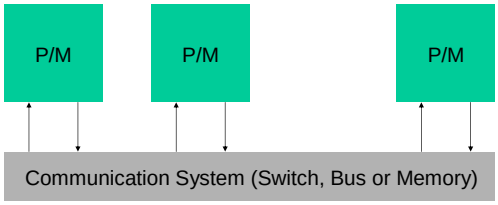
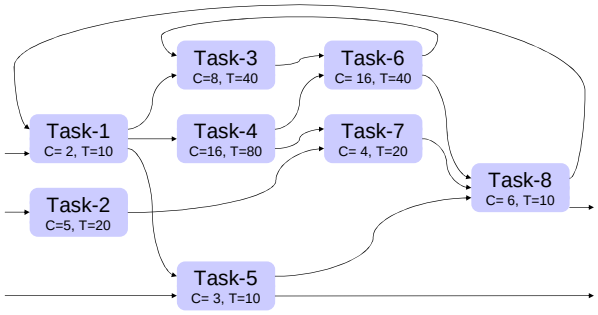
Physical (HW) System Structure



P/M = Processor or memory or both processor and memory

TASK Dependency Structure



Task dependencies

T-1 in{8} out{3,4,5}

T-2 in{7}

T-3 in{1,6} out{6}

T-4 in{1} out{6,7}

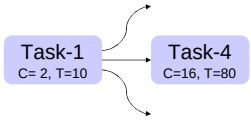
T-5 in{1} out{8}

T-6 in{3,4} out{3,8}

T-7 in{2,4} out{8}

T-8 in{5,6,7} out{1}

Different activation periods



Assumption:
A sending task (source) will produce a message in every period and send it individually or aggregated in a package according to the needs of the receiving task and its execution period. A slow receiver may read several messages at a time from its input buffer.

For example:
Task-4 will get eight individual or one aggregated/packaged message from Task-1 when activated.

Total Work Load

T-1 (C = 2, T = 10) U = 0.2

T-2 (C = 5, T = 20) U = 0.25

T-3 (C = 8, T = 40) U = 0.2

T-4 (C = 16, T = 80) U = 0.2

T-5 (C = 3, T = 10) U = 0.3

T-6 (C = 16, T = 40) U = 0.4

T-7 (C = 4, T = 20) U = 0.2

T-8 (C = 6, T = 10) U = 0.6

U_{tot} = 2.35
Requiring at least 3 Processors

Partitioning/clustering algorithm

A partitioning algorithm could work as follows:

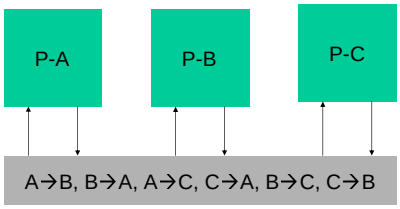
1. **calculate** the average utilization and use it as a pivot point for following decisions
2. **sort** tasks in decreasing size order
3. **find** the largest (i.e. the first) task remaining to be allocated to a partition (cluster) and make it to a task set
4. **if** sum of task set is larger than the pivot **return**(task set)
5. **else** add smallest task to task set and **go to 4**

(There are certainly better partitioning and bin packing methods but the problem is still in the class NP hard combinatorial)

Which allocation is best?

- Even load
- Even utilization
- Many internal connections, few external
- Many internal messages, few external
- Large internal bandwidth, little external

Assume TDMA communication



You will also have local communication between
tasks allocated to the same processor

Feasible task → processor allocation

- 8 different tasks and 3 processors of the same type and capacity
- All allocations are not feasible (to high utilization to enable scheduling) since the total processor utilization is quite high (2.35)
- In how many ways can eight tasks be allocated to three processors assuming that all three processors are used?

Task → Processor Allocation, cont.

