Modern Internet architecture, technology & philosophy

Advanced Internet Services
Dept. of Computer Science
Columbia University

Henning Schulzrinne

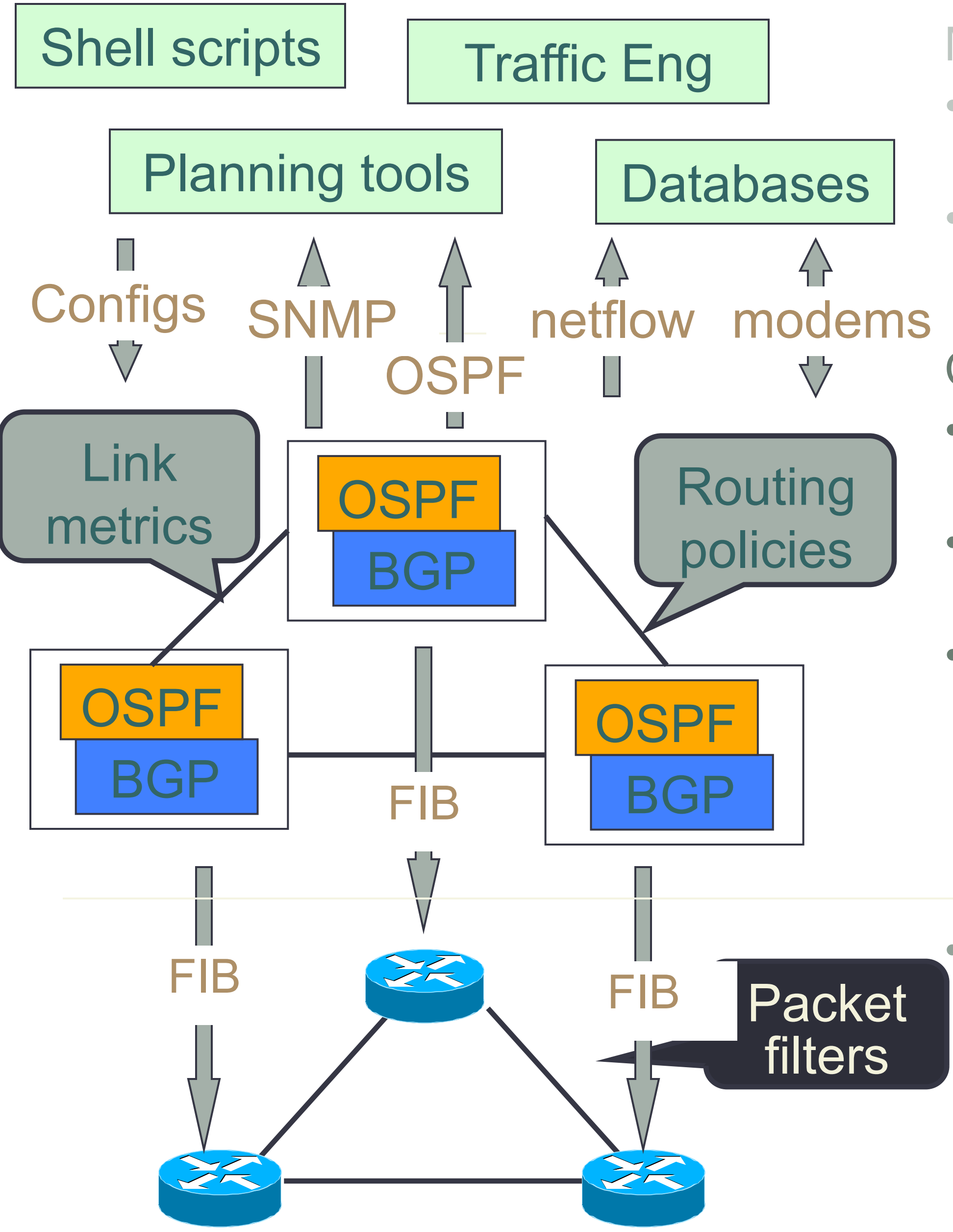
Spring 2015
02/09/2015

Key concepts

- The concept of network planes: data, control, management
- Centralized vs. distributed computing
  - properties and trade-offs
- Unitary vs. public-private Internet split
  - how do network addresses reflect network structure?
- Network protocols
  - similarities and differences to APIs
  - the notion of a contract
  - introduction to layering

