CS 423 – Operating Systems Design

Lecture 31 – Load Balancing Distributed Computing (Part 5)

Klara Nahrstedt
Fall 2011

Based on slides by Andrew S. Tanenbaum and paper

Sharma, Singh, Sharma, "Performance analysis of Load Balancing Algorithms", 2008 (http://www.waset.org/journals/waset/v38/v38-47.pdf)

Administrative

- MP4 is out
- Deadline – December 2 + bonus days if you have left any
- Interviews for Linux projects – Monday, December 5 (morning/afternoon)
- Interview/Competition of Android projects – Monday, December 5 - evening

Load balancing

- Processor allocation algorithms
  - Processes assigned to nodes in efficient manner
- Difference
  - What we assume is known
    - CPU requirement, memory usage, amount of communication among processes
  - What the goal is
    - Minimize CPU cycles due to lack of local work
    - Minimize total communication bandwidth
    - Ensure fairness to users and processes

Load Balancing Parameters

- Load balancing algorithms are evaluated based on their performance
- Performance of load balancing algorithms is measured by various parameters
  - Overload rejection
  - Fault Tolerance
  - Forecasting Accuracy
  - Centralized versus Decentralized
  - Static versus Dynamic
  - Cooperative versus Non-cooperative
  - Process Migration
  - Resource Utilization

ALGORITHM DESIGN

Overload Rejection

- If load balancing is not possible, additional overload rejection is needed
- When overload situation ends, then first overload rejection measure should be dropped
- After short guard period, load balancing should be also closed down.

Fault-Tolerant Parameter

- Is load balancing algorithm able to tolerate major faults or not?
- If performance of load balancing algorithm decreases, decrease should be proportional to seriousness of failure

Forecasting Accuracy

- Forecasting is defined as degree of conformity of calculated results to its actual value that will be generated after execution
- Note
  - Static algorithms provide more accuracy than dynamic algorithms

Stability

- This parameter is characterized in terms of delays in transfer of information between processors and gain in load balancing algorithm by obtaining faster performance by specific amount of time.

Centralized vs Decentralized

- Centralized scheme stores global information at designated node
  - All sender/receiver nodes access designated node and calculate amount of load transfer and check that tasks are to be sent/received from
- Decentralize scheme means that every node executes balancing separately
  - Idle nodes can obtain load during runtime from shared global queue of processes

Nature of Load Balancing Algorithm

- Static algorithm assigns load to node probabilistically or deterministically without considering runtime trends.
  - Under this algorithm, it is not possible to make predictions of arrival times of loads and processing times required for future loads
- Dynamic algorithm uses load distribution during run-time and it is based on running processes rates and network loads.

Cooperative

- If processes share information among them in making processes allocation decision during execution or not!
  - This parameter defines extend of independence that each process has in concluding how to use its own resources
- In cooperative situation, all processes have accountability to carry out its own portion of scheduling tasks, but also processes work together to achieve goal of better efficiency.
- In non-cooperative situation, processes are independent and arrive at decision about the use of their resources without any effect of their decision on the rest of system

Resource Utilization

- Automated load balancing towards efficient resource usage

Static Load Balancing

- Depending on processor performance, workload is distributed in the start by master processor
- Slave processes calculate their allocation work and submit results to master
- Task is executed on the processor to which it was assigned – non-preemptive
- Goal – reduce overall execution time of concurrent program while minimize communication delays
- Disadvantage ?

Round Robin and Randomized Algorithms

- RR – processes are divided evenly between all processors
- Each new process is assigned to new processors in RR order
- Allocation assumes independence of remote processors and communication delay
- Approach works well with
  - # processors > #processes

Central Manager Algorithm

- Central manager selects host for new process
- Minimally loaded processor is selected when process is created
- Load manager selects – load balancing judgment
- Remote processors update load info and send message each time load changes
- Load information:
  - Waiting time of parent's process of completion of its children's process
  - End of execution
- Disadvantage?

