# “Resource Scheduling in Dependable Integrated Modular Avionics”

by Yann-hang Lee, Daeyoung Kim, Mohamed Younis, Jeff Zhou, James McElroy

# Outline

- System model
- Scheduling approach
- Algorithm evaluation
- Conclusions

# Integrated Modular Avionics(IMA)

- Integration of mixed-criticality real-time applications
- For each Integrated application
  - Meet timing constraints
  - Share avionics computer resources

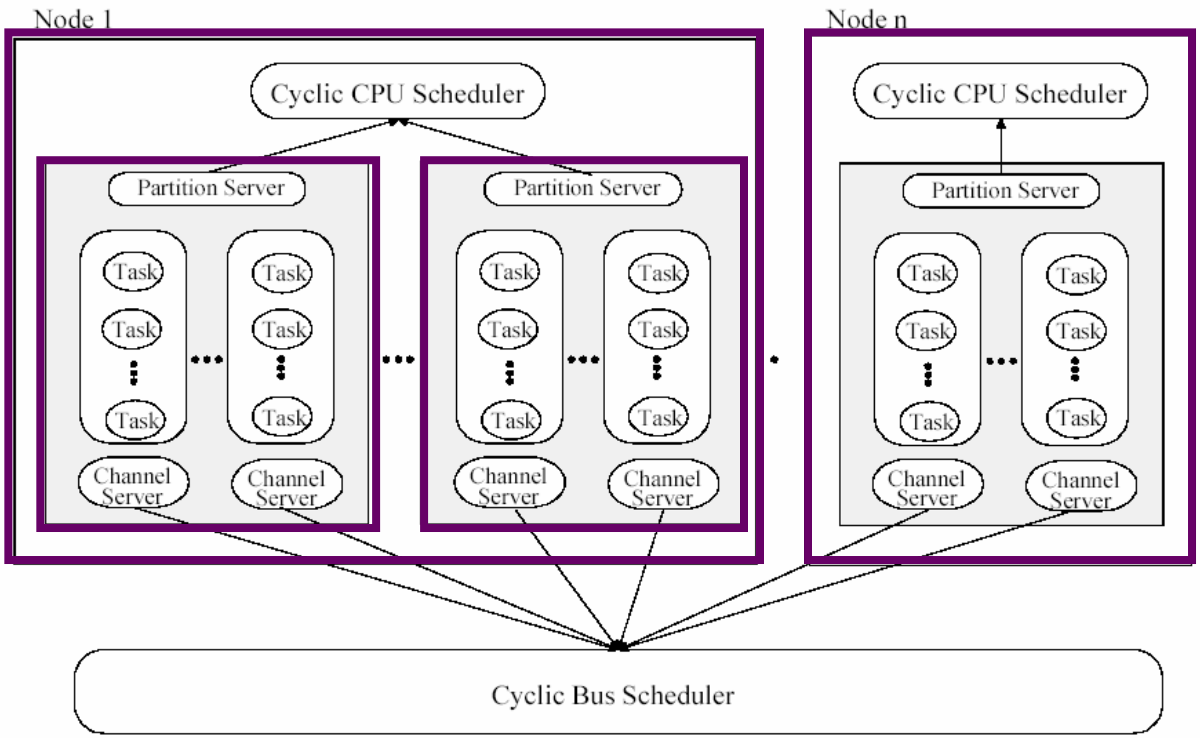
Spatial and temporal partitioning

# Strongly Partitioned Real-Time System(SP-RTS)

- Includes multiple processors
  - Inter-connected by a communication bus
- Each processor has several execution partitions
  - Communication channels are assigned to a subset of tasks running in a partition

This paper investigated the issues related to the partition and channel scheduling in SP-RTS