NETWORK ARCHITECTURE

Inside a Single Network



Management Plane

- Figure out what is happening in network
- Decide how to change it

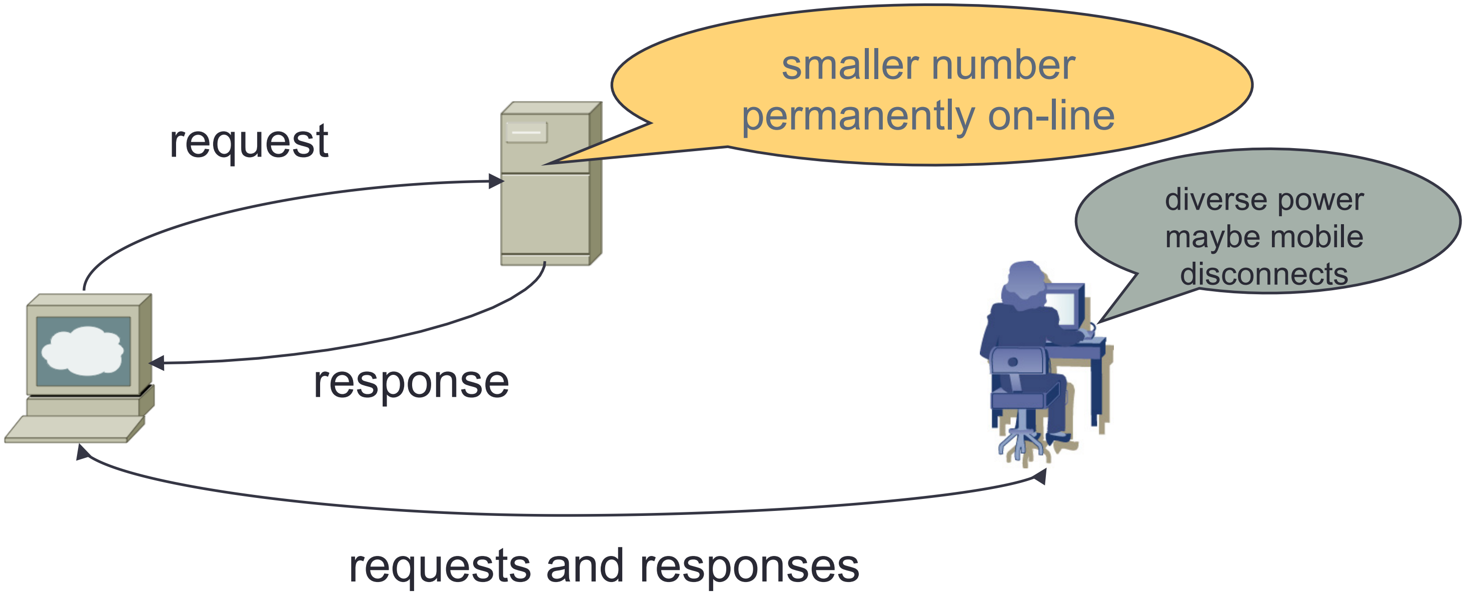
Control Plane

- Multiple routing processes on each router
- Each router with different configuration program
- Huge number of control knobs: metrics, ACLs, policy

- Data Plane
  - Distributed routers
  - Forwarding, filtering, queuing
  - Based on FIB or labels

Server-based vs. peer-to-peer

- Hard to define precisely
  - resource sharing?
  - no central control?
- All Internet applications are peers in a network sense


request
smaller number permanently on-line
diverse power maybe mobile disconnects
response
requests and responses

Defining peer-to-peer systems

Each peer must act as both a client and a server.

