دانشگاه بجنورد

CE177

Advanced Operating Systems

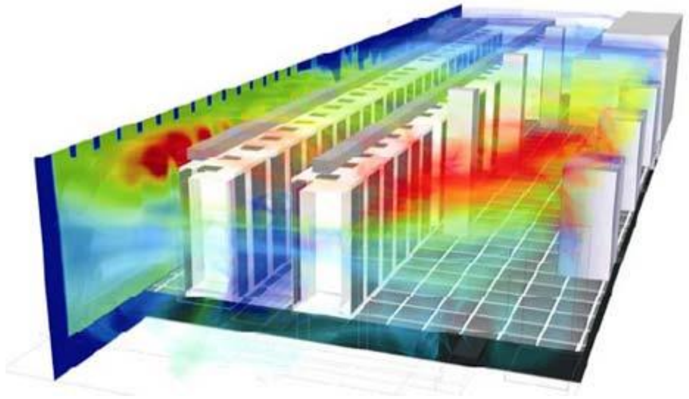
Hamid Fadishei, Assistant Professor
Computer Engineerign Departemnt,
University of Bojnord

Fall 2018

Energy Efficient Systems

Energy Efficiency

- The importance of EE
- The adverse affects of energy inefficiency
  - Increased electricity billing costs
  - Increased cooling costs
  - Decreased reliability of the systems, and even power network failures during peak hours
  - Decreased scalability of the systems
  - An obstacle to Eco-friendly computing



Energy Efficiency

- Servers around the world have consumed $7.2bn electricity in 2005
- This is the double of amount used in 2000

Koomey, Jonathan G. "Estimating total power consumption by servers in the US and the world." (2007).

Electrical Background

- Current (I)
  - The flow of electric charge
  - Measured in Amperes (A)
  - Define the amount of electric charge transferred by a circuit per second
- Power (P) vs Energy (E)
  - P: The rate at which the system performs the work
  - E: the total amount of work performed over a period of time
  - P is measured in Watts(W)
  - E is measured in Watt Hours (Wh)
  - When a charge of 1A is transferred through a voltage difference of 1V, the power rate is 1W
  - When a system with a power rate of 1W works for an hour, an energy of 1Wh is consumed
  -

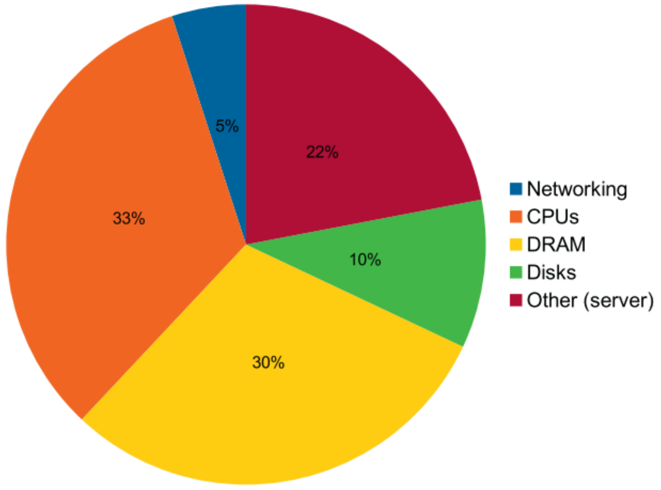
$$E = P \cdot T$$

Computational Power Consumption

- Power consumption of digital (CMOS) circuits has two main components
  - Static Power
    - *Mainly caused by the leakage current*
    - *Present in any active circuit*
    - *Independent of clock rate and usage scenario*
  - Dynamic Power
    - *Consumed the moment the transistors switch*
    - *Depends on clock rate and supply voltage*

$$P_{dynamic} = a \cdot C \cdot V^2 \cdot f \,,$$

Power consumption sources



Barroso, Luiz André, Jimmy Clidaras, and Urs Hölzle. "The datacenter as a computer: An introduction to the design of warehouse-scale machines." Synthesis lectures on computer architecture 8.3 (2013): 1-154.

