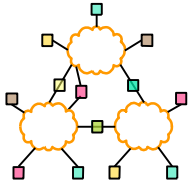


Computer Networks

(Graduate level)

Lecture 5: **Internetworking Design Principles**

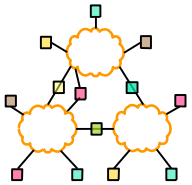
University of Tehran

Dept. of EE and Computer Engineering

By:

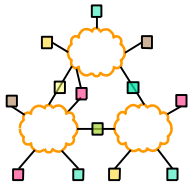
Dr. Nasser Yazdani

Network Design



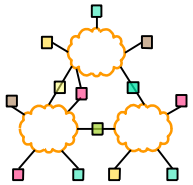
- Network Design principle
- How to determine split of functionality
  - Across protocol layers
  - Across network nodes
- Assigned Reading
  - End-to-end Arguments in System Design
  - Design Philosophy of the DARPA Internet Protocols
  - Architectural Consideration for a New Generation of Protocols
  - Tussle in Cyberspace
  - Rethinking the Design of the Internet: The End-to-End Arguments vs. the Brave New World
  - The End-to-End Argument and Application Design: The Role of Trust
  - A Protocol for Packet Network Intercommunication
  - IP Next Generation Overview
  - Chapt. 4 of the book

Goals



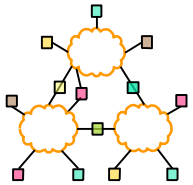
0. Connect existing networks
   - initially ARPANET and ARPA packet radio network
1. Survivability
   - ensure communication service even in the presence of network and router failures
2. Support multiple types of services
3. Must accommodate a variety of networks
4. Allow distributed management
5. Cost effective
6. Allow host attachment with a low level of effort
7. Allow resource accountability

Possible solutions



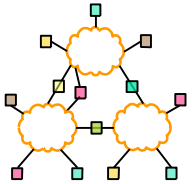
- Reengineer and develop one global packet switching network standard
  - not economically feasible
  - not deployable
- Decision: packet switching
  - Existing networks already were using this technology
- Have every host implement the protocols of any network it wants to communicate with
  - Complexity/node = O(n)
  - $O(n^2)$ global complexity

Internetworking



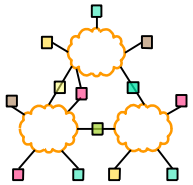
- Communication between networks.
- Problems & Challenges
  - Different Networking technologies (Heterogeneity).
  - So many Networks (Scaling).
- Some terminologies:
  - "internetworking" refer to an arbitrary collection of connected networks.
  - "Internet" the global internetwork.
  - "Network" either directly connected or switched network using any LAN technology such as Ethernet, Token ring, ATM, etc.

Connect Existing Networks .....



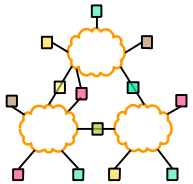
- Existing networks: ARPANET and ARPA packet radio
- Many differences between networks
  - Address formats
  - Performance – bandwidth/latency
  - Packet size
  - Loss rate/pattern/handling
  - Routing
  - .....

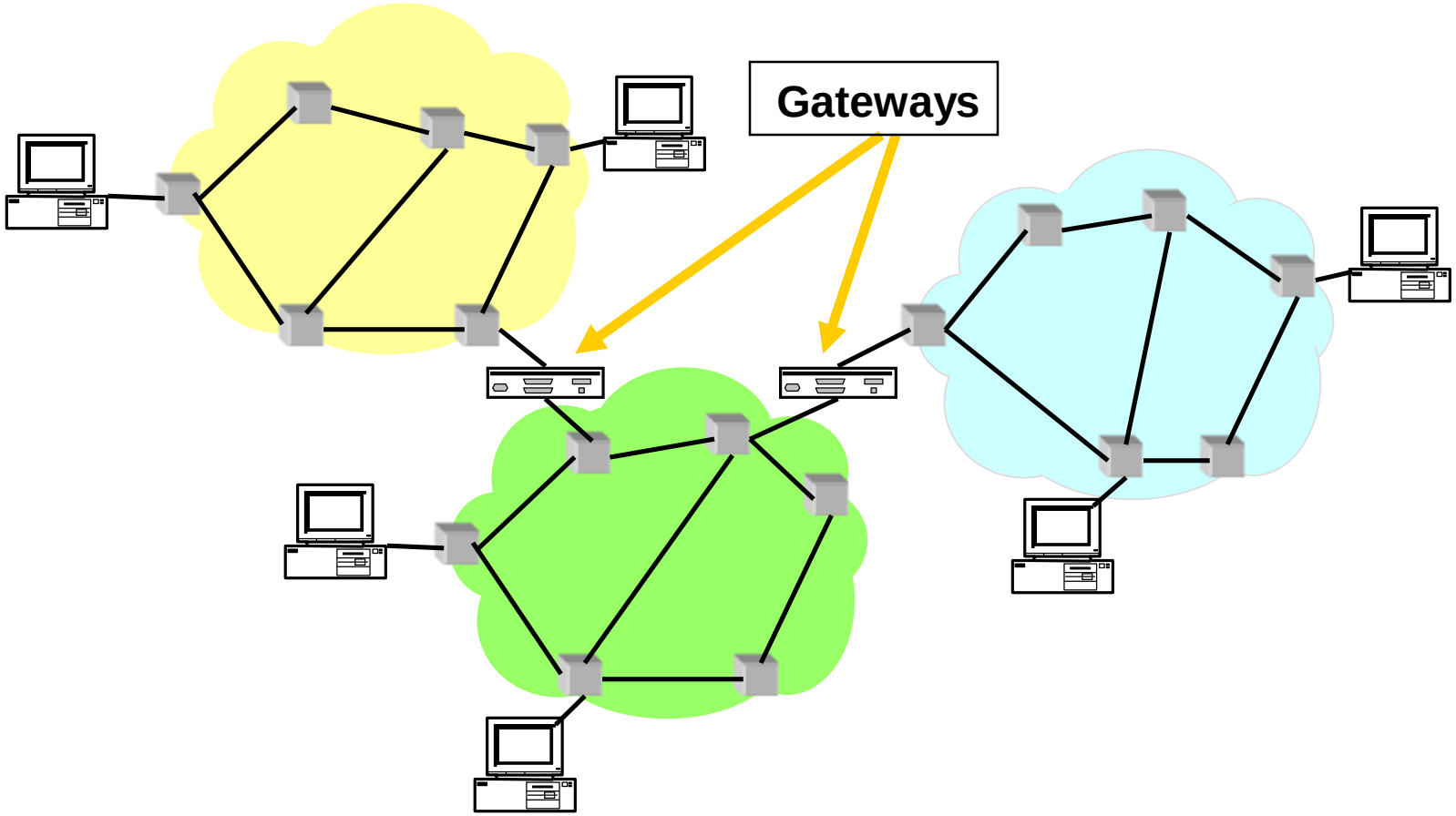
Gateway Alternatives



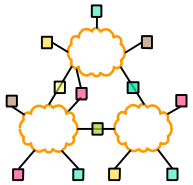
- Translation
  - Difficulty in dealing with different features supported by networks
  - Scales poorly with number of network types (N^2 conversions)
- Standardization
  - **"IP over everything" (Design Principle 1)**
  - Minimal assumptions about network
  - Hourglass design

