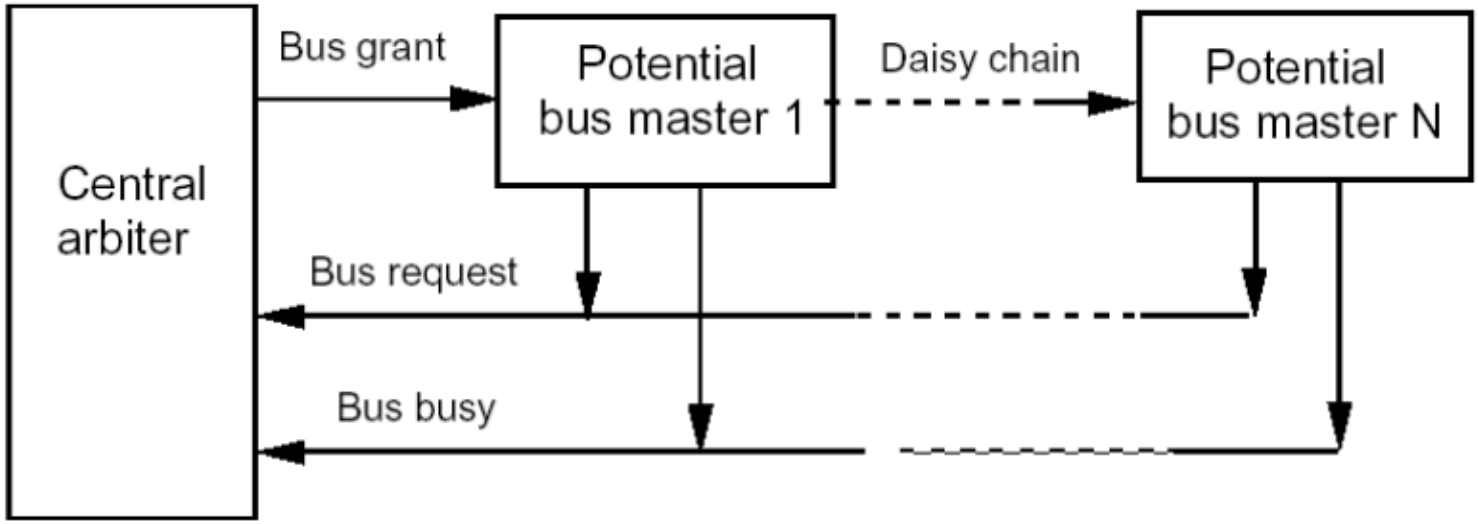
Bus Arbitration

Daisy chain: Permission passed along the chain of devices



Priority is based on proximity to arbiter

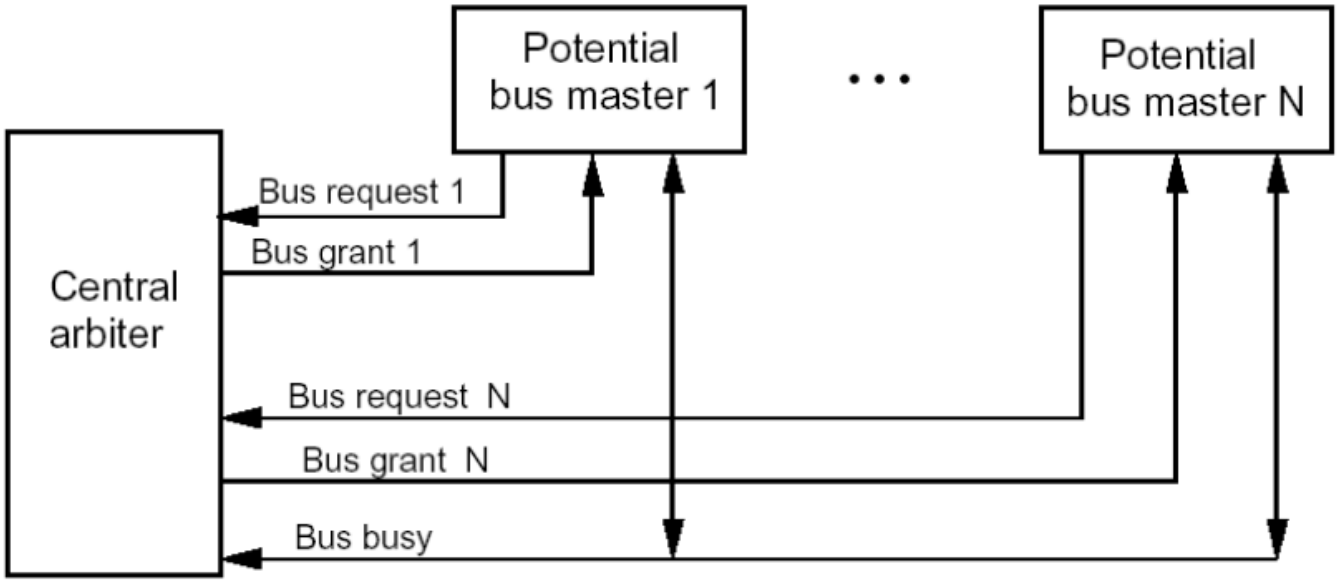
Malfunctioning device can lockout others that follow