Threshold Algorithm

- Processes are assigned to hosts immediately upon creation
- Hosts for new processes are selected locally without sending remote messages
- Each processor keeps private copy of system load
- Load of processor characterized by
  - Under-loaded load < *tunder*
  - Medium *tunder* ≤ load ≤ *tupper*
  - Over-loaded load > *tupper*

Threshold Algorithm

- Two important thresholds: tunder and tupper
- Algorithm
  - Initially all processors considered under-loaded
  - When load state exceeds load level limit, then it sends message regarding new load state to all remote processors
    - Host regularly updates the actual load state of entire system
  - If load state not overloaded -> process allocated locally
  - If load state overloaded -> select remote under-loaded processor
  - If load state overloaded and no remote under-loaded processors exists -> process allocated locally
- Disadvantage?

Dynamic Load Balancing

- Workload distributed among processors at runtime
- Master assigns new processes to slaves based on new load information collected
- Dynamic algorithm allocates processes when one of the processors becomes under-loaded
- Processes are buffered in queue at main host and allocated dynamically upon request from remote hosts

Central Queue Algorithm

- Principle of dynamic load distribution applies
- Queue manager stores new activities and unfulfilled requests as cyclic FIFO queue on main host
- Each new activity is inserted to the queue
- When new request for activity arrives, queue manager removes first activity from queue and sends it to requester

Central Queue Algorithm

- When processor load falls under threshold, the local load manager sends request for new activity to central load manager
- Central load manager answers request immediately if ready activity is found in the process-request queue

Local Queue Algorithm

- Dynamic process migration support assumed
- Idea: static allocation of all new processes with process migration initiated by host, when its load falls under threshold limit
  - Threshold limit is user-defined parameter
- Optimize Parameter – minimal number of ready processes the load manager attempts to provide on each processor

Local Queue Algorithm

- Initially, new processes created on main host are allocated to all under-loaded hosts
- Number of parallel activities created on main host is usually sufficient for allocation on all remote hosts
- When host is under-loaded, local load manager attempts to get several processes from remote hosts
- It randomly sends requests with number of local ready processes to remote load managers
- When load manager receives request, it compares local number of ready processes with received number
- If former is larger than latter, then some of running processes are transferred to requester and acknowledgement with number of processes transferred is returned

Comparative Results

Parameters	Round Robin	Random	Local Queue	Central Queue	Central Manager	Threshold
Overload Rejection	No	No	Yes	Yes	No	No
Fault Tolerant	No	No	Yes	Yes	Yes	No
Forecasting Accuracy	More	More	Less	Less	More	More
Stability	Large	Large	Small	Small	Large	Large
Centralized/Decentralized	D	D	D	C	C	D
Dynamic/Static	S	S	Dy	Dy	S	S
Cooperative	No	No	Yes	Yes	Yes	Yes
Process Migration	No	No	Yes	No	No	No
Resource Utilization	Less	Less	More	Less	Less	Less