Solution



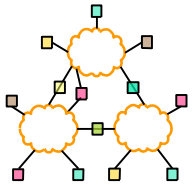


Solution

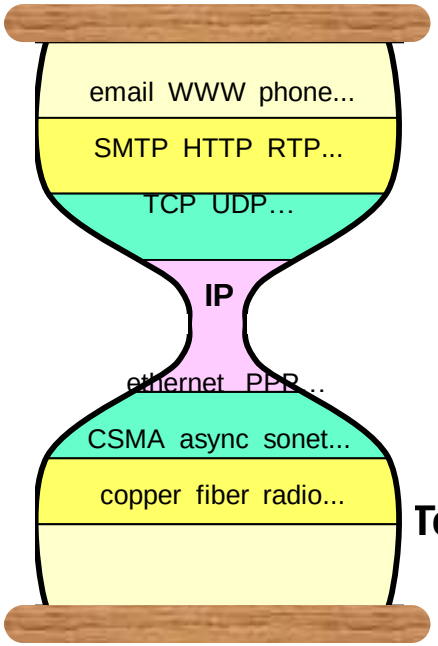


- **Add an extra layer**: internetworking layer
  - hosts:
    - understand one network protocol
    - understand one physical/link protocol
  - Routers/Switchses/gateways:
    - understand one network protocol
    - understand the physical/link protocols of the networks they gateway
  - Complexity to add a node/network: O(1) with respect to number of existing nodes

IP Hourglass



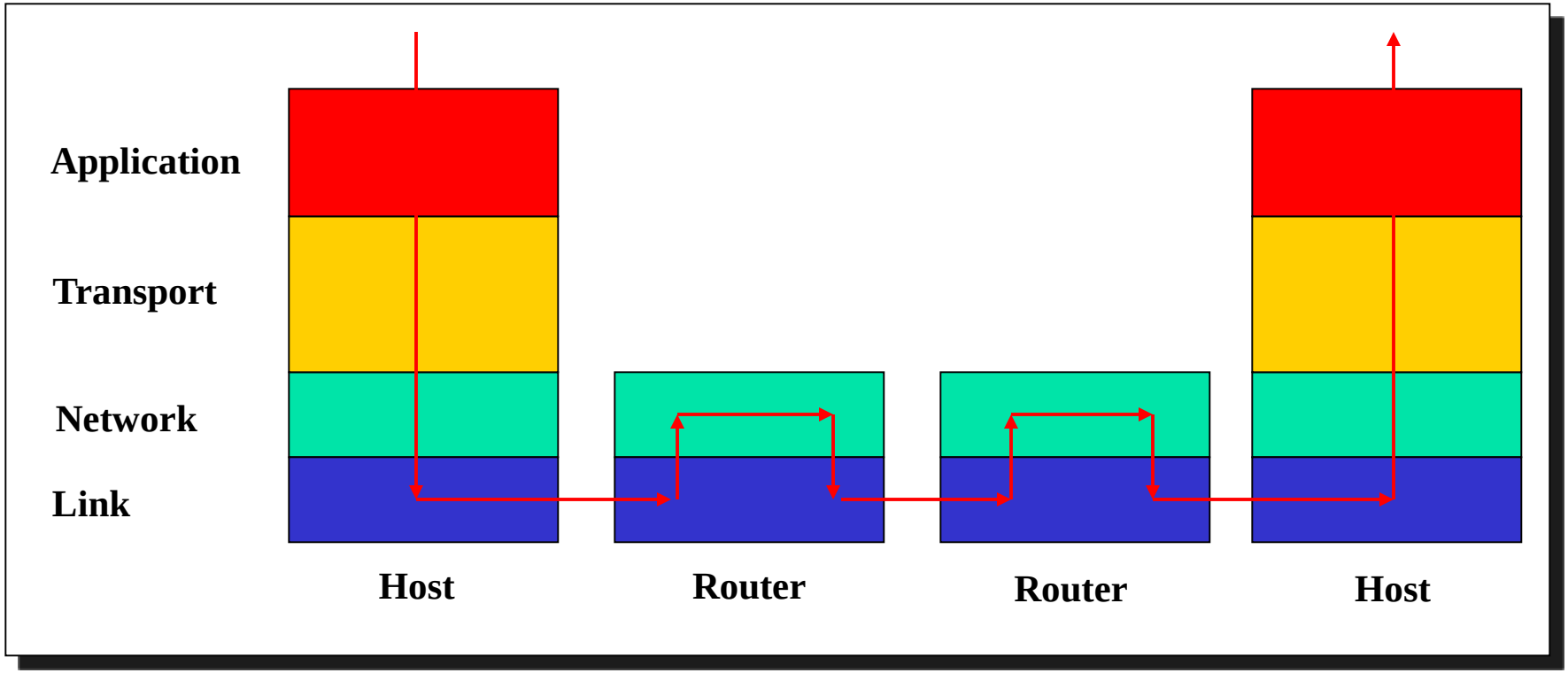
- Need to interconnect many existing networks
- Hide underlying technology from applications
- Decisions:
  - Network provides minimal functionality
  - "Narrow waist"



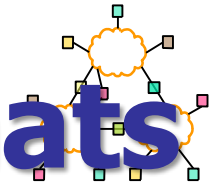
Tradeoff: **No assumptions, no guarantees.**

IP Layering (Principle 2)

- Relatively simple modle

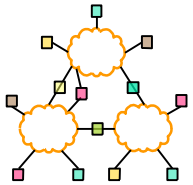


Different Address Formats



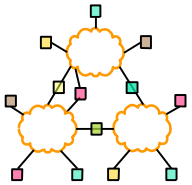
- Options:
  - Map one address format to another. Why not?
  - Provide one common format
    - map lower level addresses to common format
- Common Address Format:
  - Initially: 8b network 16b host 24b total
  - Before Classless InterDomain Routing (CIDR):
    - 7b/24b, 14b/16b, or 21b/8b 32b total
  - After CIDR: Arbitrary division 32b total
  - NAT: 32b + 16b simultaneously active
  - IPv6: 128b total