- One can select three tasks out of the eight tasks to allocate to one processor in 56 different ways
- Three more of the remaining 5 tasks can be assigned to a second processor in 10 ways
- If the two remaining are allocated to the third processor we get 560 possible (but not necessary feasible) allocations
- We also have all possible combinations starting with six, five or four tasks on the first processor and …

Combinatory problems

The number of combinations without repetitions (the binomial coefficient):
is the number of ways that *k* objects can be chosen only once from among *n* objects, regardless of order

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

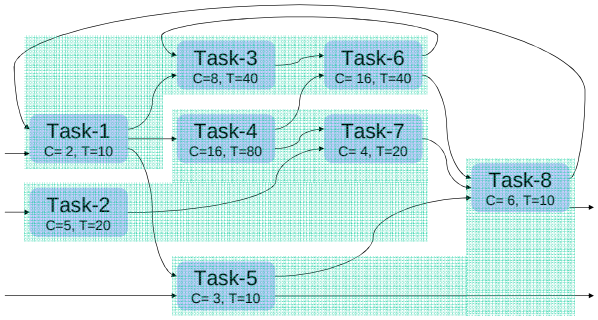
Example: 8 tasks on 3 processors

If choice of processor does not matter:
Place 6 of 8 on P1; 1 on P2 and 1 on P3
Place 5 of 8 on P1; 2 of 3 on P2 and rest on P3
Place 4 of 8 on P1; 3 of 4 on P2 and rest on P3
Place 4 of 8 on P1; 2 of 4 on P2 and rest on P3
Place 3 of 8 on P1; 2 of 5 on P2 and rest on P3

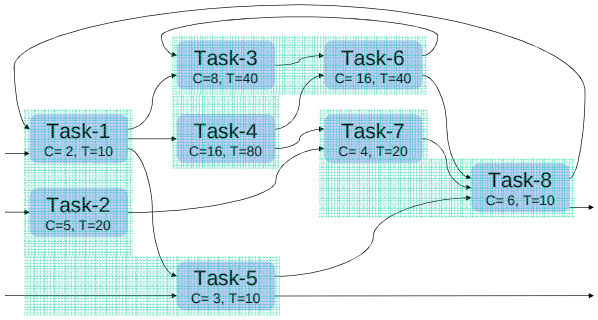
$$\binom{8}{6}+\binom{8}{5}\binom{3}{2}+\binom{8}{4}\left(\binom{4}{3}+\binom{4}{2}\right)+\binom{8}{3}\binom{5}{2}$$

This gives 28 + 56*3 + 70*(4+6) + 56*10 = **1456**

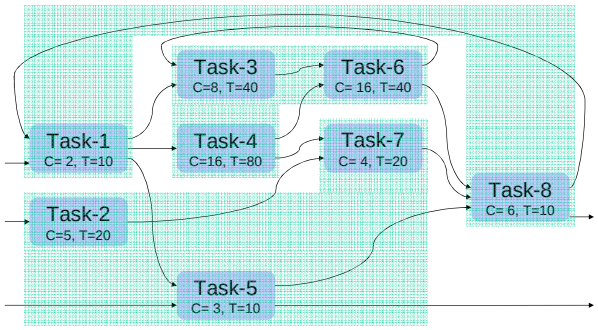
Clustering/Allocation, example 1



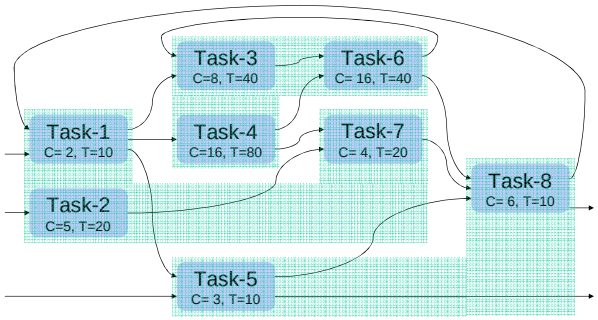
Clustering/Allocation, example 2



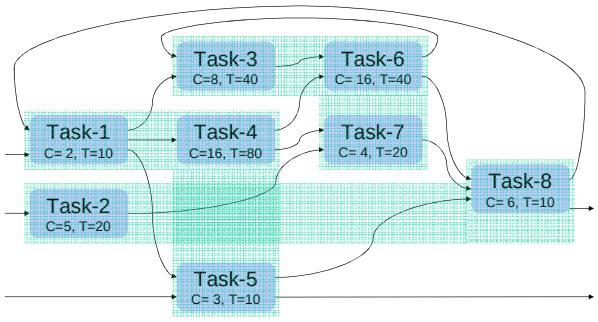
Clustering/Allocation, example 3



Clustering/Allocation, example 4



Clustering/Allocation, example 5



Cluster Evaluation (heuristic)