# System Model



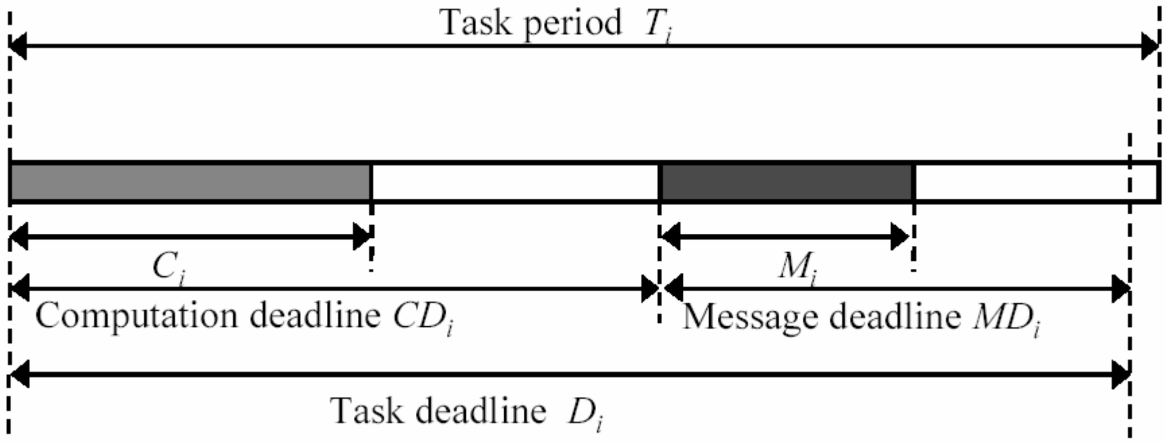

- Processors
- Partitions
- Applications
- Tasks

The architecture model for strongly partitioned real-time systems (SP-RTS)

# System Model——in task model

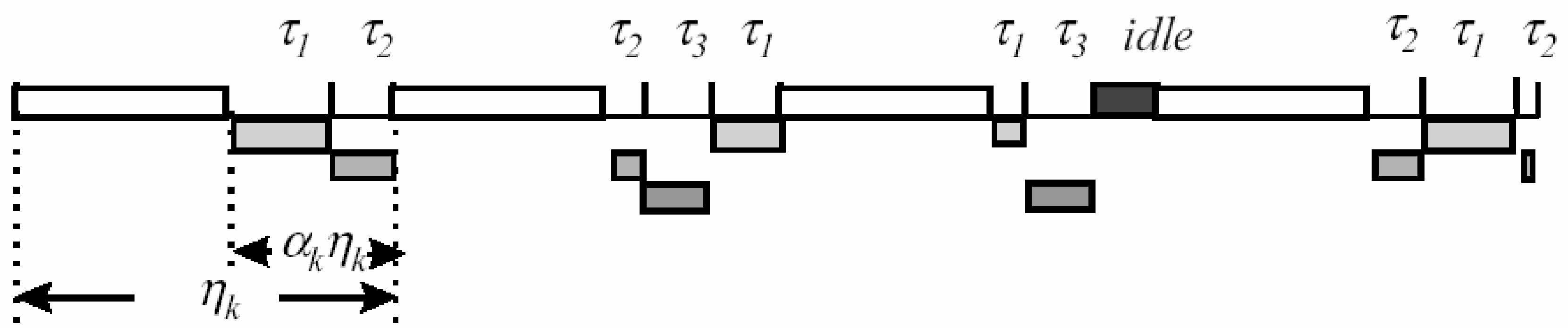
- Each task
  - arrives periodically
  - Must complete computation and send an output message before deadline
  - parameters:

invocation period ($T_i$), worst-case execution time ($C_i$), deadline ($D_i$), message size ($M_i$)



# System Model—in Processor Model

- The scheduling is done in a two-level hierarchy
  - Each partition server, $S_k$, is scheduled periodically with a fixed period--*partition cycle* $\eta_k$
  - In each partition cycle, the server can execute the tasks in the partition for an interval $\eta_k \alpha_k$ , $\alpha_k$ is *partition capacity* ($\alpha_k \geq 1$). For the remaining interval of $(1- \alpha_k)\ \eta_k$, the server is blocked



# System Model

- Message (preemptive, fixed-priority)