Different Packet Sizes



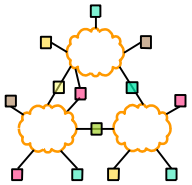
- Need to define maximum packet size
- Options:
  - Take the minimum of the maximum packets sizes over all networks
  - Implement fragmentation/reassembly
    - Flexibility to adjust packet sizes as new technologies arrive
    - IP: fragment at routers, reassemble at host
    - Why not reassemble at routers?
  - Still stuck with 1500B as de facto maximum

Other Challenges



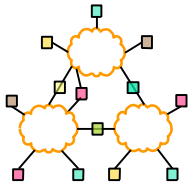
- Errors → require end-to-end reliability
  - Thought to be rarely invoked, but necessary
- Different (routing) protocols → coordinate these protocols
- Accounting
  - Did not envision script kiddies
- Quality of Service
  - Not addressed

Key problem: Heterogeneity



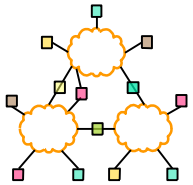
- Not described as such in paper
- Heterogeneous addressing conventions => common addressing scheme
- Heterogeneous MTU => fragmentation
- Heterogeneous queuing/transit delays =>adaptive "timing procedures"
- Heterogeneous reliability => end-to-end recovery mechanisms
- Heterogeneous control/status info => common error/coordination mechanism

Principle 3



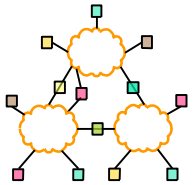
- **Best effort delivery**
- All packets are treated the same
- Relatively simple core network elements
- Building block from which other services (such as reliable data stream) can be built
- Contributes to scalability of network

Principle 4



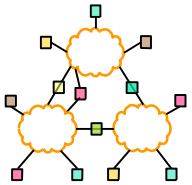
- **Fate sharing**
- Critical state only at endpoints
- Only endpoint failure disrupts communication
- Helps survivability

Principle 5



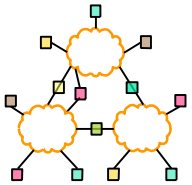
- **Soft-state**
  - Announce state
  - Refresh state
  - Timeout state
- Penalty for timeout – poor performance
- Robust way to identify communication flows
  - Possible mechanism to provide non-best effort service
- Helps survivability

Principle 6



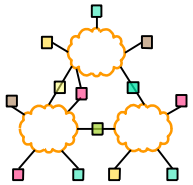
- **Decentralization**
- Each network owned and managed separately
- Will see this in BGP routing especially

Principle 7



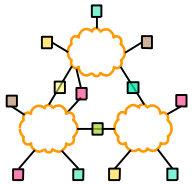
- Be conservative in what you send and liberal in what you accept
  - Unwritten rule
- Especially useful since many protocol specifications are ambiguous
- E.g. TCP will accept and ignore bogus acknowledgements

Survivability



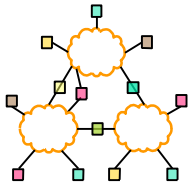
- Continue to operate even in the presence of network failures (e.g., link and router failures)
  - As long as the network is not partitioned, two endpoint should be able to communicate, moreover, any other failure (excepting network partition) should be transparent to endpoints
- Decision: maintain state only at end-points (fate-sharing)
  - Eliminate the problem of handling state inconsistency and performing state restoration when router fails
- Internet: stateless network architecture

2. Types of Services



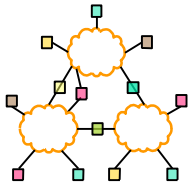
- Add UDP to TCP to better support other types of applications
  - e.g., "real-time" applications
- Probably main reason for separating TCP and IP
- Provide datagram abstraction: lower common denominator on which other services can be built
  - service differentiation considered (ToS header bits)
  - was not widely deployed (why?)

3. Variety of Networks



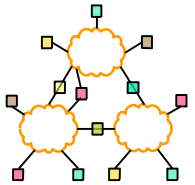
- Very successful
  - because the minimalist service; it requires from underlying network only to deliver a packet with a "reasonable" probability of success
- …does not require:
  - reliability, in-order delivery, single delivery, QoS guarantees
- **IP over everything**
  - Then: ARPANET, X.25, DARPA satellite network..
  - Now: ATM, SONET, WDM, PPP, USB, 802.11b, GSM, GPRS, DSL, cable modems, power lines

Key Advantages



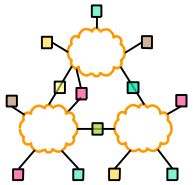
- The service can be implemented by a large variety of network technologies
- Does not require routers to maintain any fined grained state about traffic. Thus, network architecture is
  - **Robust**
  - **Scalable**

Other Goals



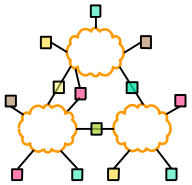
- Allow distributed management
  - each network can be managed by a different organization
  - different organizations need to interact only at the boundaries
  - doesn't work so well for routing, accounting
- Cost effective
  - sources of inefficiency
    - header overhead
    - retransmissions
    - Routing
    - Congestion control
    - .....

Other Goals (Cont)



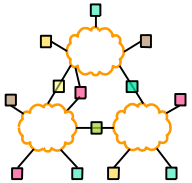
- Low cost of attaching a new host
  - not a strong point → higher than other architecture because the intelligence is in hosts (e.g., telephone vs. computer)
    - Moore's law made this moot point, both <$100
  - bad implementations or malicious users can produce considerably harm (remember fate-sharing?)
    - DDoS possibly biggest threat to Internet
- Accountability
  - very little so far

Services



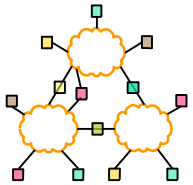
- At network layer provides one simple service: best effort datagram (packet) delivery
- Only one higher level service implemented at transport layer: reliable data delivery (TCP)
  - performance enhancement; used by a large variety of applications (Telnet, FTP, HTTP)
  - does not impact other applications (can use UDP)
- Everything else implemented at application level

What About Other Services?



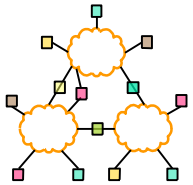
- Multicast?
- Quality of Service (QoS)?
- Security, virus?
- Trust?
- ….

IP Design Weaknesses



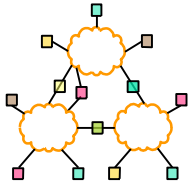
- Greedy sources aren't handled well
- Weak accounting and pricing tools
- Weak administration and management tools
- Trust is problem now
- Incremental deployment difficult at times
  - Result of no centralized control
  - No more "flag" days
  - Are active networks the solution?