Requires relatively few bus lines

Bus Arbitration

Centralized parallel:

Each device is directly connected to the arbiter



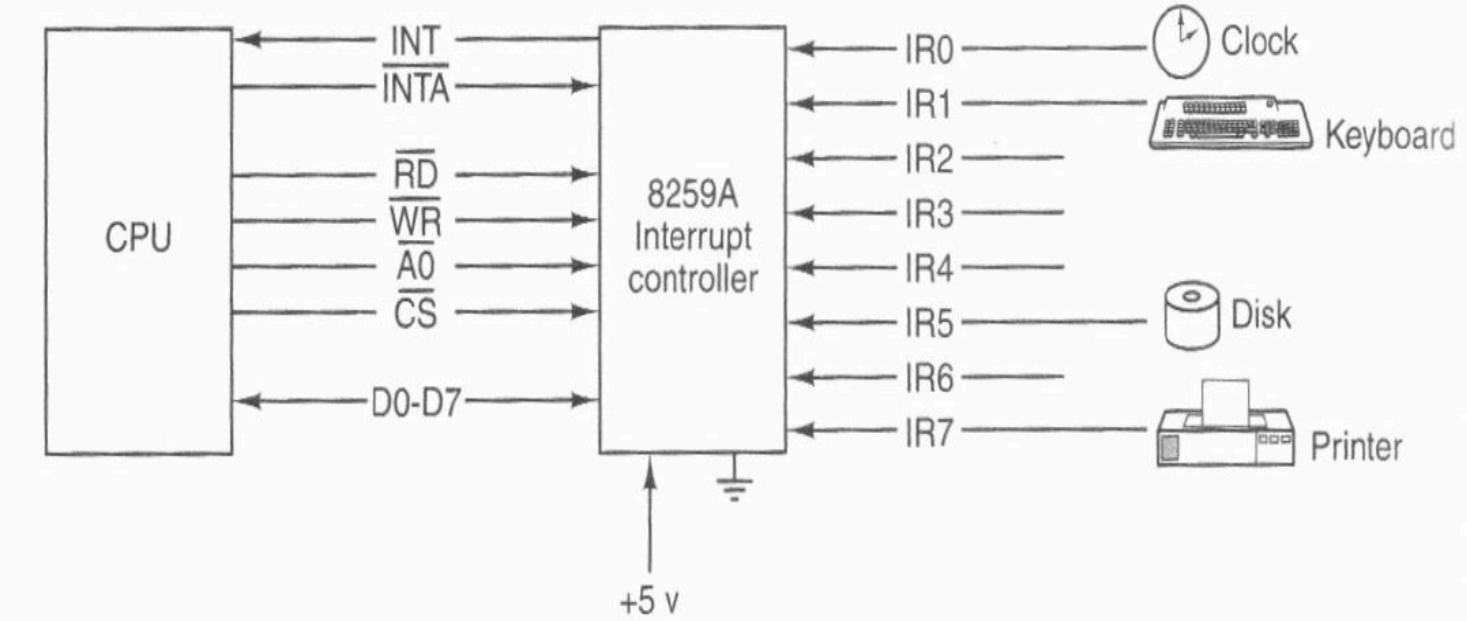
Arbiter assigns priority based on separate grant lines

