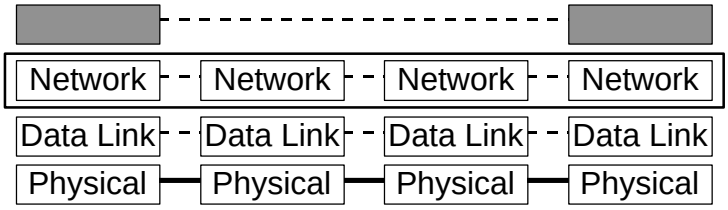
Distributed Algorithms (1)

- Peers must cooperate to perform network functions
- A distributed algorithm is decomposed into one or more local algorithms
- Each local algorithm proceeds based on the data received from other layers or peers, and the order in which the data is received

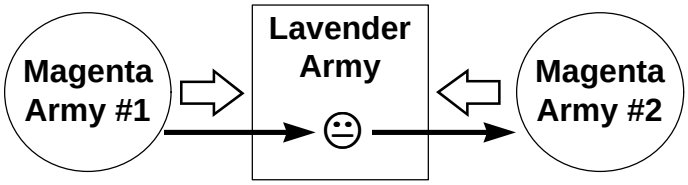


Distributed Algorithms (2)

- These algorithms are complex because underlying services may be unreliable
- Data may:
  - Never arrive (due to transmission error, overflow, etc.)
  - Arrive late (due to arbitrary network delay)
  - Arrive out of order (due to differing network paths)
- It may be impossible to ensure correct operation 100% of the time
  - Maximize probability of success
  - Detect errors

Example: Magenta and Lavender Armies (1)

- Magenta Armies #1 and #2 must attack simultaneously to defeat the Lavender Army
- Magenta Army #1 wants to send a messenger (😐) to Magenta Army #2 to set a time for the attack
- The messenger must go through enemy territory (an unreliable communication channel)
  - may be delayed (until after the attack time)
  - may be captured (so that message is never delivered)



Example: Magenta and Lavender Armies (2)

- Possible solution: require Magenta Army #2 to send another messenger to acknowledge that the first messenger arrived with the message
  - Acknowledgment messenger may be delayed or captured
  - Magenta Army #2 would think that the attack is on, but Magenta Army #1 cannot know if it is on or not
- There is no possible solution to the problem with probability 1 of success
- The attack can be synchronized with high probability
  - For example, send many messengers to increase likelihood of one reaching Magenta Army #2