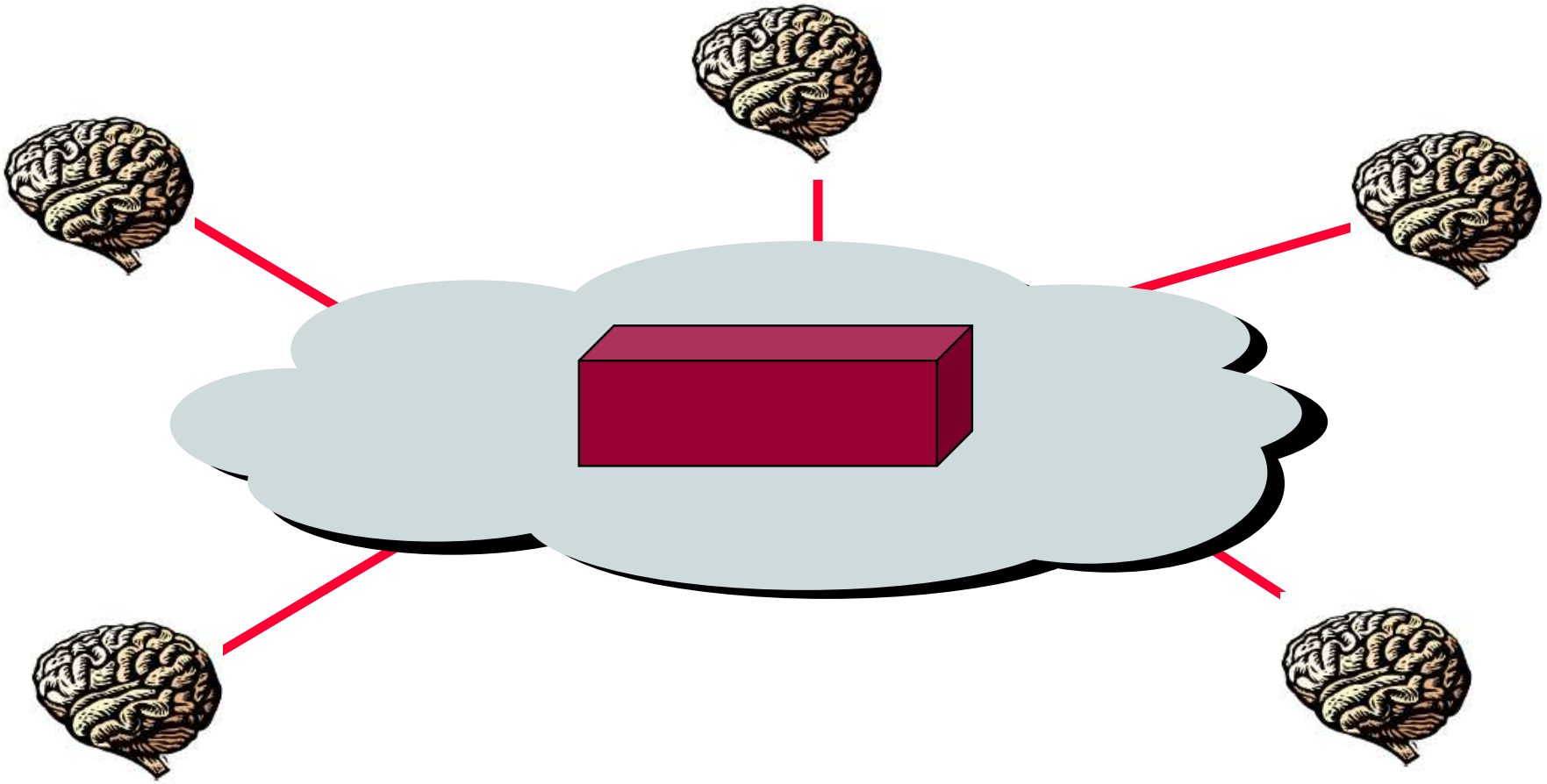
Summary: Minimalist Approach



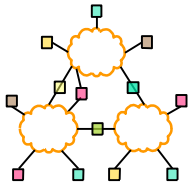
- Dumb network
  - IP provide minimal functionalities to support connectivity
    - Addressing, forwarding, routing
- Smart end system
  - Transport layer or application performs more sophisticated functionalities
    - Flow control, error control, congestion control
- Advantages
  - Accommodate heterogeneous technologies (Ethernet, modem, satellite, wireless)
  - Support diverse applications (telnet, ftp, Web, X windows)
  - Decentralized network administration

