpose is to ensure that the bursty traffic and resulting overloads on any connection do not adversely affect the performance of other connections. If the message stream on a connection is periodic (i.e., never bursty) when it enters a switched network, variations in delay at upstream switches may cause the arrivals at downstream switches to be bursty. Overloads that arise can be managed by exercising rate control at individual switches. However a better way is to apply rate control at the first switch but (delay) jitter control at subsequent switches along the route of each connection whenever the delay through every switch can be bounded. Jitter control is usually done by preserving the traffic pattern of the first switch at every switch. (The term traffic pattern refers to the spacing between the time instants at which packets become ready for transmission.) As a result of jitter control, not only the end-to-end delay jitter and required buffer space at each switch en route are kept constant, rather than growing linearly with the number of hops from the source when there is no jitter control, but also overloads at downstream switches and the destination are managed as byproducts.

Service disciplines are divided into two types in the literature.

- rate-allocating,
- rate-controlled

address of the destination station (or the ID of the message). When a station is not transmitting, it listens on the network. It copies the packet into its buffer (i.e., it receives the packet) when it hears its own address in the packet header (or the ID of a message it is prepared to receive). We focus here on the real-time aspects of the MAC protocols.

2.4.1 Medium Access Protocols in CAN and IEEE 802.5 Token Ring

We measure the size of a broadcast network in term of the ratio of the network round-trip delay to the transmission time of a maximum length packet. If this ratio is small (say on the order of 10^{-2} or smaller), every station can hear the transmission of every other station almost immediately after the transmission starts. We say that the network is small when this ratio is small. In a small network, circulating control information among the stations takes a small fraction of packet transmission time. The stations on the network can coordinate their decisions and actions without incurring significant performance penalty. By so doing, they can carry out a centralized scheduling algorithm in a distributed manner.

2.4.1.1 FIXED-PRIORITY SCHEDULING IN CAN

Controlled Area Networks are the examples of small networks. CANs are used to connect component of embedded controllers. For example, an automobile control system, whose components control the engine, the brakes, the environment and so on. At the transmission rate of one Mbits per second, the end to end length of a CAN must be no greater than 50 meters. It means, within a fraction of a bit-time after statin starts to transmit, all the stations on the network can hear the transmission. Therefore, the network functionally behaves like a local bus. The output of all the stations are wire-ANDed together by the bus: the bit on the network during a bit-time is a logical 0 if the output of any station is a 0 and a logical 1 only when the outputs of all stations are 1. The MAC protocol for CAN take advantage of this feature.

Each message stream transmitted in a network has a unique message ID. Each packet in the stream begins with this ID, with the most significant bit first. A station on the network determines whether to receive a packet based on the ID number of the packet. Finally, a packet contains 1 to 8 bytes of data.

CAN MAC protocol is a CSMA/CD (Carrier-Sense Multiple Access/Collision Detection) protocol. A station with a packet to send waits until it hears that the network is idle and then commences to transmit the ID number of the packet. At the same time the station listens. Whenever it hears a 0 on the network while it is transmitting a 1, it interrupts its own transmission. This way, network contention is resolved in favor of the packet with smallest ID among all contending packets.

So, the packets in each message stream are given a fixed priority that is equal to the ID of the message. The smaller the ID the higher the priority. Packets are transmitted non-preemptively based on their priorities.

2.4.1.2 PRIORITIZED ACCESS IN IEEE 802.5 TOKEN RINGS

In an IEEE 802.5 token ring network, packets are transmitted along a circular transmission medium in one direction. A station transmits a packet by breaking the network and placing its packets on the output link to the network. As the packet circulates around the network, the station identified by the destination address in the packet header copies the packet. When the packet returns to the source station, the station removes the packet from the network.

Polling

Network contention is resolved by a polling mechanism called token passing. For polling, each packet has an 8-bit Access Control (AC) field in its header. One of the bits in AC field is called the token bit. A station can determine whether the network is busy or free by examining tokenbit. As a polling

selects an output link for the message based on the information provided in the header. If the output link is free at the time, the header moves forward on that link to the next switch, leaving the input link it used to reach the current switch to the second flit in the message. Similarly, the third flit follows, using the link freed by the second flit, and so on. On the other hand, if an output link chosen for a message is in use, the header is buffered and waits at the switch until the output link becomes free. In the meantime, subsequent flits that have reached upstream switches occupy the flit buffers there, one per switch. The associated input links (i.e., output links of the corresponding upstream switches) are not available to other messages. We call this phase of message transmission the routing phase. The routing of a message starts when its header leaves the source processor and completes when the header reaches the destination processor. Hereafter, the message has all the links along the path between its source and destination. Each link is occupied by one of the message's flits. The flits shift downstream by one link (i.e., one switch) in each time step without intervention of the switches. Thus, the message "worms" its way nonpreemptively through the network without being queued at any switch. Its transmission completes when its last flit is delivered to the destination processor. In short, the message delay through a wormhole network is the sum of its routing time and transmission time. The latter is essentially the total propagation delay of all links on its transmission path. The time required to route a message depends on the overall network traffic and is the nonpredictable component of message delay.

2.7.1.2 Path Selection and Scheduling

To make the network scalable with the number of processors in the system, algorithms used to select paths for messages are typically simple. As an example, in a two-dimensional mesh network, each switch is connected to four neighboring switches. It is common to use one-bend paths for unicast messages. A one-bend path between two processors in a mesh consists of a segment of column links and a segment of row links. If the path of every message traverses a column segment first and then a row segment (or vice versa), there is no deadlock. There are equally simple, deadlock-free routing algorithms for high-dimensional interconnection networks.

Typically, algorithms used to schedule nonreal-time messages do not prioritize messages. They can be divided into two categories: greedy and throttling. According to a greedy algorithm, each message is routed along a deadlock-free path as soon as it is ready for transmission at the source. In contrast, according to an algorithm that does throttling, a message may wait at the source. This intentional delay to start routing serves a purpose similar to traffic shaping in packet-switched networks, that is, to reduce the worst-case delay.