ASSIGNMENT APPROACHES OF PROCESSES TO PROCESSORS

Graph-Theoretic Deterministic Algorithm

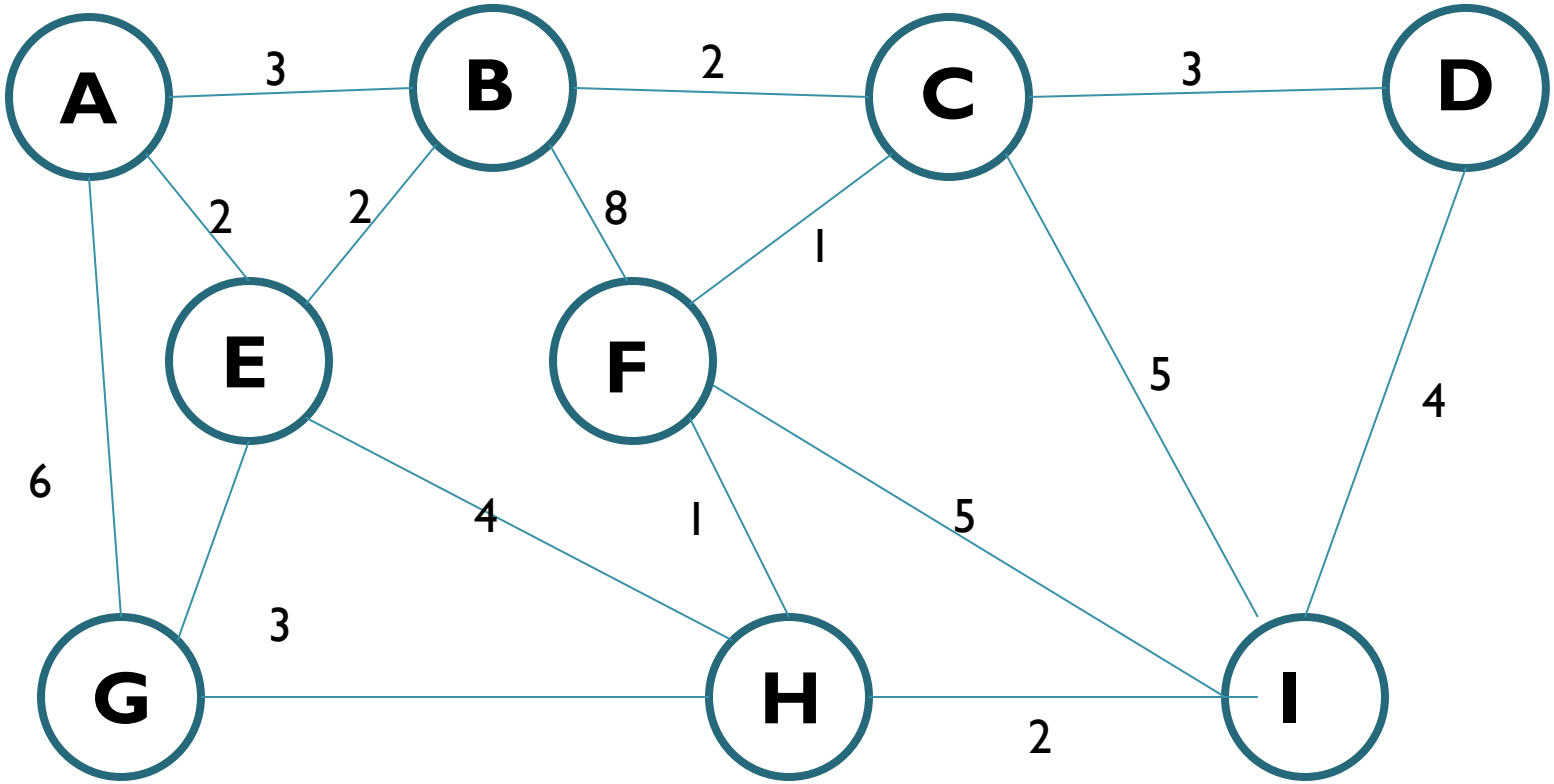
- Number of processes is greater than number of CPUs, then *k* processes are assigned to the same CPU

- Goal is to perform assignment such that to minimize network traffic

Graph-Theoretic Deterministic Algorithm

- System can be represented as a weighted graph
- Each vertex – process
- Each edge – flow of messages between two processes
- Mathematically – find a way to partition (cut) the graph into *k* disjoint sub-graphs
  - subject to constraints (e.g., CPU and memory requirements)

