Enabling an Energy-Efficient Future Internet Through Selectively Connected End Systems

Mark Allman (ICSI)
Ken Christensen (USF)
Bruce Nordman (LBNL)
Vern Paxson (ICSI)

ACM SIGCOMM HotNets
November 2007

“The devil opened up his case and he said: I'll start this show. And fire flew from his fingertips as he resined up his bow.”
Motivation

• Studies have found ..... 

• 67% of office desktop computers fully powered after work hours [14]

• Average residential computer is on 34% of the time [15]

• Half the time no one is actively using the machine [15]

• Possible energy savings: $0.8-2.7 billion per year in the US alone [5]
Motivation (cont.)

- Why are these machines fully powered?
- Sporadic, occasional access:
  - User remote access
  - Administrative access (patches, backups, etc.)
  - Service provider access (set-top boxes, VoIP systems, etc.)
- Preservation of network state
Motivation (cont.)

• Underlying reason: our networking principles

• Our architecture assumes connected hosts

• Disconnectedness is dealt with as a problem
Related Work

- More energy-efficient hardware
- Disruption-tolerant networking
- One-off retrofits to today’s protocols
- Tiered power usage in wireless devices

- We take an architectural approach to the problem
Selective Connectivity

Connected

Traditional Internet

Disconnected

DTNs

Selective Connectivity
Example

- Consider an IM client
- Helps powered-down hosts by re-connecting after a host returns from sleep
- Our goal: to enable reception of messages while machine is selectively connected
- Perhaps even from a select set of “buddies”
- I.e., while sleeping the machine retains its *standing* in the network
Network Chatter

• Naive approach: wake machine when engaged

• Problem: hosts are continuously engaged for various reasons
  • Some important
  • Some not
Network Chatter (cont.)

- Computer not previously engaged in the network receives 6 pps over a 12+ hour period on a campus network [7]
- Over 20 network protocols detected
Network Chatter (cont.)

• Assess “chatter” on internal LBNL networks
• Take a 60 second slice starting at 3:18 AM from each of the 72 traces in our collection
• 4.49 pps on average (across traces)
• Various types: backup traffic, Windows services, NFS, NTP, DHCP, SSH
Network Chatter (cont.)

• How much of this chatter reflects communication with a fully powered machine?

• We look for two-way flows
  • Based on MAC address

• 66 of our 72 slices include two-way flows
  • Median of 3 two-way flows per trace
Architecture

- Initial set of new architectural components / concepts
- Could be wrong
- Surely incomplete
Assistants

• Perform routine and mundane operations on hosts’ behalf

• Keep state alive by responding to keep-alives

• Vet incoming traffic to allow only “important” activity to wake a host

• Inform remote hosts to “re-try”
Exposing State

• Expose the level of connectivity across protocol stack and possibly to peers

• E.g., exposing a host’s reason for being selectively connected might enable peer to also move to a reduced-energy state

• Tussle: exposing energy state may expose too much information
Evolving Soft State

• Soft state is one of the architectural successes of the Internet

• Maintaining soft state across selectively connected hosts poses a problem

• Two possible approaches:
  • *Proxyable State*: maintenance of state by assistant
  • *Limbo State*: recognition of distinction between “inexplicably gone” and “asleep”
Host-based Control

- We want to leave how selectively connected hosts are seen by others as a *policy decision*

- E.g., what is exposed to which peers

- E.g., what tasks are delegated

- E.g., what events should wake the host
Application Primitives

• Could we design general application primitives to aid selective connectivity?

• E.g., a generalized keep-alive that goes beyond a binary answer

• E.g., a way to share a list of files the host makes available on a p2p network

• Perhaps there are not a set of primitives, but we would need to provide a program that encodes our needed functionality to an assistant
Security

• Security issues cut across our thinking

• Many questions:
  • How can tasks be securely delegated?
  • How does a peer know an assistant has authority to act on behalf of a host or app?
  • How do we layer our use of crypto to expose information needed by an assistant without exposing sensitive data?
Final Thoughts

• We are early in our thinking of the issues
• We likely don’t have all the right models
  • That’s why we’re here!

• While energy savings has been our driver we think the resulting components could well be useful in other contexts
  • E.g., mobile hosts
Questions? Comments?