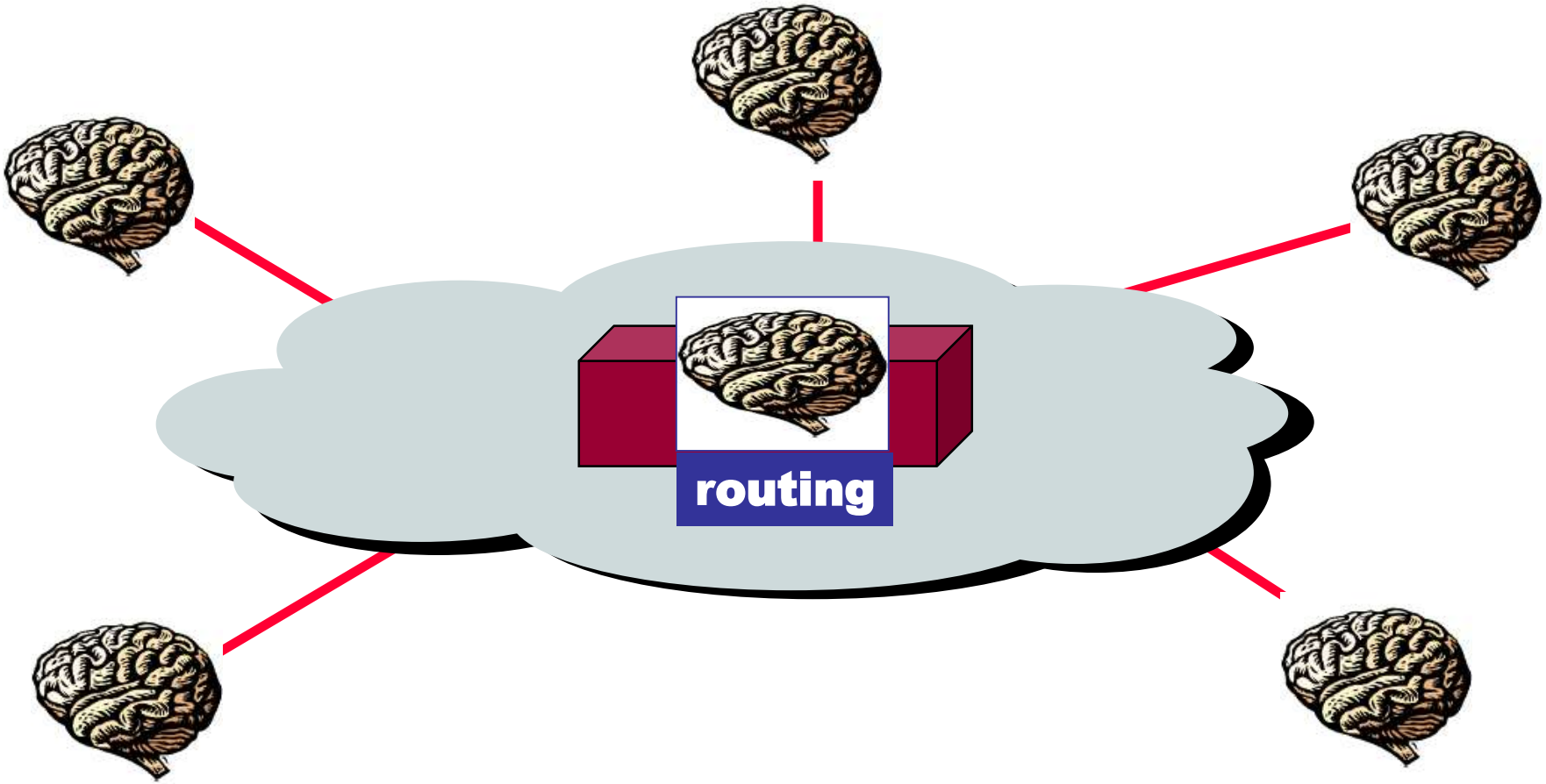
Common View of the IP Network



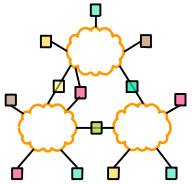


Needs some intelligence!



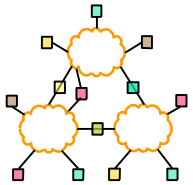


How IP was implemented?



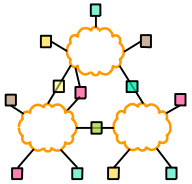
- IP addressed facing problems in different ways?
- For details read Paper: "A Protocol for Packet Network Intercommunication".
- Also see "IP Next Generation Overview "

Transmission Control Program



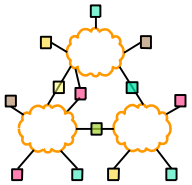
- Original TCP/IP (Cerf & Kahn)
  - no separation between transport (TCP) and network (IP) layers
  - one common header flow control, but not congestion control (why not?)
  - fragmentation handled by TCP
- Today's TCP/IP
  - separate transport (TCP) and network (IP) layer (why?)
  - split the common header in: TCP and UDP headers
  - fragmentation reassembly done by IP
  - congestion control

How is IP Design Standardized?



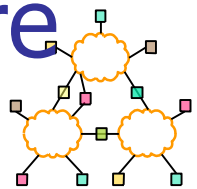
- IETF
  - Voluntary organization
  - Meeting every 4 months
  - Working groups and email discussions
- "We reject kings, presidents, and voting; we believe in rough consensus and running code" (Dave Clark 1992)
  - Need 2 independent, interoperable implementations for standard

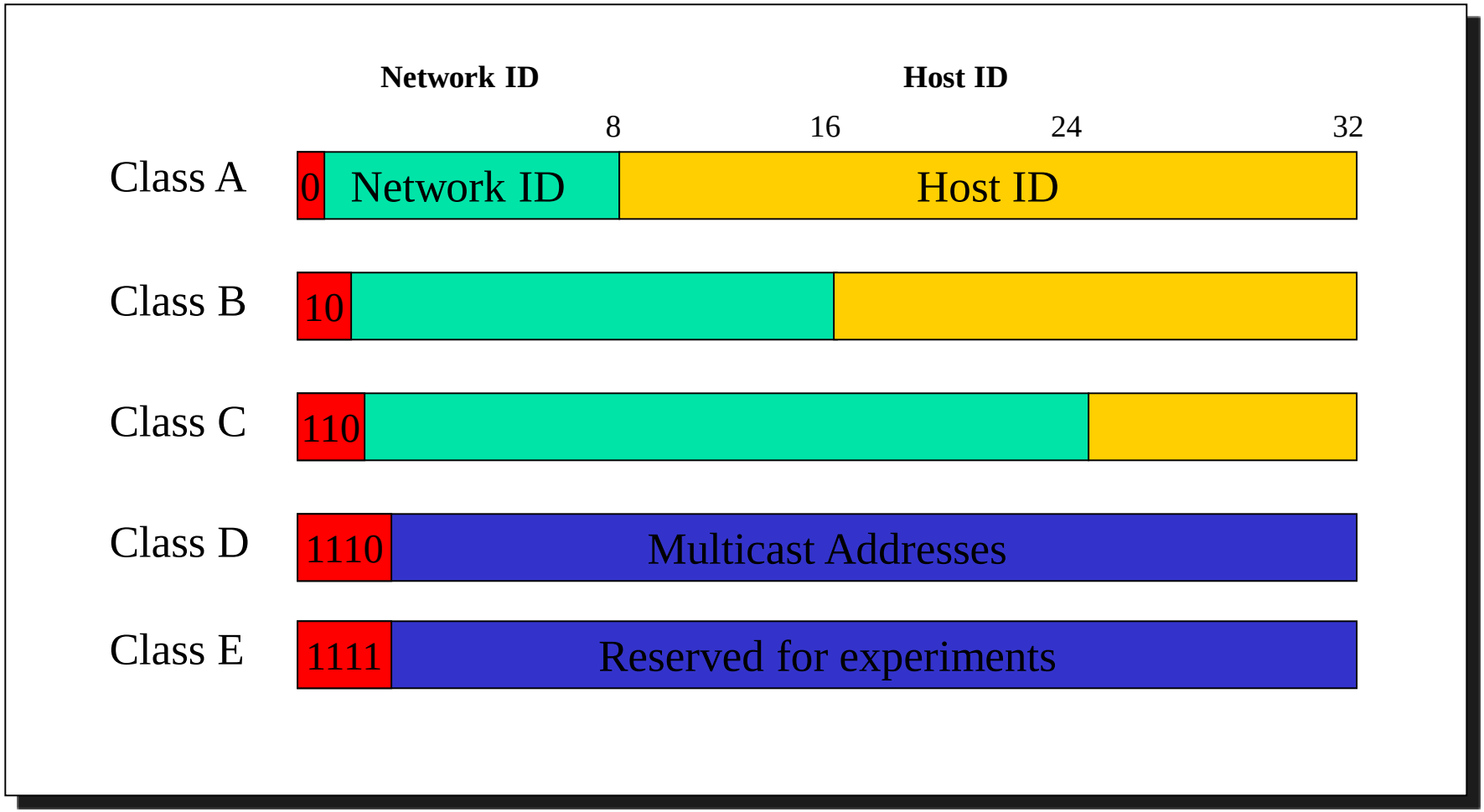
Problems in reality



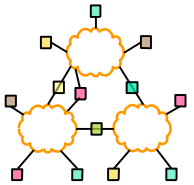
- IP address problem
- New requirements for services
  - QoS
  - Security, trust
- So many protocols, fast changing environments, …
  - SDN is a solution?