  message is the only form of communication among applications

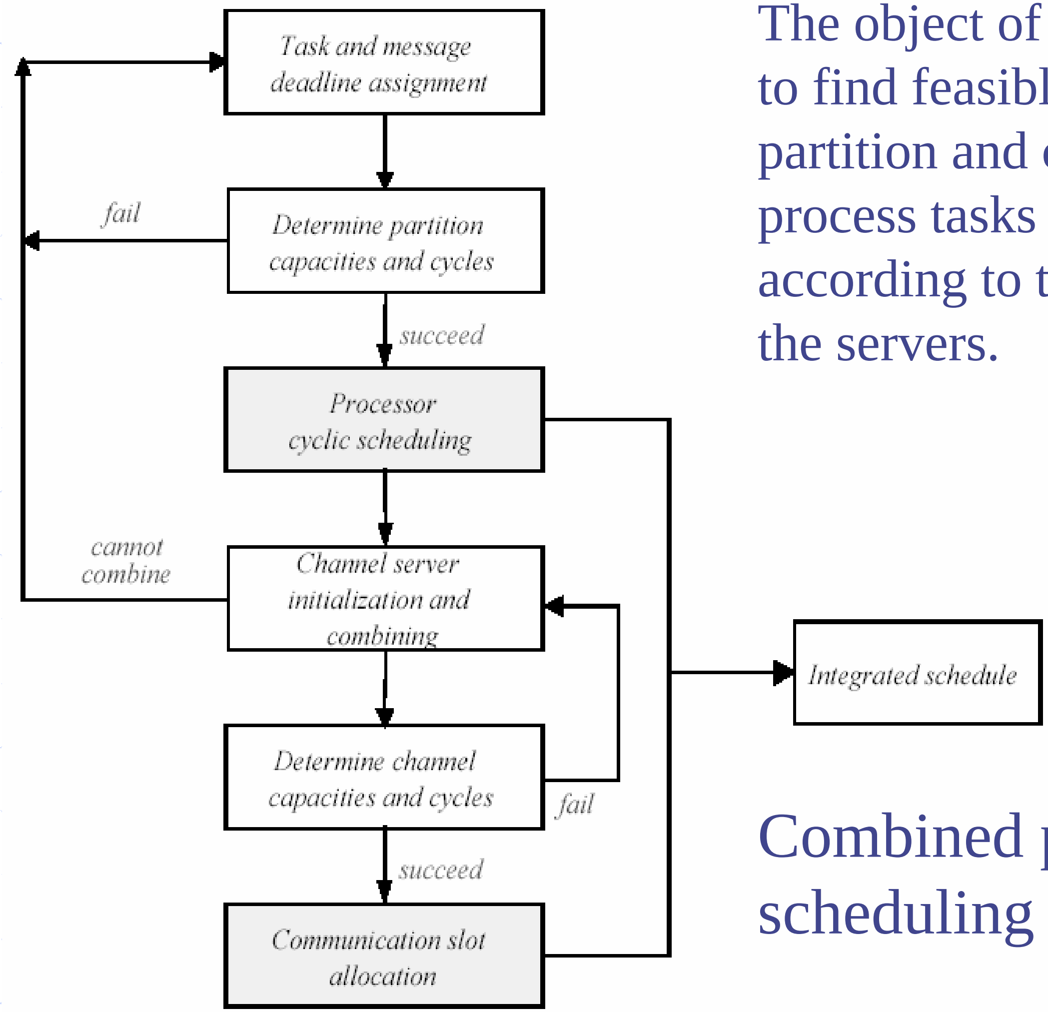
- Channel

  *channel cycle:* $\mu_k$ *partition cycle :* $\eta_k$

  *channel capacity:* $\beta_k$ *partition capacity:* $\alpha_k$

  a sequence of communication slots are assigned to each channel sever according to its channel cycle

# Scheduling Approach




The object of our scheduling approach is to find feasible cyclic schedules for partition and channel servers which process tasks and transmit messages according to their fixed priorities within the servers.

Combined partition and channel scheduling approach

# Scheduling Approach

- Deadline Decomposition
- Partition and Channel Scheduling
- Channel Combining
- Cyclic Scheduling for Partition and Channel Servers

# Deadline Decomposition

- A deadline decomposition algorithm is used to assign these deadlines in a heuristic way.
- *Di = CDi + MDi*

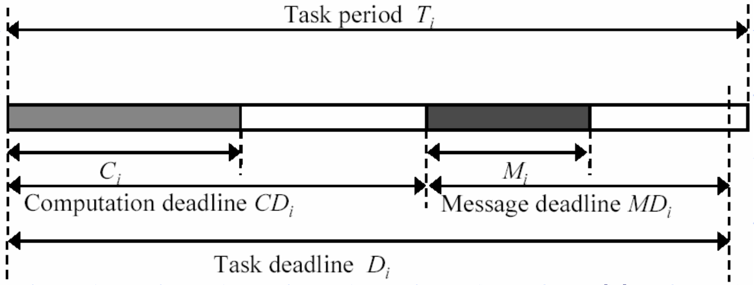
$$\text{Message Deadline, } MD_i = \left(D_i \frac{ST * M_i}{C_i + ST * M_i}\right) f_i$$

$$\text{Computation Deadline, } CD_i = D_i - MD_i$$

where *fi* is an adjusting factor for each task.