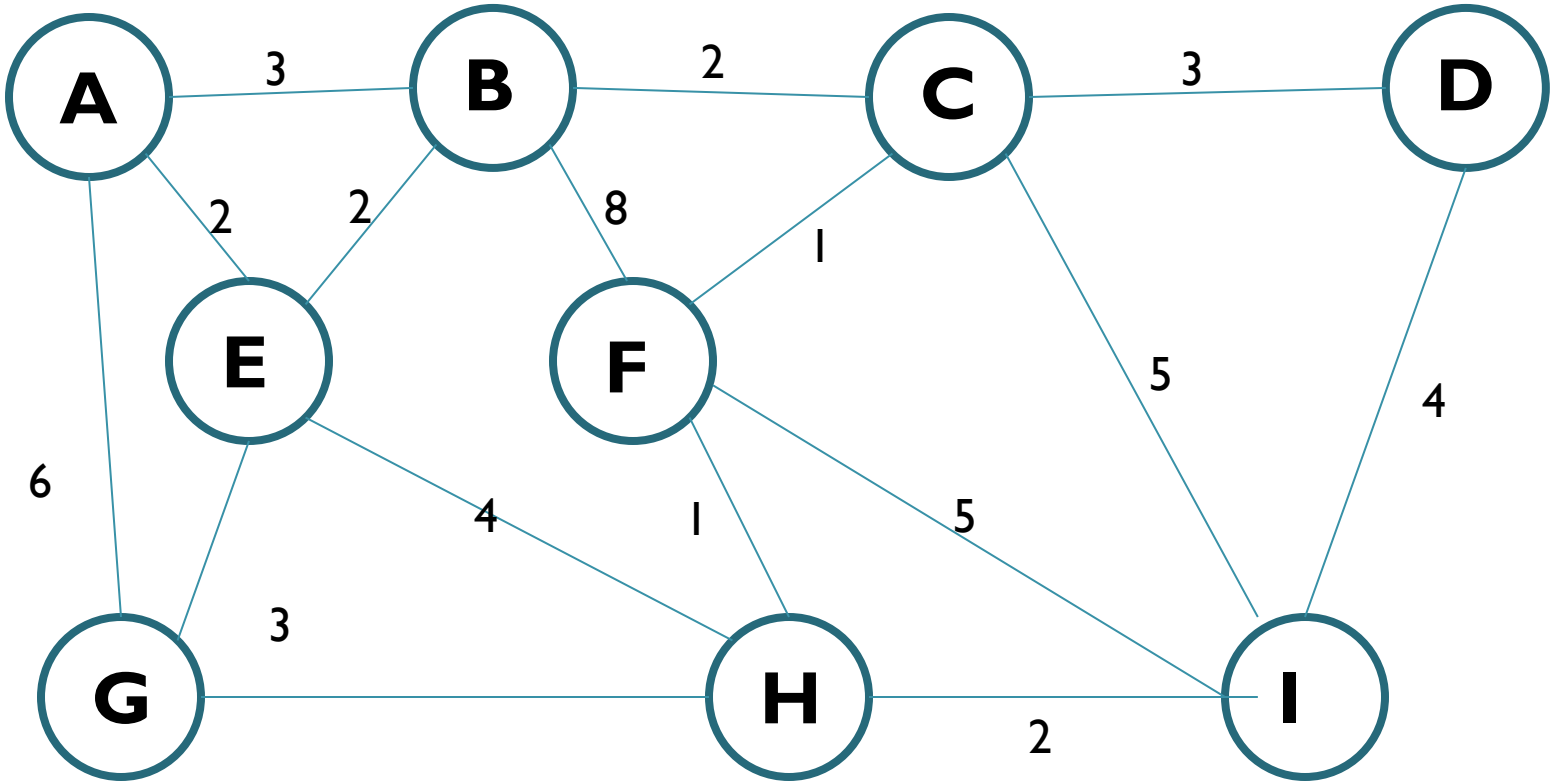
Example



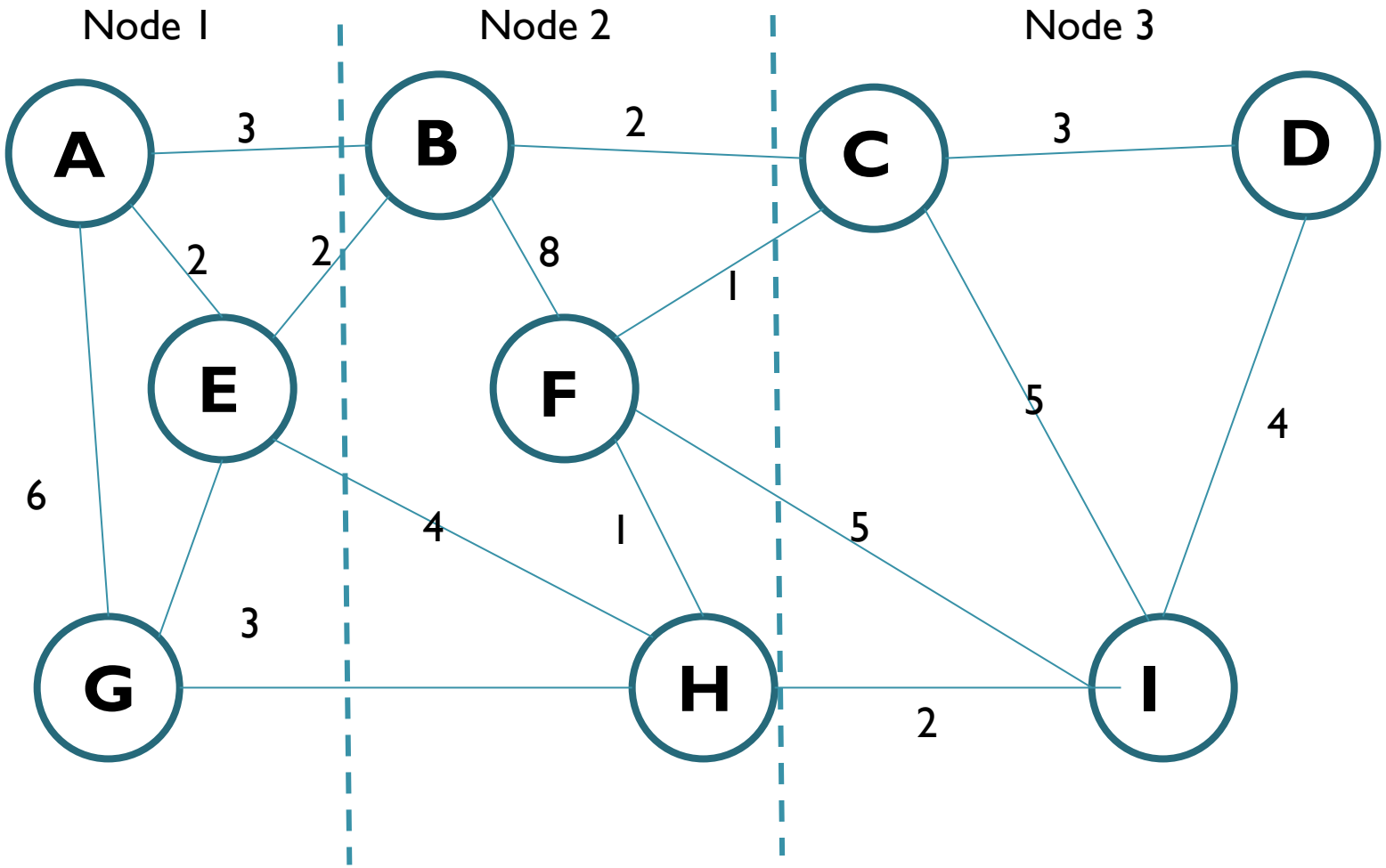
Deterministic Algorithm

- For each solution
- Arcs that are entirely within a single sub-graph represent intra-machine communication and can be ignored
- Arcs that go from one sub-graph to another represent network traffic
- Goal is to find partitioning that minimizes network traffic while meeting constraints