Assume: **Even utilization simplifies scheduling**

$U_{ratio} = U_{min}/U_{max}$ (where U = Utilization)

Assume: **Communication has a cost related to number of messages and links**

Give value to number of cluster internal messages

$M_{ratio} = \Sigma M_{internal}/M_{external}$ (where M = Message)

Give value to number of cluster internal links

$L_{ratio} = \Sigma L_{internal}/L_{external}$ (where L = Link)

Cluster Evaluation (heuristic)

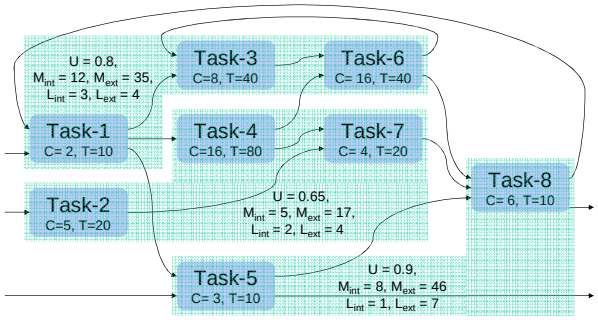
Calculate the cluster allocation value as:

Value-1 = $10*U_{ratio}^{2} * M_{ratio} * L_{ratio}$

Value-2 = $1000*U_{ratio} / (M_{ext} * L_{ext})$

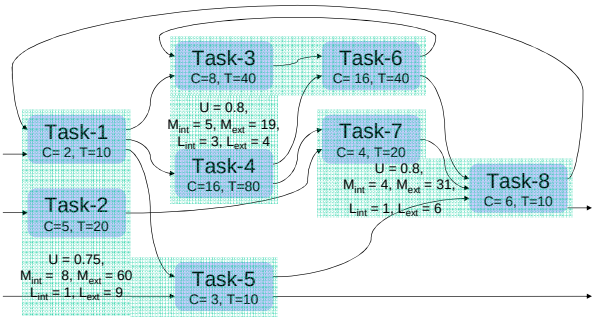
Value-3 = $U_{ratio}^{2} / (C_m*M_{ext}^{2} + C_l*L_{ext}^{2})$

Clustering/Allocation 1



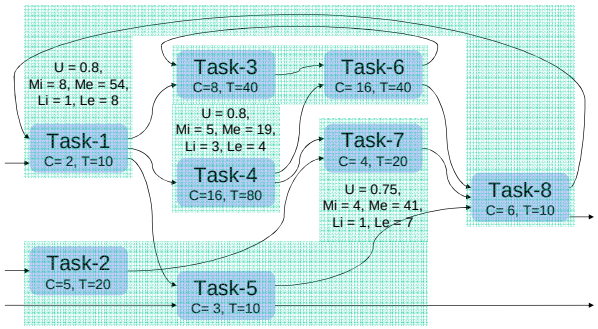
Value-1 = 0,532, Value-2 = 0,491 , Value-3 = 0,343

Clustering/Allocation 2



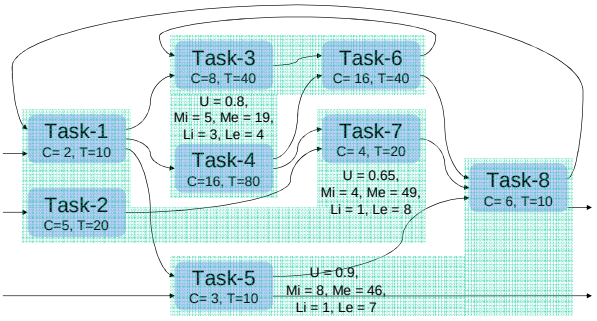
Value-1 = 0,340, Value-2 = 0,426, Value-3 = 0,398