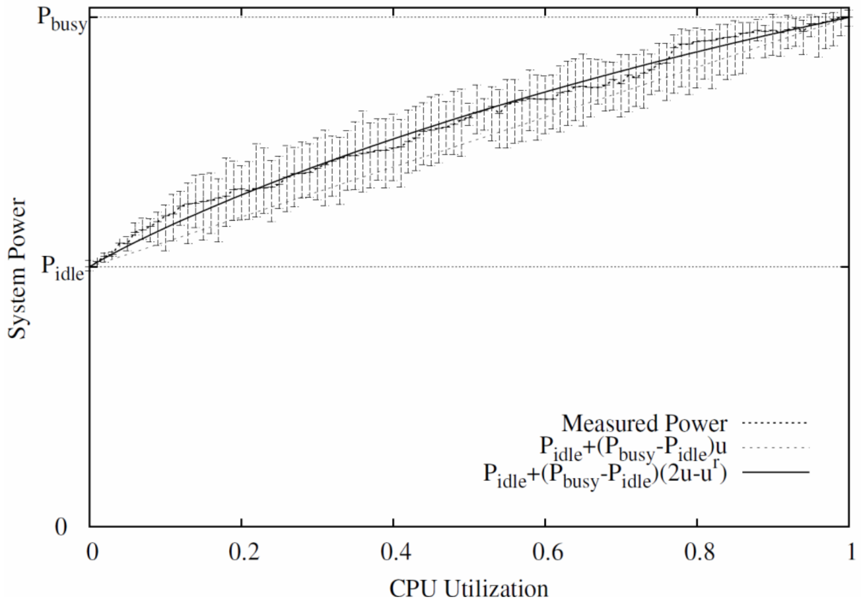
Power consumption model

- Observation: Total power consumption grows linearly with CPU utilization
  - Simplified model of server power consumption:

$$P(u) = P_{idle} + (P_{busy} - P_{idle}) * u$$

  - A more accurate nonlinear model with a calibration parameter:

$$P(u) = P_{idle} + (P_{busy} - P_{idle}) \cdot (2u - u^{r})$$



Fan, Xiaobo, Wolf-Dietrich Weber, and Luiz Andre Barroso. "Power provisioning for a warehouse-sized computer." ACM SIGARCH computer architecture news. Vol. 35. No. 2. ACM, 2007.

Power consumption model

- The utilization-based model is only accurate for cpu-intensive workloads
  - Poor accuracy for I/O-intensive and memory intensive workloads
- To address this problem, machine learning models are proposed by the researchers
  - A richer set of inputs like…
    - *Instructions per cycle*
    - *Memory accesses per cycle*
    - *Cache miss rate*
    - *etc*
- Modern architectures provide performance counters for counting such events
  - See Linux Kernel Perf Events Interface
- Some modern architectures even provide power consumption sensors on hardware components
  - See Intel jRAPL
  - Then is a power consumption model still necessary? Why?

Dhiman, Gaurav, Kresimir Mihic, and Tajana Rosing. "A system for online power prediction in virtualized environments using gaussian mixture models." Proceedings of the 47th Design Automation Conference. ACM, 2010.

Hands on
Linux PERF and RAPL interfaces

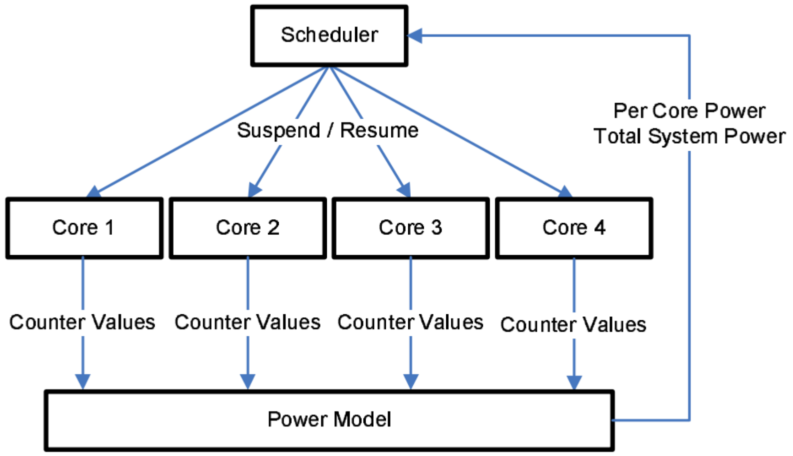
Power modeling using perf. events

- Singh et al use piecewise regression for predicting power consumption of CPU cores from the count of performance events
- They also proposed a simple scheduler that suspends cores to keep the power usage under a predefined envelope