IP Address Classes (Some are Obsolete)



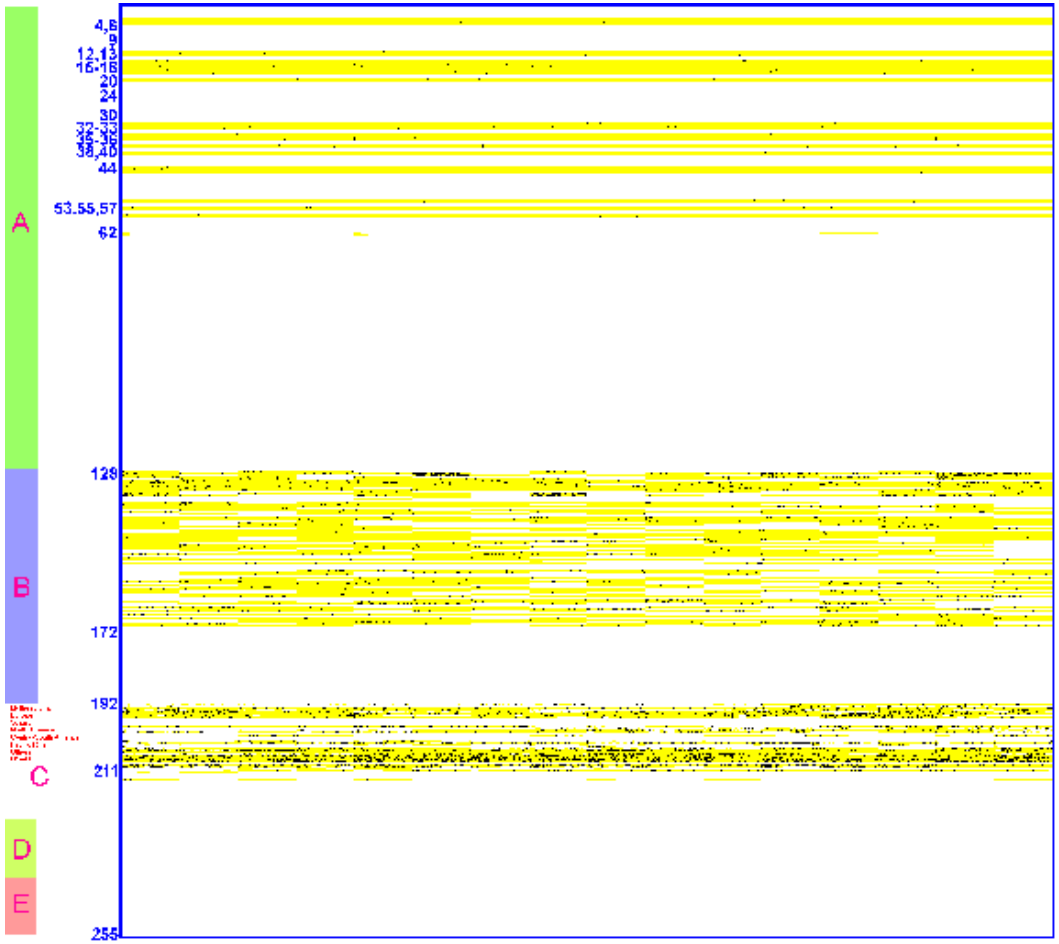


IP Address Problem (1991)



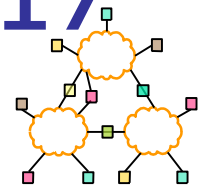
- Address space depletion
  - In danger of running out of classes A and B
- Why?
  - Class C too small for most domains
  - Very few class A – IANA (Internet Assigned Numbers Authority) very careful about giving
  - Class B – greatest problem

IP Address Utilization ('98)



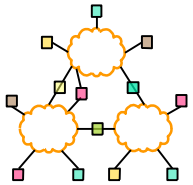
http://www.caida.org/outreach/resources/learn/ipv4space/

Subnet Addressing – RFC917 (1984)