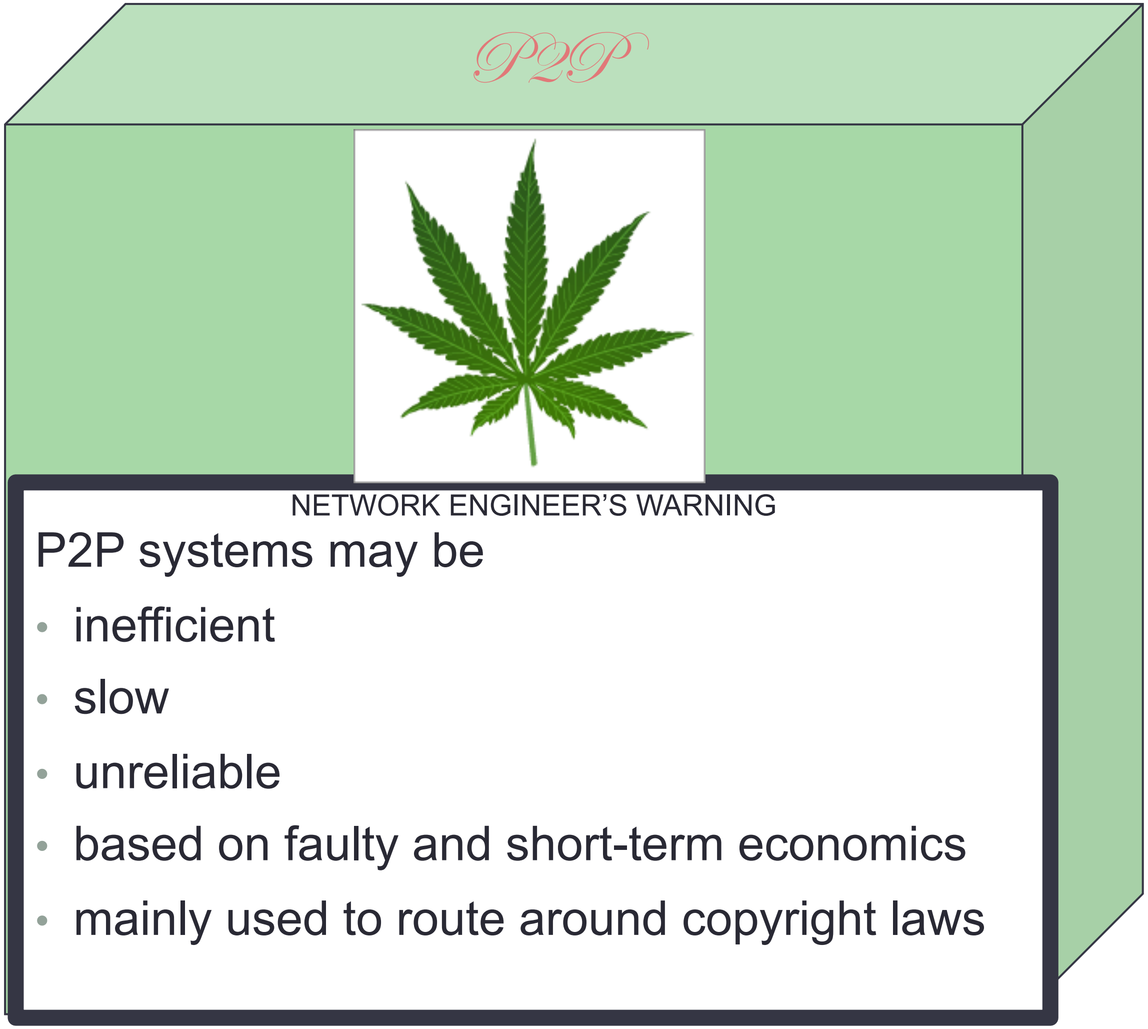
Peers provide network, computational or storage resources for ***other*** peers.

Self-organizing and scaling.

1 & 2 are not sufficient:

- DNS resolvers provide services to others
- Web proxies are both clients and servers
- SIP B2BUAs are both clients and servers

P2P systems are …



NETWORK ENGINEER’S WARNING

P2P systems may be

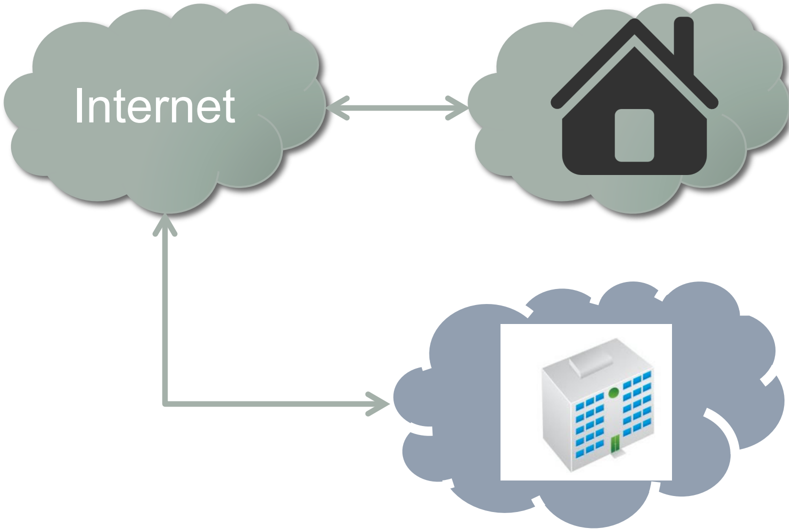
- inefficient
- slow
- unreliable
- based on faulty and short-term economics
- mainly used to route around copyright laws

Distributed vs. centralized

Property	Distributed computation	Centralized
Terms	P2P, fog computing, Blockchain, TOR, mesh networks	client-server, cloud computing
Who owns resources?	end users (mostly)	enterprise, IaaS, PaaS providers
Isolation	necessary	processes, hypervisor
Virtualization	fog computing	mostly
Functionality	storage, restricted computation	generic (IaaS) or language environment (PaaS)
Advantages	somebody else is paying heat homes for free! limit legal exposure	management trust
Challenges	trust (Tor, P2P) - byzantine free riders failure management	

