# Contemporary Operating Systems are not ready for Peer Computing

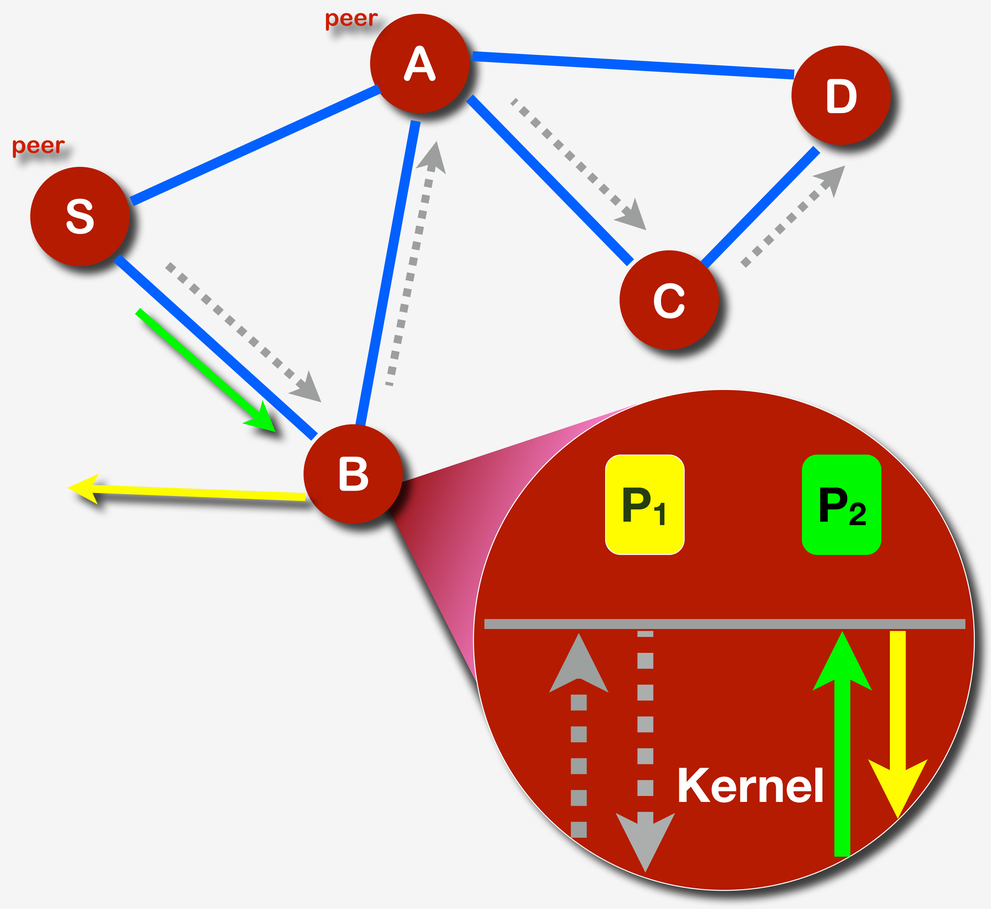
Surendar Chandra, University of Notre Dame

**Operating System**

manage system resources among schedulable entities

**Peer computing**

federate spare resources of independently owned peers




## Scenario 1: Ad hoc networks

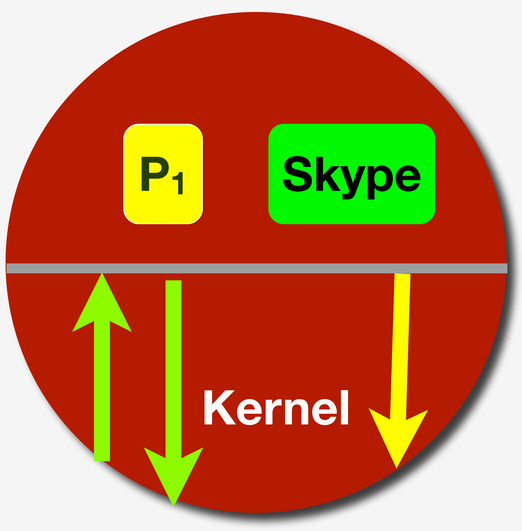
**Operating System**

Fair allocation of CPU, I/O, storage resources for processes $P_1$, $P_2$

**Peer computing**

Route S→B→A→C→D based on bandwidth/ latency/energy..

OS always routes with no spare capacity notions
- higher priority than for $P_1$ and $P_2$
- e.g., WinXP 2GHz laptop, Real media 256 kbps: 80% kernel CPU load for forwarding (interrupt overhead)




## Scenario 2: Peer to peer systems

**Operating System**

Manage resources (e.g., battery) for processes $P_1$ and Skype

**Peer computing - Skype**

Local user: Use other peers for VOIP calls

Peer computing: Route and relay calls

Lowering Skype process priority: unacceptable solution
- makes realtime application unusable for local user
- requests eventually serviced to lower priority process

OS does not control spare resource allocation, all requests are eventually serviced. Peers expected to be nice and limit requests (~ tit-for-tat levels)

➡Our approach: peer requests treated as second class citizens

- Solution: Less than best effort scheduling
  - Abundant resources - no difference
  - Resource constrained - peers receive no service, even if idle

Challenge: Balance local resource control & peer computing needs