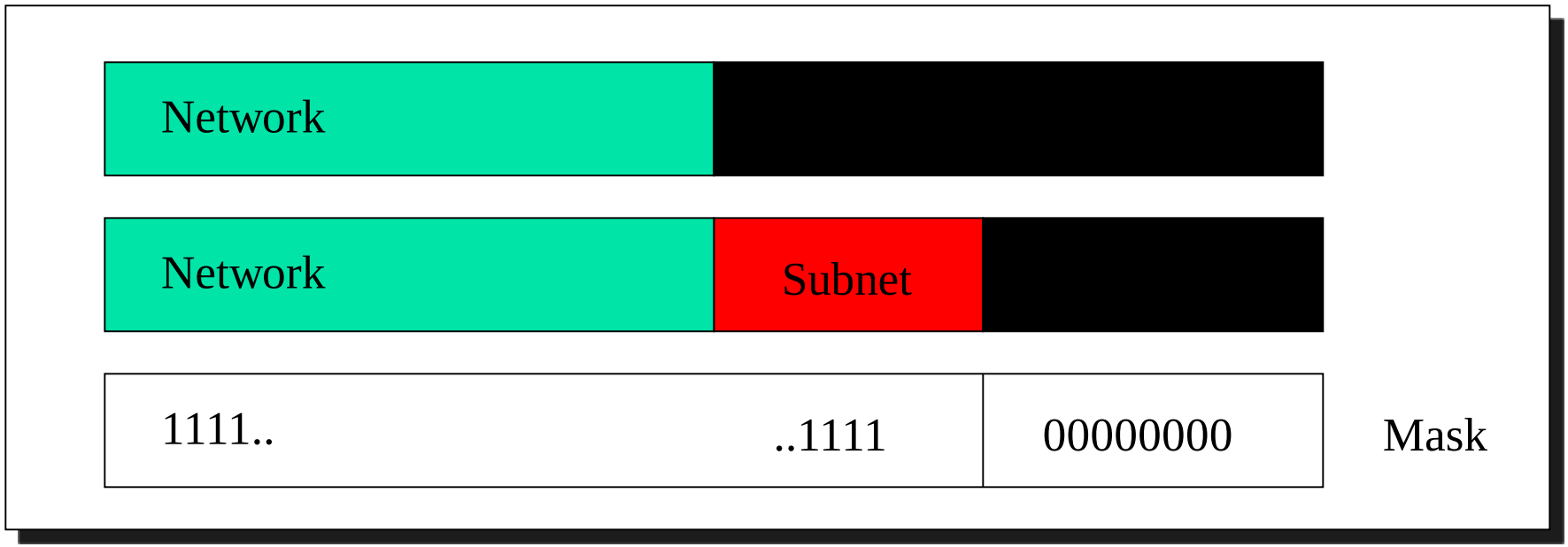


- For class B & C networks
- Very few LANs have close to 64K hosts
  - For electrical/LAN limitations, performance or administrative reasons
- Need simple way to get multiple "networks"
  - Use bridging, multiple IP networks or split up single network address ranges (subnet)
  - Must reduce the total number of network addresses that are assigned

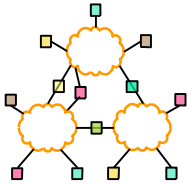
Subnetting



- Variable length subnet masks
  - Could subnet a class B into several chunks

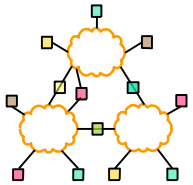


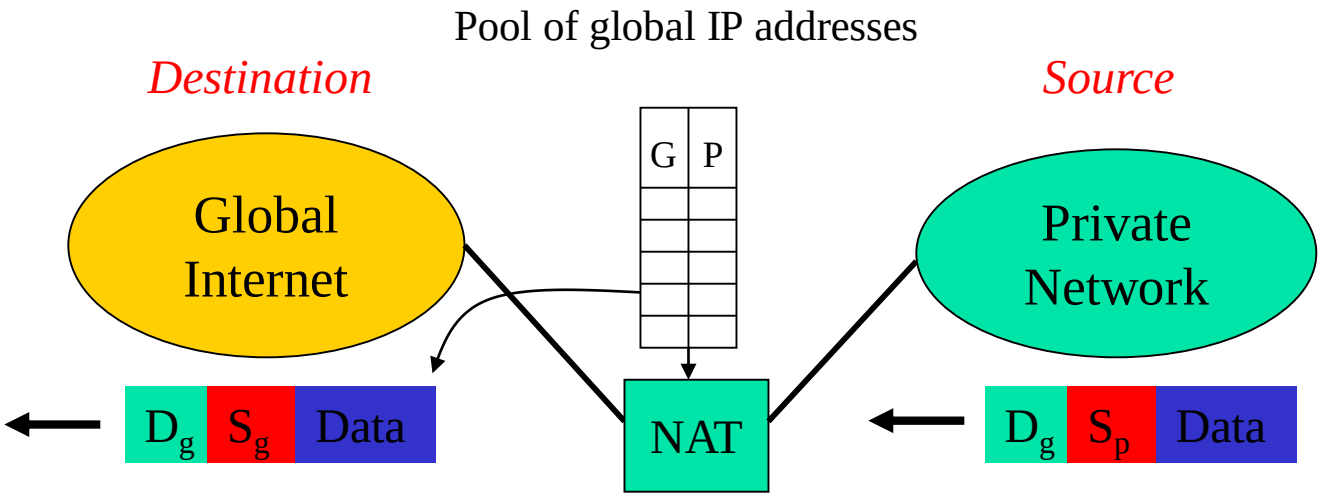
Network Address Translation (NAT)



- Sits between your network and the Internet
- Translates local network layer addresses to global IP addresses
- Has a pool of global IP addresses (less than number of hosts on your network)

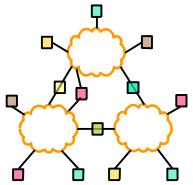
NAT Illustration





- **Operation: Source (S) wants to talk to Destination (D):**
  - Create S_g-S_p mapping
  - Replace S_p with S_g for outgoing packets
  - Replace S_g with S_p for incoming packets

Classless Inter-Domain Routing

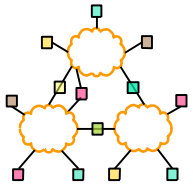


- Do not use classes to determine network ID
- Assign any range of addresses to network
  - Use common part of address as network number
  - e.g., addresses 192.4.16 - 196.4.31 have the first 28 bits in common. Thus, we use this as the network number
  - netmask is /28, /xx is valid for almost any xx
- Enables more efficient usage of address space (and router tables)

IP Address Utilization (‘06)