Aside: alternatives to the big-I model

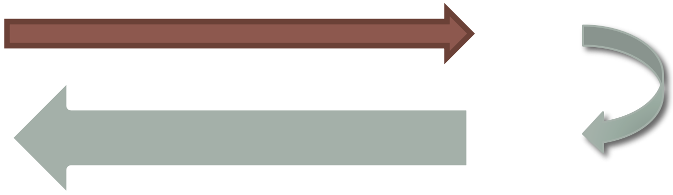
- Assumes end-to-end (logical) single network
- But reality messier
  - split address spaces (NATs)
  - UDP and TCP port numbers
  - middle boxes
  - cf. PBX extensions, mail stops, in-office mail delivery, PO boxes
- Architectural themes
  - separate "public" from "private" network
  - address space? tunneling?
  - IPv6 addressing



PROTOCOLS & LAYERS

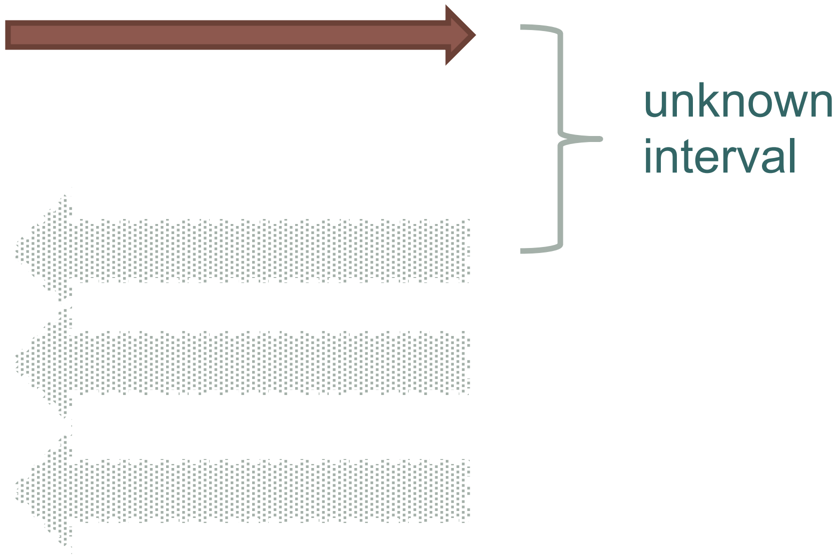
Fundamental interaction patterns

- Request-response
  - paced by requestor



- Continuous media
  - paced by sender

- Events
  - publish/subscribe



Internet protocols

- Protocols support these applications:
  - data plane
    - HTTP, ftp data part, SMTP, IMAP, POP, NFS, SMB, RTP
  - control plane
    - identifier mapping (id → id, id → data)
      - ARP, DNS, LDAP
    - configuration (= specialized version of identifier → data)
      - DHCP, LoST
    - session setup
      - RTSP, SIP, ftp control; RSVP, NSIS
    - routing
      - BGP, OSPF, RIP
  - management plane
    - SNMP, NETCONF (YANG), SNMP, netflow
- May be integrated into one protocol or general service function ("middleware"?)

Layers vs. planes - examples

Layer (5)	Data plane	Control plane	Management plane
Application	HTTP	DNS	SNMP
Transport	TCP	UDP or TCP	UDP or TCP
Internet	IPv4, IPv6		
Link	Ethernet, cellular link, …		
Physical	fiber, radio		

Protocols

- Protocols define format & order of messages sent and received among network entities
    - and actions taken on message transmission or receipt
- Often includes notions of time
    - what happens if there is no response?
- Similar to Application Programming Interfaces (APIs)
    - size_t fwrite (const void * ptr, size_t size, size_t count, FILE * stream);
    - differences?
- Can also consider a “contract”
    - “if I provide you X, you will provide Y”

Why layering?

- Perform functions once
  - upper layers rely on lower layers
  - in theory (see: “end-to-end principle”)
- Common in engineering and society
  - postal system, operating systems & other APIs, buildings, …
  - but not always formal or deep
  - model of a (legal) contract
    - “*The elements of a contract are "offer" and "acceptance" by "competent persons" having legal capacity who exchange "consideration" to create "mutuality of obligation.”* (Wikipedia)
- Change implementation without affecting relying parties
  - minimize communications, “information hiding”, “isolation”
  - “black box”
- Topological, economic and administrative scoping
  - single *physical* connection technology
  - single vs. multiple *administrative* domains