Contemporary Operating Systems are not ready for Peer Computing
Surendar Chandra, University of Notre Dame

Managing system resources among schedulable entities

**Scenario 1: Ad hoc networks**
- Fair allocation of CPU, I/O, storage resources for processes P₁, P₂
- Route S→B→A→C→D based on bandwidth/latency/energy..
- OS always routes with no spare capacity notions:
  - higher priority than for P₁ and P₂
  - e.g., WinXP 2GHz laptop, Real media 256 kbps: 80% kernel CPU load for forwarding (interrupt overhead)

**Scenario 2: Peer to peer systems**
- Manage resources (e.g., battery) for processes P₁ and Skype
- 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