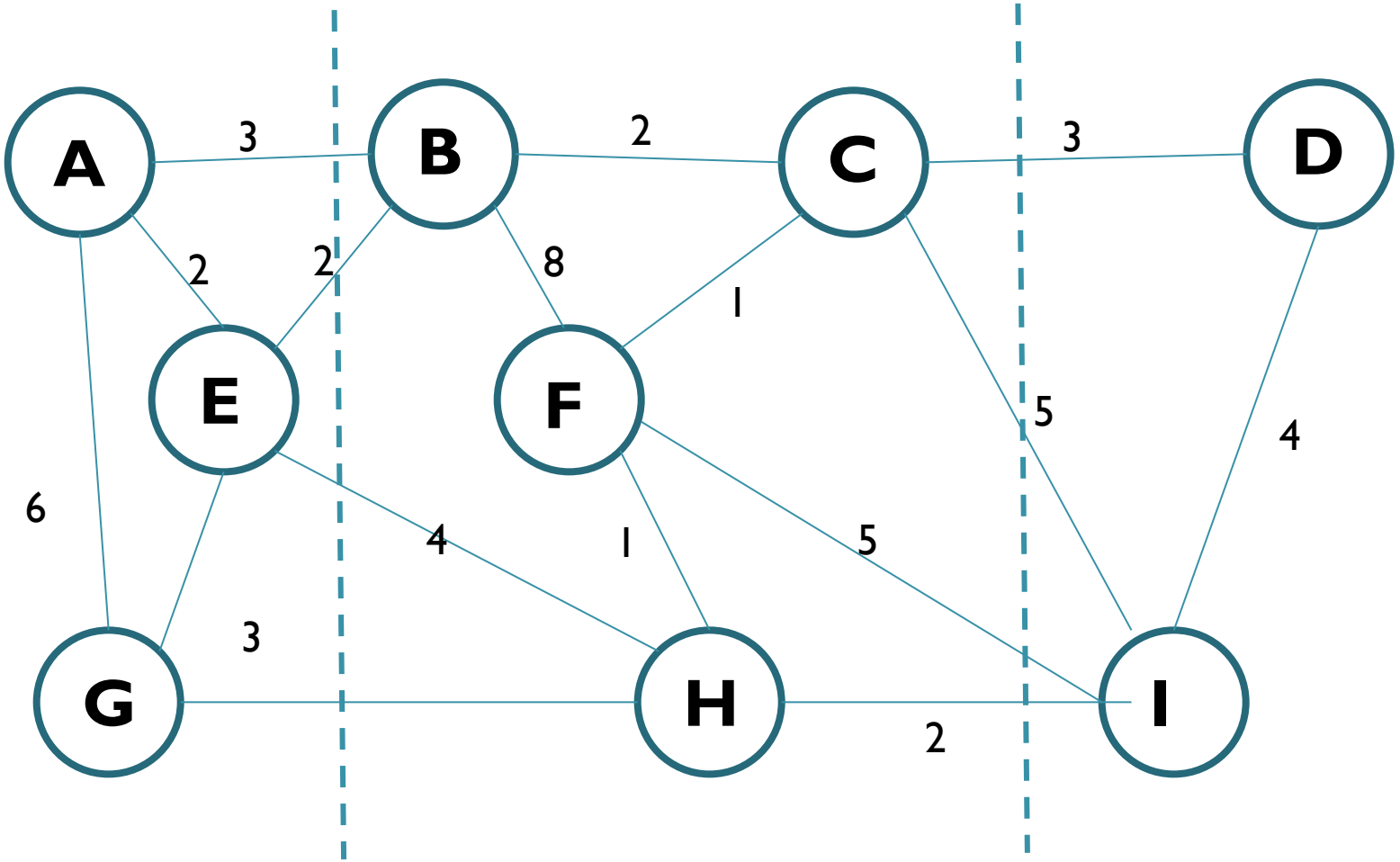
Example



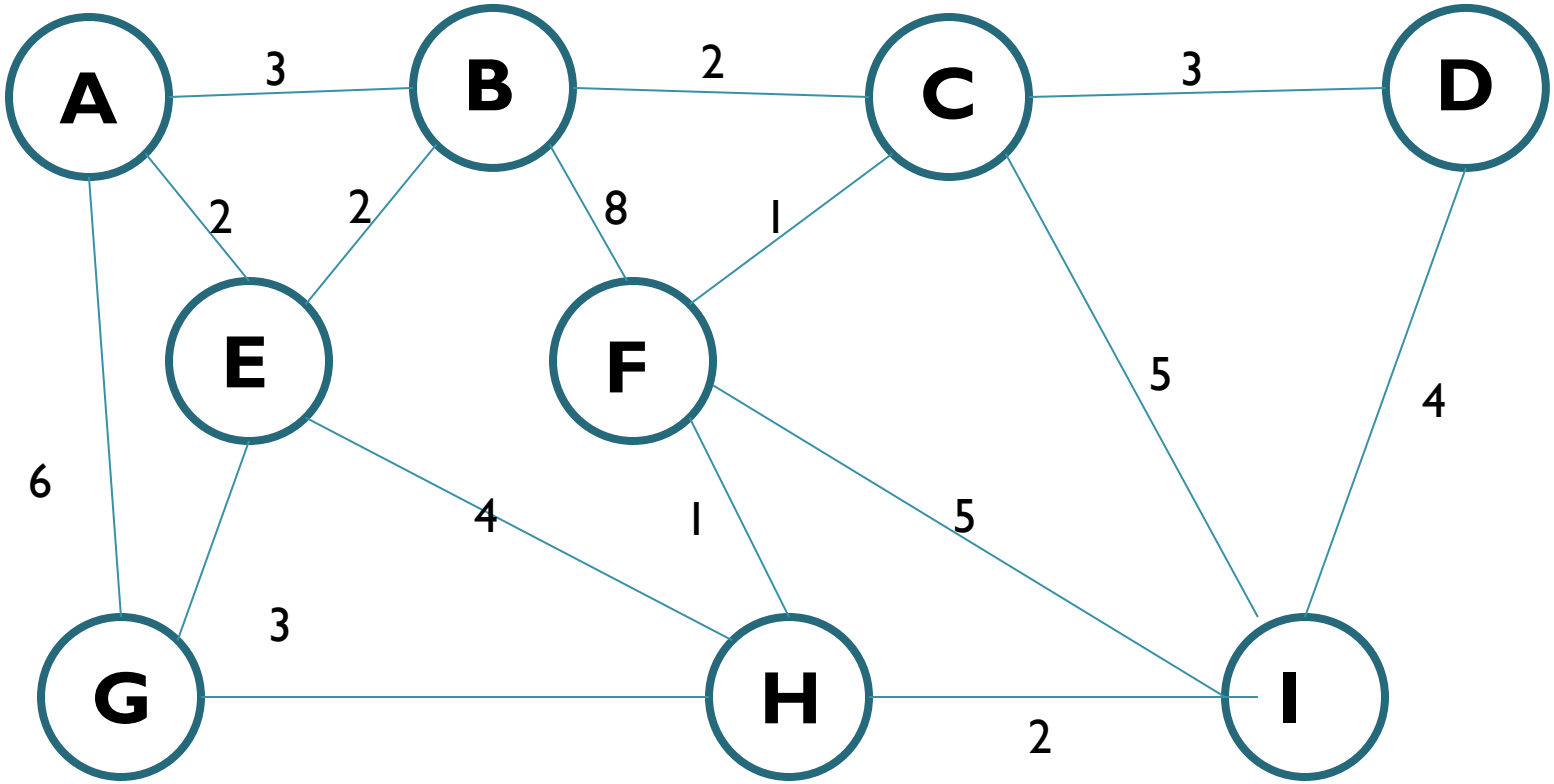
Example (Cut 1)



Example (Cut 2)



Example : Any Other Cut?



Sender-initiated distributed heuristic algorithm (Eager Algorithm)

- **Step 1**: when a process is created , it runs on the node that created it unless node is overloaded
  - Overload metrics – too many processes; too big total working set, …
- **Step 2**: If a node is overloaded, the node selects another node at random and asks it what its load is
- **Step 3:** If probed node's load is below some threshold value, the new process is sent there

Sender-Initiated Load Balancing

- **Step 4:** If not, another machine is chosen for probing
  - Probing is not done forever,
- **Step 5**: If no suitable host is found within *N* probes, algorithm terminates and the process runs on the originating machine

- Advantages – Disadvantages ?

Receiver-initiated distributed heuristic algorithm

- Step 1: Whenever a process finishes , system checks to see if it has enough work,
- Step 2: If not, it picks some machine at random, and asks for work,
- Step 3: If that machine has nothing to offer, a second and then third machine is asked

Receiver- initiated Load Balancing

- Step 4: If no work is found after *N* probes, the node temporarily stops asking, does any other work it has queued up, and tries again after another process finishes.
- Step 5: If no work is available, machine is idle.
- Step 6: After some fixed interval , it begins probing again.

- Advantages and Disadvantages?

Conclusion

- Careful consideration of load balancing algorithms in terms of their parameters
- Hybrid algorithms are used combining several parameters and approaches