$$\hat{P}_{core} = \begin{cases} 1.15 + 65.88 * r_1 + 11.49 * r_2 + -3.356 * r_3 + 17.75 * r_4, & r_1 < 10^{-5} \\ 23.34 + 0.93 * log(r_1) + 10.39 * r_2 + -6.425 * r_3 + 6.43 * log(r_4), & r_1 \geq 10^{-5} \end{cases}$$

$$where \quad r_i = e_i / 2200000000$$

e_1: L2_CACHE_MISS:ALL
e_2: RETIRED_UOPS
e_3: RETIRED_MMX_AND_FP_INSTRUCTIONS:ALL
e_4: DISPATCH_STALLS



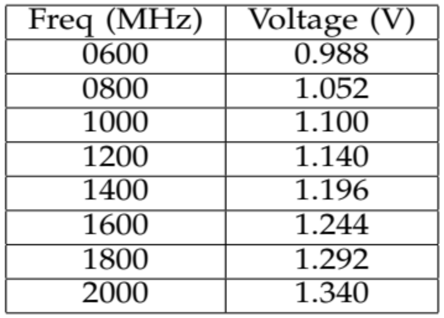
Singh, Karan, Major Bhadauria, and Sally A. McKee. "Real time power estimation and thread scheduling via performance counters." ACM SIGARCH Computer Architecture News 37.2 (2009): 46-55.

Speed Scaling

- The previous techniques is sometimes referred to as CPU idling
- But what if we could avoid idling and Just slow the CPU down instead of suspending it?
- DVFS in modern processors + “Slack” opportunities

DVFS

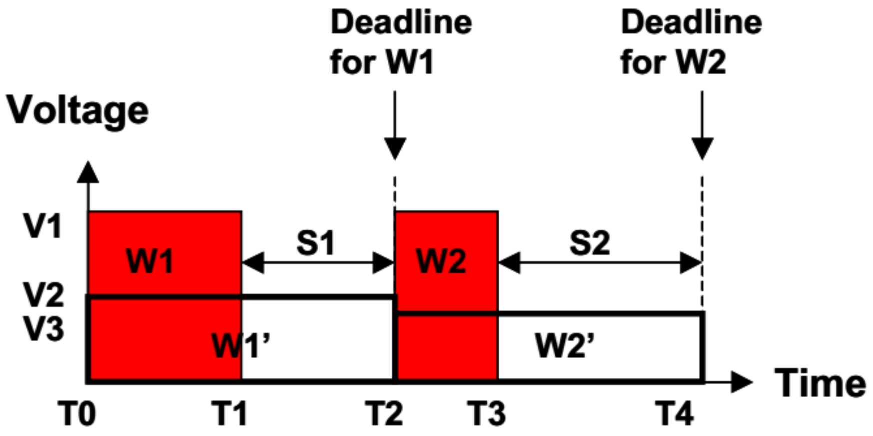
- Dynamic Voltage/Frequency Scaling
  - A technique for power management in modern CPUs
  - The processor can work in several steps of F/V value pairs
  - Lower values => less energy usage at the cost of performance loss
- Remember the analytical CMOS power model?
  - Which of the F/V has the greater impact on power consumption?



Freq (MHz)	Voltage (V)
0600	0.988
0800	1.052
1000	1.100
1200	1.140
1400	1.196
1600	1.244
1800	1.292
2000	1.340

DVFS (exploit deadline slacks)

- Sometimes, performance loss does not hurt us
  - e.g. in realtime tasks if the deadline is not violated
- DVFS can exploits the soft slack times
- Idea: Use the minimum frequency that does not violate the deadline of the subtask



Choi, Kihwan, et al. "Frame-based dynamic voltage and frequency scaling for a MPEG decoder." Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design. ACM, 2002.