ST=slot-length / bus-bandwidth

*ST*Mi* is the transmition time of *Mi*



# Deadline Decomposition

lower bound and upper bound of the adjusting factor *f* :

$$\frac{1}{D_i\left(\frac{1}{C_i + ST * M_i}\right)} \le f_i \le \frac{D_i - C_i}{D_i\left(\frac{ST * M_i}{C_i + ST * M_i}\right)}$$

*CDi= Di-MDi,*
*MDi=ST*Mi*

*CDi= Ci,*
*MDi=Di-Ci*

Set the initial value $f_i = 1$

# Deadline Decomposition

- The deadline decomposition algorithm

Initialization for all tasks

$MinF = 1 / (D_i * (1/(C_k + ST*M_k)));$

$MaxF = (D_i - C_i) / (D_i * (ST*M_i /(C_i + ST*M_i)));$

$f_i = 1.0;$

Iterative change of $f_k$ when either partition or channel scheduling fails

*If (Partition scheduling fails) {*

$MaxF = f_i; f_i = (MinF + f_i) / 2.0;$

*}*

*else if (Channel scheduling fails) {*

$MinF = f_i; f_i = (MaxF + f_i) / 2.0;$

*}*

# Scheduling Approach

- Deadline Decomposition
- Partition and Channel Scheduling
- Channel Combining
- Cyclic Scheduling for Partition and Channel Servers

# Partition Scheduling

- Partitions are cyclically scheduled, ($\eta_k$, $\alpha_k$ )
- Partition, $P_k$, has n tasks
- Task level use DM schedule $(\tau_1 > \tau_2 > \ldots > \tau_n)$
- Task $\tau_i$ is schedulable if there exists a

  t $\in$ $H_i = \{CD_i \cup lT_j \mid j=1,2,\ldots,i\text{-}1;\ l=1,2,\ldots, \lfloor CD_i/T_j \rfloor\}$

  that:

$$W_i(\alpha_k, t) = \sum_{j=1}^{i} \frac{C_j}{\alpha_k} \left\lceil \frac{t}{T_j} \right\rceil \le t$$

- $W_i\ (\alpha_k,\ t)$ -- the worst cumulative execution time

  by tasks whose priority ≥ $\tau_i$ during $[0, t]$

# Partition Scheduling

- Define:

  level-i inactivity period

  $$B_i(\alpha_k) = max_{\; t \in Hi} \{t - W_i(\alpha_k, t)\}$$

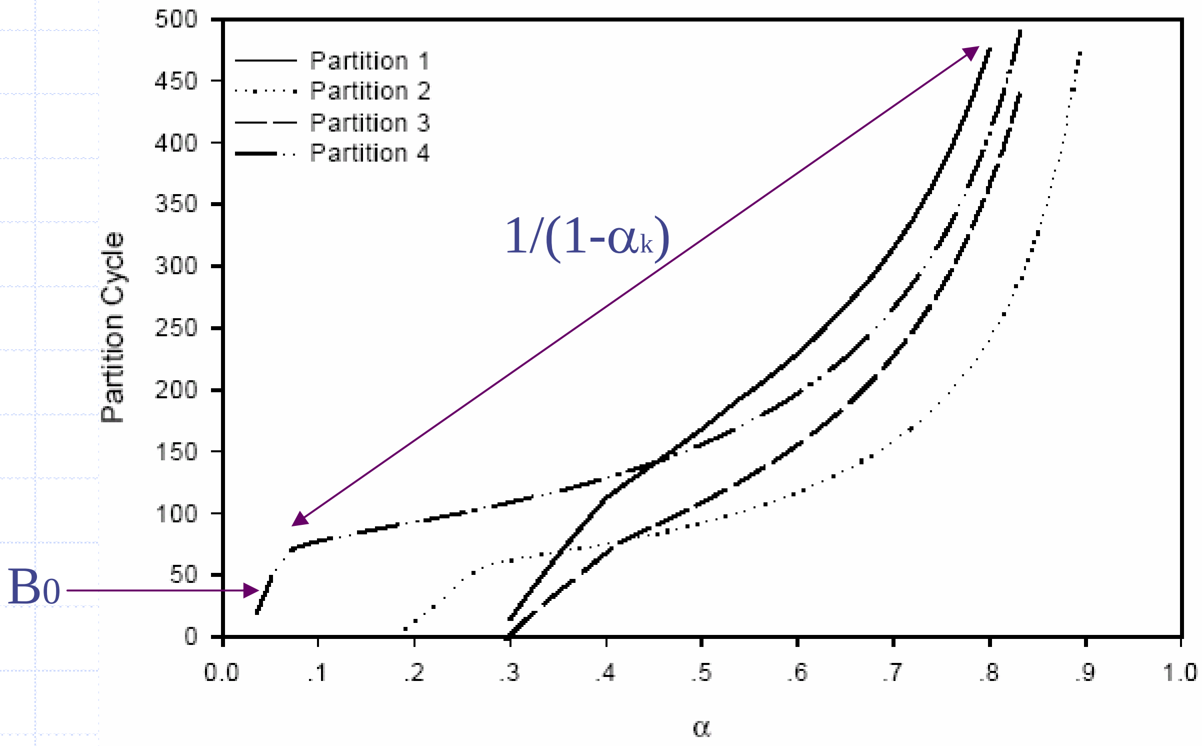
  let

  $$B_0(\alpha_k) = min_{\; i=1,2,..n} B_i(\alpha_k)$$

- ***Theorem 1.*** *The partition server $S_k$ is schedulable if $S_k$ is schedulable at a dedicated processor of capacity $\alpha_k$, and $\eta_k \leq B_0(\alpha_k)/(1-\alpha_k)$*