Malfunctioning devices can be ignored

Requires more bus lines than daisy chained alternative

Bus Arbitration

Centralized parallel example:



Up to 8 I/O controller chips can be connected

Using the 8 IRQ (interrupt request) lines of the 8259A controller

Used on early PC systems (Intel)

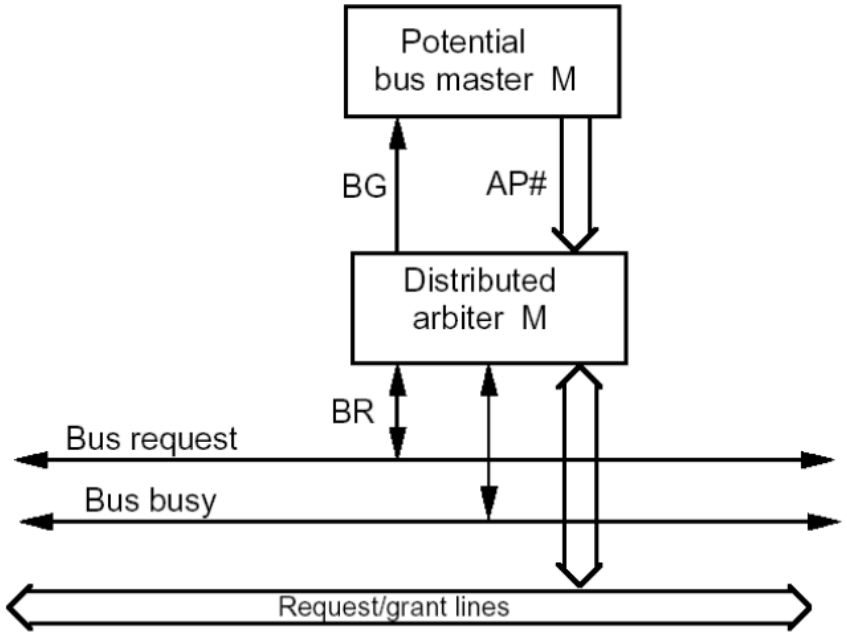
Bus Arbitration

- Distributed arbitration using self-detection:
  - Devices decide among themselves who gets the bus

- Arbitration priority numbers (AP#) determine order
  - Requester AP# is compared with that of other current requesters
  - Requests with lower AP# are removed
  - Remaining request is granted

- Distributed arbitration using collision-detection:
  - Any device can try to use the bus
  - Waits and tries again if a collision occurs

Bus Arbitration



- Starvation can occur with priority based arbitration
  - Very frequent high priority requests can lockout others

- Fairness can be imposed by centralized or distributed arbiter
  - Limit number of consecutive grants to high priority devices

Summary

Option	Higher performance	Lower cost
Type	Separate address, data and control lines operate in parallel	Multiplexed lines carry address and data at different times
Transfer size	Block transfers have less overhead	each data unit incurs overhead for individual transfers
Bus masters	Multiple requires arbitration but increases flexibility	Single master requires no arbitration
Transaction type	Split transactions allow interleaving of transfers	Continuous connection may tie up lines while waiting
Width	More lines allow more data per transfer	Fewer lines reduce cost but limit amount of data in a transfer
Clocking	Synchronous is faster but may be limited by the slowest device	Asynchronous requires handshake signals (more overhead) but is not limited by the slowest device