DVFS (exploit off-chip slacks)

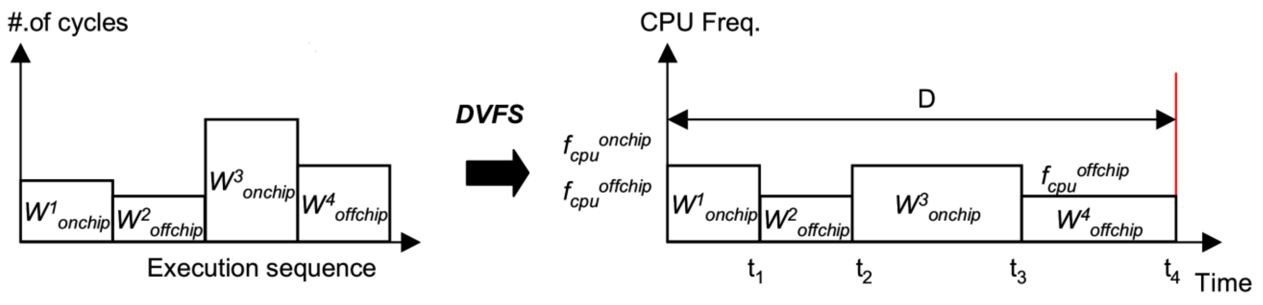
- Workloads consist of two parts: on-chip and off-chip
  - on-chip: operation inside CPU
  - off-chip: memory accesses

$$T = T_{onchip} + T_{offchip} \qquad T_{onchip} = \frac{W_{onchip}}{f_{cpu}}, \qquad T_{offchip} = \frac{W_{offchip}}{f_{mem}}$$

W is the number of clock cycles required to perform the task

- Idea: CPU can be slowed down during memory accesses

$$\frac{\Delta T}{\Delta f} = \frac{\Delta T_{onchip}}{\Delta f}, \qquad \frac{\Delta T_{offchip}}{\Delta f} \approx 0 \qquad f_{cpu}^{onchip} = \frac{W_{onchip}^1 + W_{onchip}^3}{D - \left(\frac{W_{offchip}^3 + W_{offchip}^4}{f_{mem}}\right)}, \qquad f_{cpu}^{offchip} = f_{cpu}^{min}$$



Choi, Kihwan, Ramakrishna Soma, and Massoud Pedram. "Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times." IEEE transactions on computer-aided design of integrated circuits and systems 24.1 (2005): 18-28.

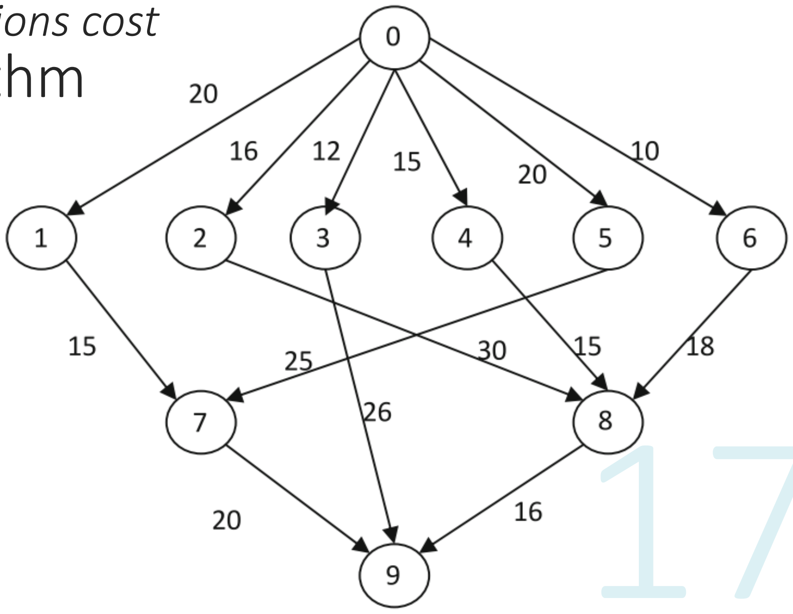
DVFS (exploit off-chip slacks)