# Partition Scheduling--example

| | Partition 1 (utilization=0.25) | Partition 2 (utilization=0.15) | Partition 3 (utilization=0.27) | Partition 4 (utilization=0.03) |
|---|---|---|---|---|
| tasks ($C_i$, $T_i$) | (4, 100)<br>(9, 120)<br>(7, 150)<br>(15, 250)<br>(10, 320) | (2, 50)<br>(1, 70)<br>(8, 110)<br>(4, 150) | (7,80)<br>(9,100)<br>(16,170) | (1,80)<br>(2,120) |




$$\eta_k \le B_0(\alpha_k)/(1-\alpha_k)$$

# Partition Scheduling

- How to choose a set of ($\alpha_k$, $\eta_k$) for partition servers?
  - Method1:
  - Find minimum $\alpha_k$;
  - If $\Sigma\alpha_k + \alpha_{reserved} < 1$, let $\alpha_k = \min\{\alpha_k\} + \varphi(1 - \Sigma\alpha_k + \alpha_{reserved})$
  - Calculate $\eta_k$ based on Theorem_1
  - Method2:
  - Find the saddle point in the $B_0(\alpha_k)/(1-\alpha_k)$ curve as initial value.
  - Do some adjustment in order to make total capacity=1

# Channel Scheduling

- Scheduling method is almost the same except for:
  - Restrict channel bandwidth $\beta_k \mu_k$ to be integer $\lceil \beta_k \mu_k \rceil$
  - Include release jitters in the schedulability test
  - For tasks in a partition, we can group a subset of tasks and let them share a channel server. (Channel combining)

# Scheduling Approach

- Deadline Decomposition
- Partition and Channel Scheduling
- Channel Combining
- Cyclic Scheduling for Partition and Channel Servers

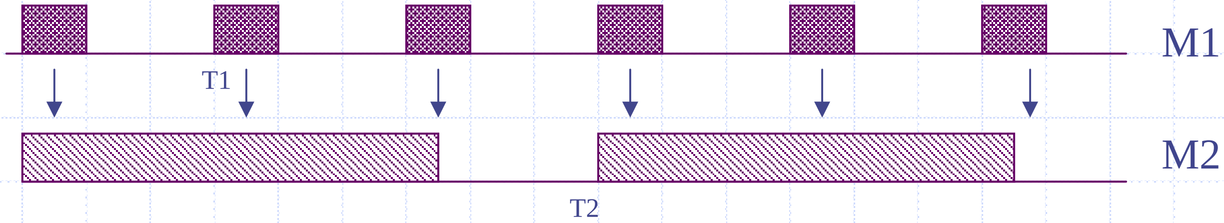
# Channel Combining

- Combine some messages and let them share a common channel server ⇨ bandwidth reduction
- Example:

  $(M_1,MD_1,T_1)$ $(M_2,MD_2,T_2)$, $M_1$ has higher priority

$$CB_1 = M_1/MD_1,\quad CB_2 = M_2/MD_2,$$
$$CB_{12} = \max\{\, M_1/MD_1,\ (M_2+M_1 * \lceil MD_2/T_1 \rceil)/MD_2 \,\}$$

If $MD_1<T_1<MD_2$, then $CB_1+CB_2>CB_{12}$



# Channel Combining

- heuristic channel-combining algorithm

  minimum bandwidth requirement of a channel consisting of messages *1,2,…k.*

$$CB_{12\ldots k} = \max_{j=1,k}\{((\sum_{i=1}^{j-1} M_i * \left\lceil \frac{MD_j}{T_i} \right\rceil + M_j)/MD_j)\}$$

  Assume message *j* has a higher priority then message *j+1.*

# Scheduling Approach

- Deadline Decomposition
- Partition and Channel Scheduling
- Channel Combining
- Cyclic Scheduling for Partition and Channel Servers