Clustering/Allocation 3



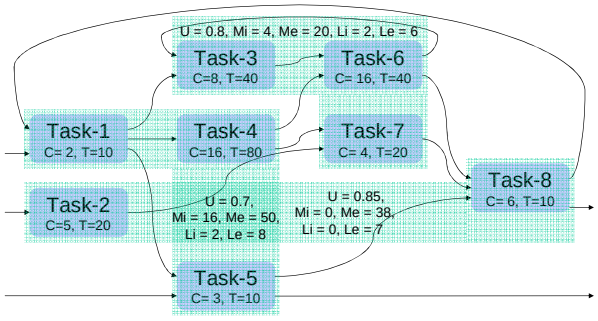
Value-1 = 0,345, Value-2 = 0,433, Value-3 = 0,399

Clustering/Allocation 4



Value-1 = 0,205, Value-2 = 0,333, Value-3 = 0,237

Clustering/Allocation 5



Value-1 = 0,299, Value-2 = 0,363, Value-3 = 0,299

Communication as Tasks

- To be more precise we should estimate a more realistic cost for:
  - internal communication
  - external communication
- Communication can be seen as tasks of their own or as extensions of existing tasks

Finishing up

- We take a few examples
- and then we repeat some concepts

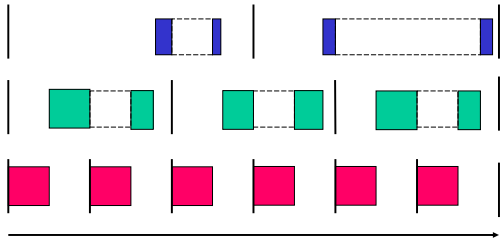
Example task set 1 {a, b, c}

Task	C	T	C/T
a	10	20	0,5
b	15	40	0,375
c	6	60	0,1

Utilization = 10/20+15/40+6/60 = **0,975**

$$U = \sum_{i=1}^{N} \frac{C_i}{T_i}$$

Schedule for task set 1



LCM = 120

Example of task set with harmonic periods

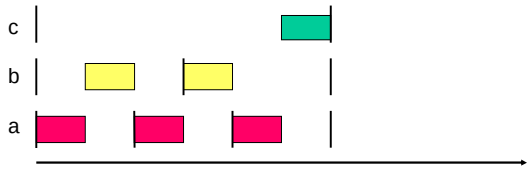
Example task set 2 {a, b, c}

Task	C	T	C/T
a	10	20	0,5
b	10	30	0,333
c	10	60	0,167

Utilization **1,00**

Schedule for task set 2

Task set is feasible



Example of task set with relatively harmonic periods

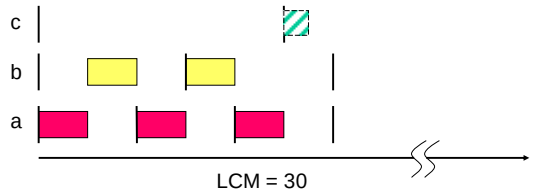
Example task set 3 {a, b, c}

Task	C	T	C/T
a	10	20	0,5
b	10	30	0,333
c	5	50	0,10
Utilization			**0,933**

Example of task set with inharmonic periods

Schedule for task set 3

No time for task c →
Task set is unfeasible
→ Deadline can't be meet

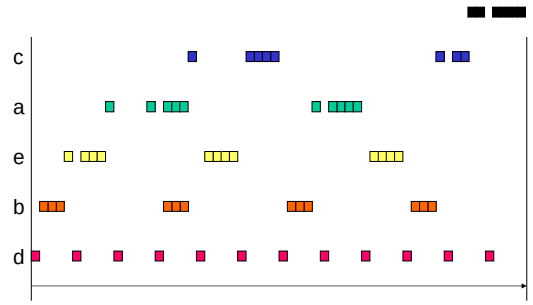


Example task set 4 {a, b, c, d, e}

Task	C	T	C/T
a	5	30	0,1667
b	3	15	0,20
c	8	60	0,1333
d	1	5	0,2
e	4	20	0,2
Utilization			**0,90**

Schedule for task set 4

Spare not utilized capacity



Important concepts and terms

Arrival (Release, Request)
A new job caused by an event, message or time point resulting in that a dormant or waiting task instance can be released and made ready to be served.