- The ratio $\beta = \frac{T_{offchip}}{T_{onchip}}$
- The degree to which energy saving is possible
- The higher the β, The higher energy saving opportunity
- Problem: β is not known for future slots
- Choi et al propose a heuristic:
  - Given a predefined performance loss PF_{loss}
  - Calculate the frequency for the slot t+1 from frequency of the current slot

$$f^{t+1} = \frac{f_{max}}{1 + PF_{loss} \cdot \left[1 + \beta^{t} \cdot \left(\frac{f_{max}}{f^{t}}\right)\right]}$$

DVFS (exploit dependency slacks)

- Another opportunity for reducing the clock frequency is when executing a workflow of several tasks
  - Common in grid and cluster systems
  - Idea: Some tasks can run slower as their result are not required any soon
- Workflow graph
  - A DAG
  - Represents execution dependency of a number of tasks
  - Each node: a task
    - *Each task has an execution time (t-level)*
    - *The critical path from each task (b-level)*
    - *Edges optionally represent the communications cost*
- An energy-efficient scheduling algorithm tries to slow-down or even turn-off nodes when possible



Workflow Scheduling

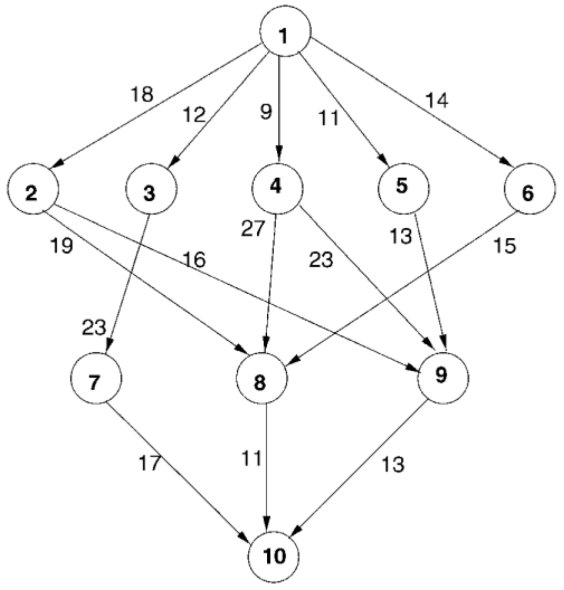
- Several algorithms exist for efficient workflow scheduling
- For example, HEFT is a well-known heuristic for this purpose
- It is designed for performance, not energy efficiency
- HEFT = Heterogeneous Earliest Finish Time
- Works by calculating a rank for each workflow node
- Rank is a recursive function calculated from entry point to the exit point of the graph

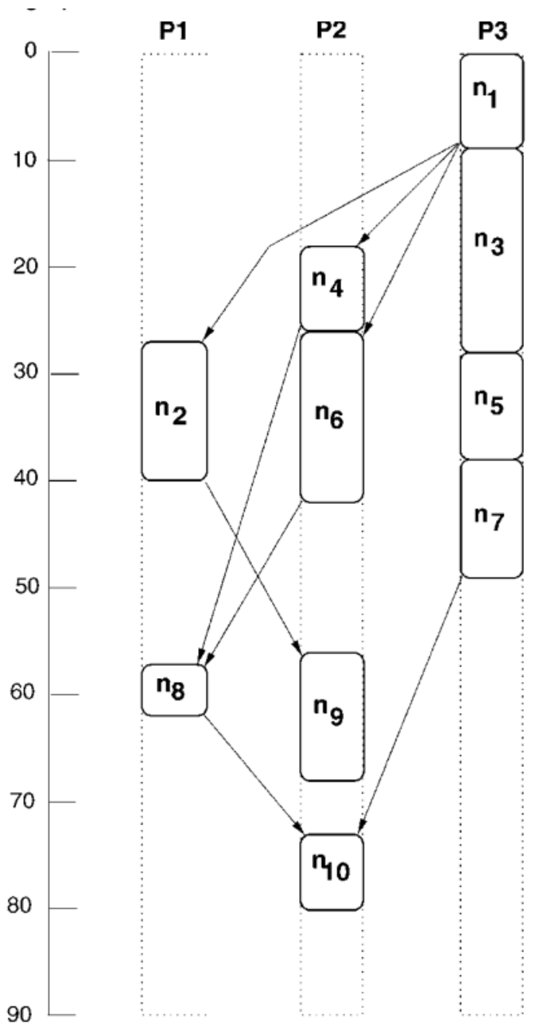
$$rank_u(n_i) = \overline{w_i} + \max_{n_j \in succ(n_i)} (\overline{c_{i,j}} + rank_u(n_j)),$$

- Nodes are reverse-sorted by rank and assigned from the top of the list one by one

Topcuoglu, Haluk, Salim Hariri, and Min-you Wu. "Performance-effective and low-complexity task scheduling for heterogeneous computing." IEEE transactions on parallel and distributed systems 13.3 (2002): 260-274.