# Cyclic Scheduling for Partition Servers

- Let a feasible set of partition capacities and cycles be $(\alpha_1, \eta_1), (\alpha_2, \eta_2), \ldots, (\alpha_n, \eta_n)$

  in non-decreasing order of $\eta_k$

- $\{\eta_k\}$ transformed into a harmonic set $\{h_k\}$

$$h_i = \eta * 2^j \leq \eta_i < \eta * 2^{j+1} = 2 * h_i,$$

  Where $\eta$ is a base partition cycle, candidates $\eta \in (\eta_1/2, \eta_1]$,

  - Find the optimal $\eta$ in sense of processor utilization

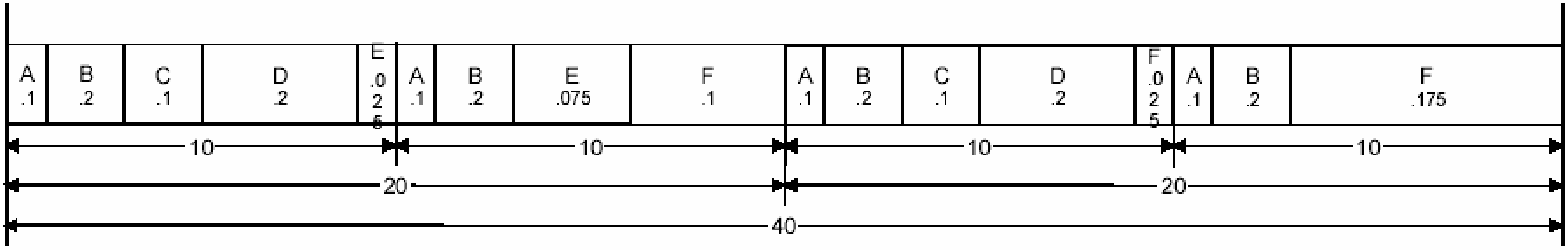
# Cyclic Scheduling for Partition Servers

- Example:

A(0.1 12), B(0.2 14), C(0.1 21), D(0.2 25), E(0.1 48), F(0.3 50)

use the optimal base of 10

A(0.1 10), B(0.2 10), C(0.1 20), D(0.2 20), E(0.1 40), F(0.3 40)



# Cyclic Scheduling for Channel Servers

- The basic method is the same as that of partition server scheduling
- Only difference:

  Channel bandwidth allocation must be done based on integer number of slots.