Blocking
Situation when a job is suspended due to a higher priority job or because it has to wait for a resource to be free.

Ceiling
The highest priority level associated with a resource or associated lock given by the highest priority task will need to reserve the resource.

Important concepts and terms

Channel
Communication resource (e.g., separated by space, time, frequency or code).

Computation Time
The effective execution time used by a processor to perform a job (task instance) from its activation until its completion.

Concurrent
Overlapping or parallel in time (but not necessarily simultaneous in the sense lock step synchronous).

Important concepts and terms

Context Switch
Activity (by scheduler) when a processor leave (suspend) one job (task instance) to work with another.

Critical Section
Piece of program describing a task that to perform some part of a job needs exclusive access to a shared resource. With exception for restricted interrupt service routines (ISR) it is not allowed to be preempted. A critical section is a form of lock.

Deadline
The time within which a real-time task instance must be finished to do a job correctly.

Important concepts and terms

Dispatching
The selection of which job to execute at the time for context switching.

Event
An occurrence or change in the external or internal execution context in which the real-time system works.

Interrupt
Event causing an operating system or a processor to suspend the ongoing job and activate an interrupt service routine.

Interrupt Service Routine
A small dedicated program that handles what must be done with high urgency at the occurrence of an interrupt. It usually ends its work by releasing a related task instance.

Important concepts and terms

Job
A piece of work done for example when executing a task instance that is activated by an internal or external message or event or at scheduled (periodic) time point(s).

Lateness
Time remaining (that need to be used) to finish a job (task instance) after its deadline. Has negative value if early.

Offset (Phase)
The delayed start of (or phase difference between) one periodic task in relation to other periodic tasks.

Overload
Situation when a computer or its operating system can not handle all arriving or activated jobs.

Important concepts and terms

Period
Time between individual job activation's of a periodic task. (Often defined by needed sample rate).

Port
Source (out port) or sink (in port) used to send or receive events or messages between tasks.

Preemption
The suspension of a task instance when it has used its time slot or when an event, message or job with higher priority has arrived.

Important concepts and terms

Priority
Importance or value given (static or dynamic) to an event, message, job, task instance or resource.

Process
1) A structure of related real world events or activities.
2) A task that has its own protected memory area (thus not has its task instance state data in a memory shared by other tasks).

Response time
The total elapsed time needed to complete a job.

Important concepts and terms

Semaphore (**flag or mutex**)
Data bit or structure used to tell if a resource is free or reserved.

Slack (**Laxity**)
Time margin between arrival, computation and deadline. Slack = D – C - A

Tardiness
The time a task remains active after its deadline. (Tardiness can not be negative.)

Important concepts and terms

Task
A set of activities to do in a way described by a program triggered by external or internal events or periodically by a clock driven schedule and that has references to its own data and execution control state.

Task instance
The state and data context of a task, needed to perform a unique individual job such as an instance of type phone call, media session or other service invocation.

Thread
Operating system concept. Data structure used to implement the control mechanisms need to handle a task instance.

Important concepts and terms

Throughput
The amount of job a computer can do in a period of time.

Utilization
The fraction of the total time that a processor, a task or other resource is used.

Value Density
Ratio between (application) value and computation time of a task.

Embedded Systems Constraints

Timing - when must a task be done

Precedence - must some tasks be done before other tasks

Sharing - to exclude conflicts between shared resources

Locality - closeness to sensors and actuators to reduce communication cost and measurement noise

Throughput - how many tasks can be handled during some period of time

Fault tolerance - require redundancy such as replication of task data

Communication - is costly – keep strongly connected tasks together

Scalability - distribute incoming load (evenly) on several processors

Power or energy - enable mobile (battery driven) operation

Weight or size - impact on total system performance

Cost and value - what cost does the market value of the product motivate

Response time distribution

- The possible response time outcome for a task can be described by a probability distribution function