HEFT Algorithm Example



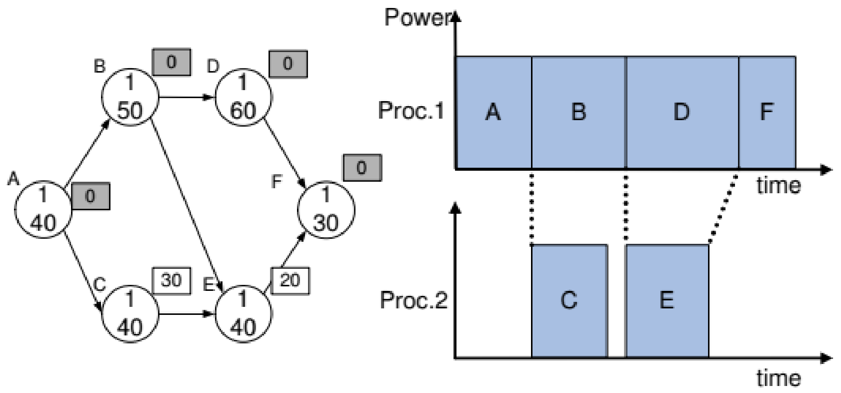


Energy-Efficient workflow scheduling

- Assume that the tasks are already assigned to nodes using an algorithm like HEFT
- We want to find opportunities to slow down some nodes without violating the total deadline

$succ(i)$ = { j | the edges from i to j exist }
$pred(i)$ = { j | the edges from j to i exist }

$$T_{start}(i) = max_{j \in pred(i)} \left(T_{start}(j) + T_{exec}(j)\right)$$
$$T_{end}(i) = max_{j \in succ(i)} \left(T_{end}(j) - T_{exec}(j)\right)$$



Kimura, Hideaki, et al. "Emprical study on reducing energy of parallel programs using slack reclamation by dvfs in a power-scalable high performance cluster." Cluster Computing, 2006 IEEE International Conference on. IEEE, 2006.

Energy-Efficient workflow scheduling

(1) Calculate slack time

Calculate the slack time for each of the tasks. Let the slack time of *task-i* be $D(i)$. The slack time is obtained by computing the earliest start time and the latest end time as follows:

$$D(i) = T_{end}(i) - (T_{start}(i) + T_{exec}(i))$$

If the $D(i) = 0$, it means there is no room to change the gear because *task-i* belongs to a critical path. Therefore, mark this *task-i* as "defined".

(2) Select the task to change the frequency

Pick the *task-k* which has the longest T_{path}. T_{path} is the sum of the execution time of the tasks on the path from *task-k* to the "defined" task.

(3) Change frequency

The Equation (2) gives the frequency $F(k)$ at which *task-k* is executed.

$$F(k) = f_s \times \frac{T_{path}}{T_{path} + D(k)} \qquad (2)$$

where f_s is the standard frequency. The execution time of *task-k* increases because the frequency is decreased. Thus, we reduce the energy consumption as uniformly as possible along a path of several tasks by using T_{path}.

(4) Update the DAG

After changing the frequency, update the execution time of *task-k* and the slack time. We now mark *task-k* as "defined".

Steps (1) to (4) are repeated until all nodes are "defined".