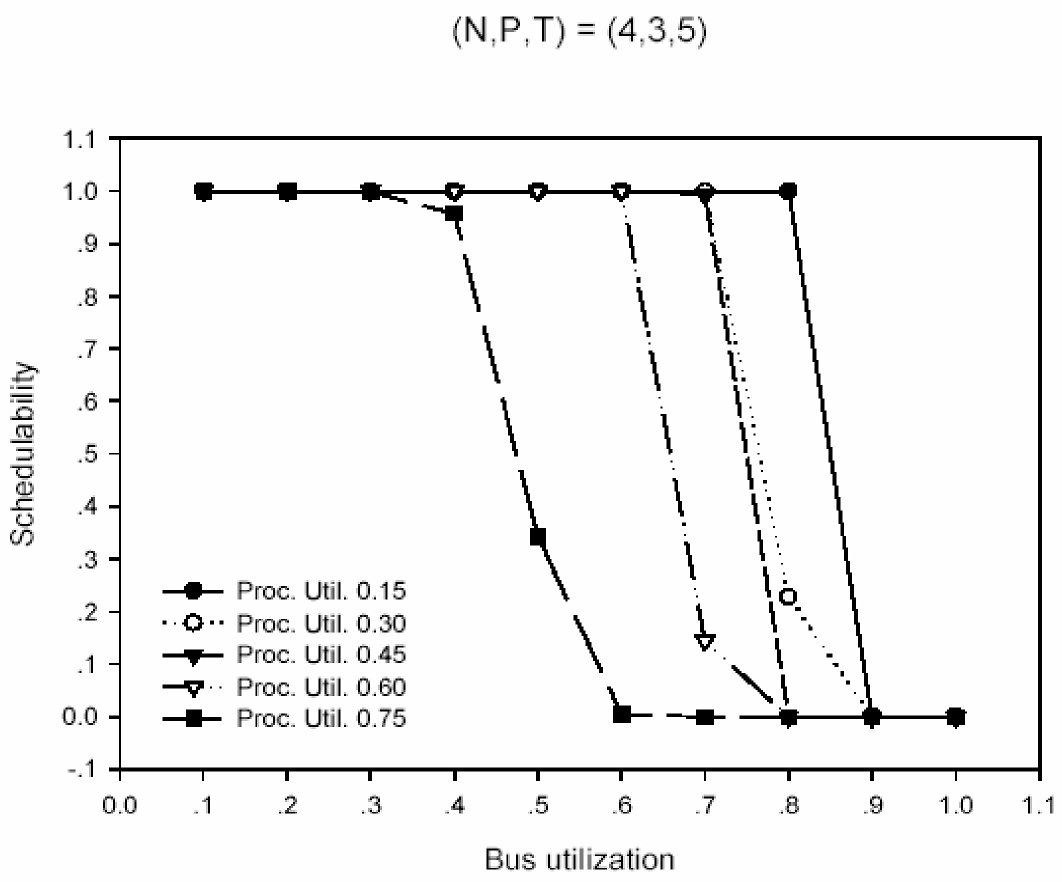
# Algorithm Evaluation

- *Schedulability Test*
- *The Effects of Deadline Decomposition and Channel Combining Algorithm*

# Schedulability Test

- System model: (4 3 5)
  - Four processors
  - Three partitions per each processor
  - Five tasks per each partition
  - Task periods: uniformly distributed between the minimum and maximum periods.
  - Random task sets with variable processor utilization: 15% 30% 45% 60% 75%
  - Message lengths: computed with a random distribution of the total bus utilization and task periods

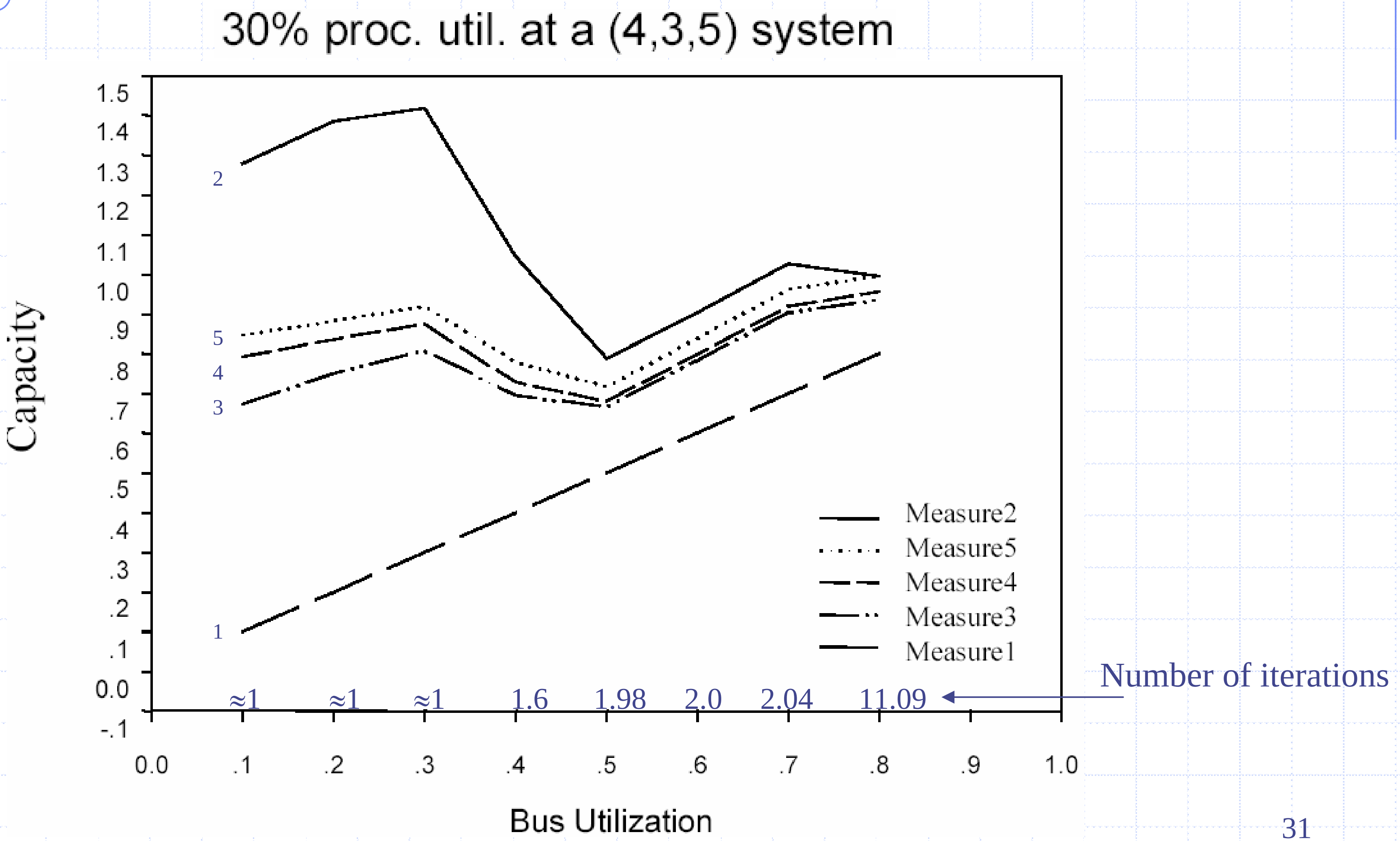
# Schedulability Test



# The Effects of Deadline Decomposition and Channel Combining Algorithm

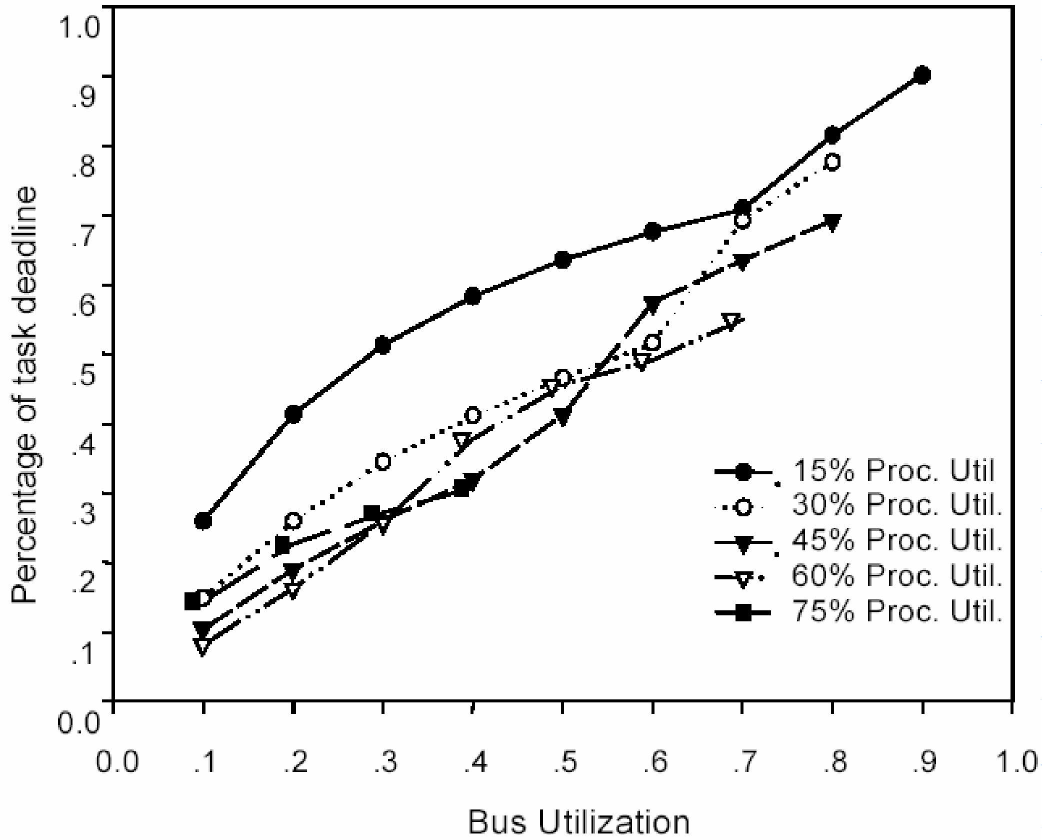
- Measure 1: total bus utilization= $\sum_{i=1}^{n} (ST*M_i)/T_i$
- Measure 2: total bus capacity= $\sum_{i=1}^{n} (ST*M_i)/MD_i$
- Measure 3: total minimum bus capacity = $\sum_{i=1}^{n} \mathrm{Min}\{\beta_k\}$
- Measure 4: total bus capacity = $\sum_{i=1}^{n} \beta_k$
- Measure 5: final bus capacity= $\sum_{i=1}^{n} \lceil \beta_k^h m_k \rceil / m_k$

# The Effects of Deadline Decomposition and Channel Combining Algorithm



# The Effects of Deadline Decomposition and Channel Combining Algorithm

- The other way of looking into the behavior of the deadline decomposition algorithm: $\frac{MD_i}{D_i} = (\frac{ST * M_i}{C_i + ST * M_i}) f_i$



# Conclusion

- Main ideal: use a two-level hierarchical schedule that activates partitions (or channels) following a distance-constraints guaranteed cyclic schedule and then dispatches tasks (or messages) according to a fixed priority schedule.
  - Use a heuristic deadline decomposition technique
  - Develop a heuristic channel-combining algorithm

The simulation analyses show promising results in terms of schedulability and system characteristics

# The end

- Thank you!