Energy-Efficient workflow scheduling

```
repeat
    for all $i$ such that $pred(i)$'s $T_{start}$ was updated do
        calculate: $T_{start}(i)$
    end for
    for all $i$ such that $succ(i)$'s $T_{end}$ was updated do
        calculate: $T_{end}(i)$
    end for
    for all $i$ do
        calculate: $D(i) = T_{end}(i) - (T_{start}(i) + T_{exec}(i))$
    end for
    pick up the $task - k$ which has longest $T_{path}$
    calculate: $F(k) = f_s \times T_{path}/(T_{path} + D(k))$
    mark $task - k$ as "defined"
until all nodes are "defined"
```

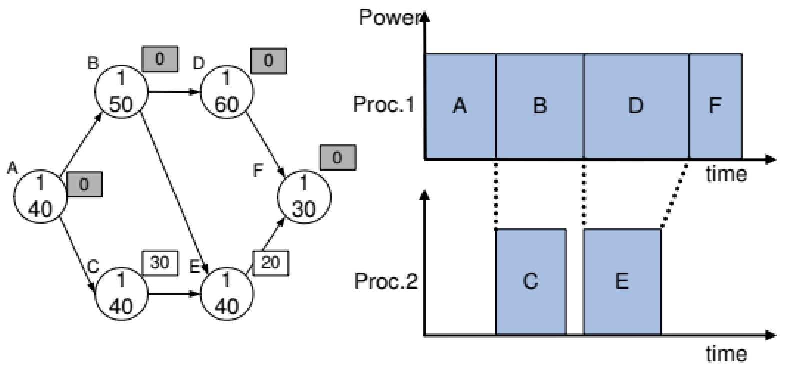


Figure 4. Example (1)

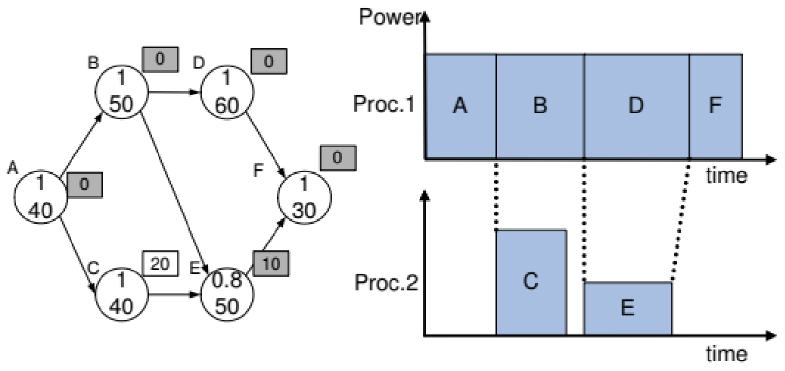
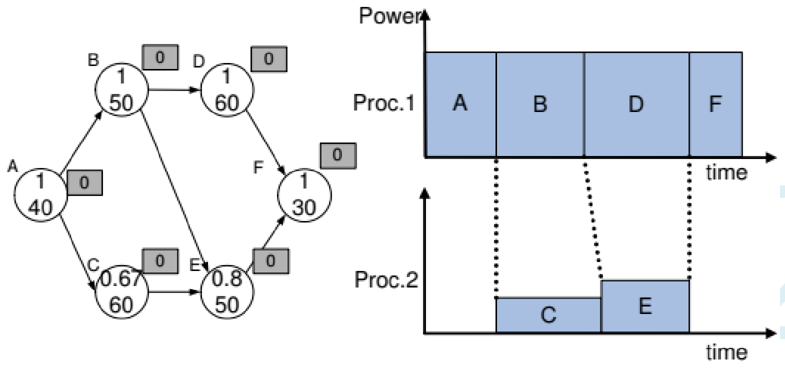


Figure 5. Example (2)



Tasks and Exercises

Task 1 (Term Project)

- Single student
  - Study the assigned paper
  - Present in class (due 2 weeks)
  - Find and study 5 related papers
  - Deliver a paper-formatted survey (due final exam)

- 1st Student:
  - Geng, Yeli, Yi Yang, and Guohong Cao. "Energy-Efficient Computation Offloading for Multicore-Based Mobile Devices." IEEE INFOCOM. 2018.
- 2nd Student:
  - Tang, Chaogang, et al. "Energy Efficient and Deadline Satisfied Task Scheduling in Mobile Cloud Computing." *Big Data and Smart Computing (BigComp), 2018 IEEE International Conference on*. IEEE, 2018.
- 3rd Student:
  - Wang, Songyun, et al. "A DVFS based energy-efficient tasks scheduling in a data center." IEEE Access 5 (2017): 13090-13102.