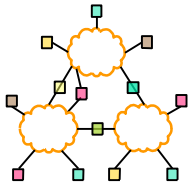
http://xkcd.com/195/

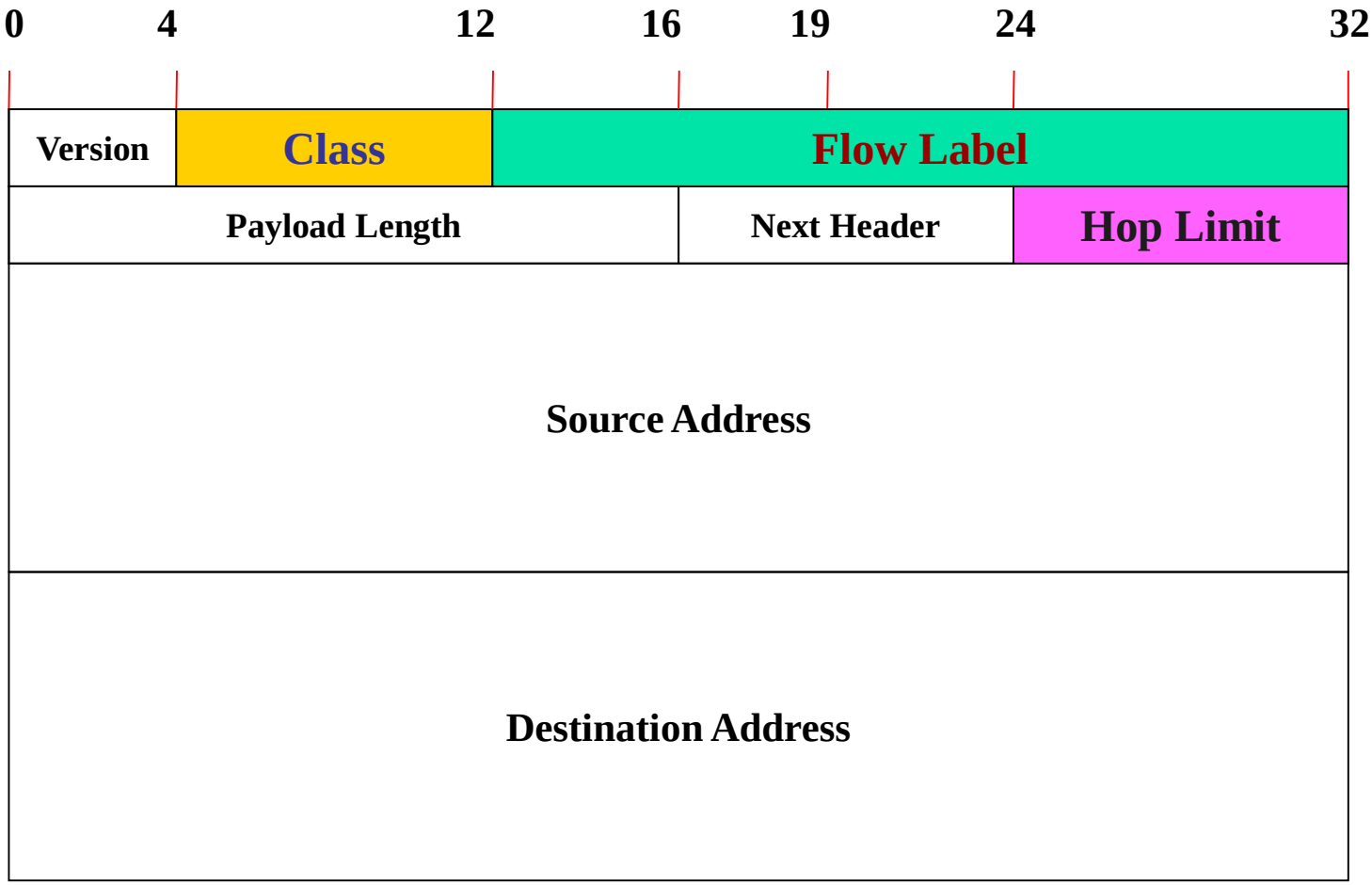
IPv6 Changes



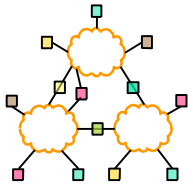
- Scale – addresses are 128bit
  - Header size?
- Simplification
  - Removes infrequently used parts of header
  - 40byte fixed size vs. 20+ byte variable
- IPv6 removes checksum
  - Relies on upper layer protocols to provide integrity
- IPv6 eliminates fragmentation
  - Requires path MTU discovery
  - Requires 1280 byte MTU

IPv6 Header



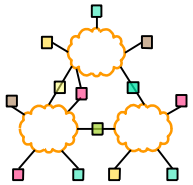

0
4
12
16
19
24
32
Version
Class
Flow Label
Payload Length
Next Header
Hop Limit
Source Address
Destination Address

IPv6 Changes



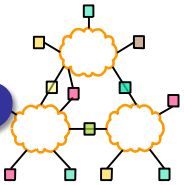
- TOS replaced with traffic class octet
- Flow
  - Help soft state systems
  - Maps well onto TCP connection or stream of UDP packets on host-port pair
- Easy configuration
  - Provides auto-configuration using hardware MAC address to provide unique base
- Additional requirements
  - Support for security
  - Support for mobility

IPv6 Changes



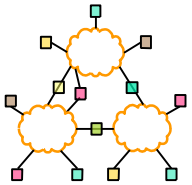
- Protocol field replaced by next header field
  - Support for protocol demultiplexing as well as option processing
- Option processing
  - Options are added using next header field
  - Options header does not need to be processed by every router
    - Large performance improvement
    - Makes options practical/useful

What About the Future?



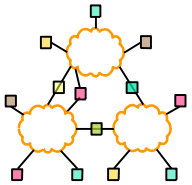
- Datagram not the best abstraction for:
  - resource management,accountability, QoS
- A new abstraction: flow?
- Routers require to maintain per-flow state (what is the main problem with this raised by Clark?)
  - state management

Devil's Advocate



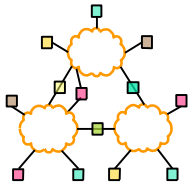
- Who cares about resource sharing?
  - 1974: cycles, storage, bandwidth expensive, people cheap
  - Today: resources cheap, people expensive
  - 1974: Share computer resources
  - Today: Communicate with people, access documents, buy, sell
- Does it still make sense to make processes the endpoint?
- Where do people and organizations fit into the ISO layering model?

End-to-End Argument (Principle 2)



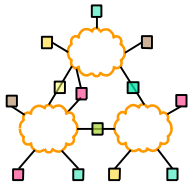
- Deals with where to place functionality
  - Inside the network (in switching elements)
  - At the edges
- Argument
  - There are functions that can only be correctly implemented by the endpoints do not try to completely implement these at them elsewhere
  - Can provide a partial form as performance enhancement
  - Guideline not a law

End-to-End Argument



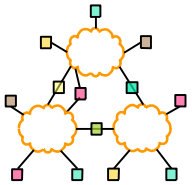
- Think twice before implementing a functionality that you believe that is useful to an application at a lower layer
- If the application can implement a functionality correctly, implement it a lower layer only as a performance enhancement

Example: Reliable File Transfer



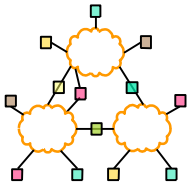
- Solution 1: make each step reliable, and then concatenate them
- Solution 2: end-to-end check and retry

Discussion



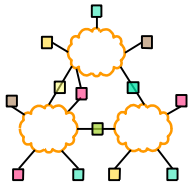
- Solution 1 not complete
  - What happens if the sender or/and receiver misbehave?
- The receiver has to do the check anyway!
- Thus, full functionality can be entirely implemented at application layer; no need for reliability from lower layers

Discussion



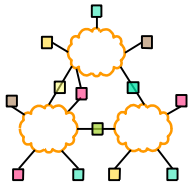
- Is there any need to implement reliability at lower layers?
- Yes, but only to improve performance
- Example:
  - Assume a high error rate on communication network
  - Then, a reliable communication service at data link layer might help

Trade-offs



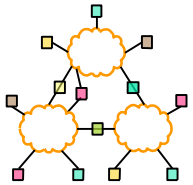
- Application has more information about the data and the semantic of the service it requires (e.g., can check only at the end of each data unit)
- A lower layer has more information about constraints in data transmission (e.g., packet size, error rate)
- Note: these trade-offs are a direct result of layering!

Rule of Thumb



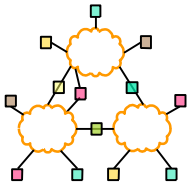
- Implementing a functionality at a lower level should have minimum performance impact on the application that do not use the functionality

Example: File Transfer



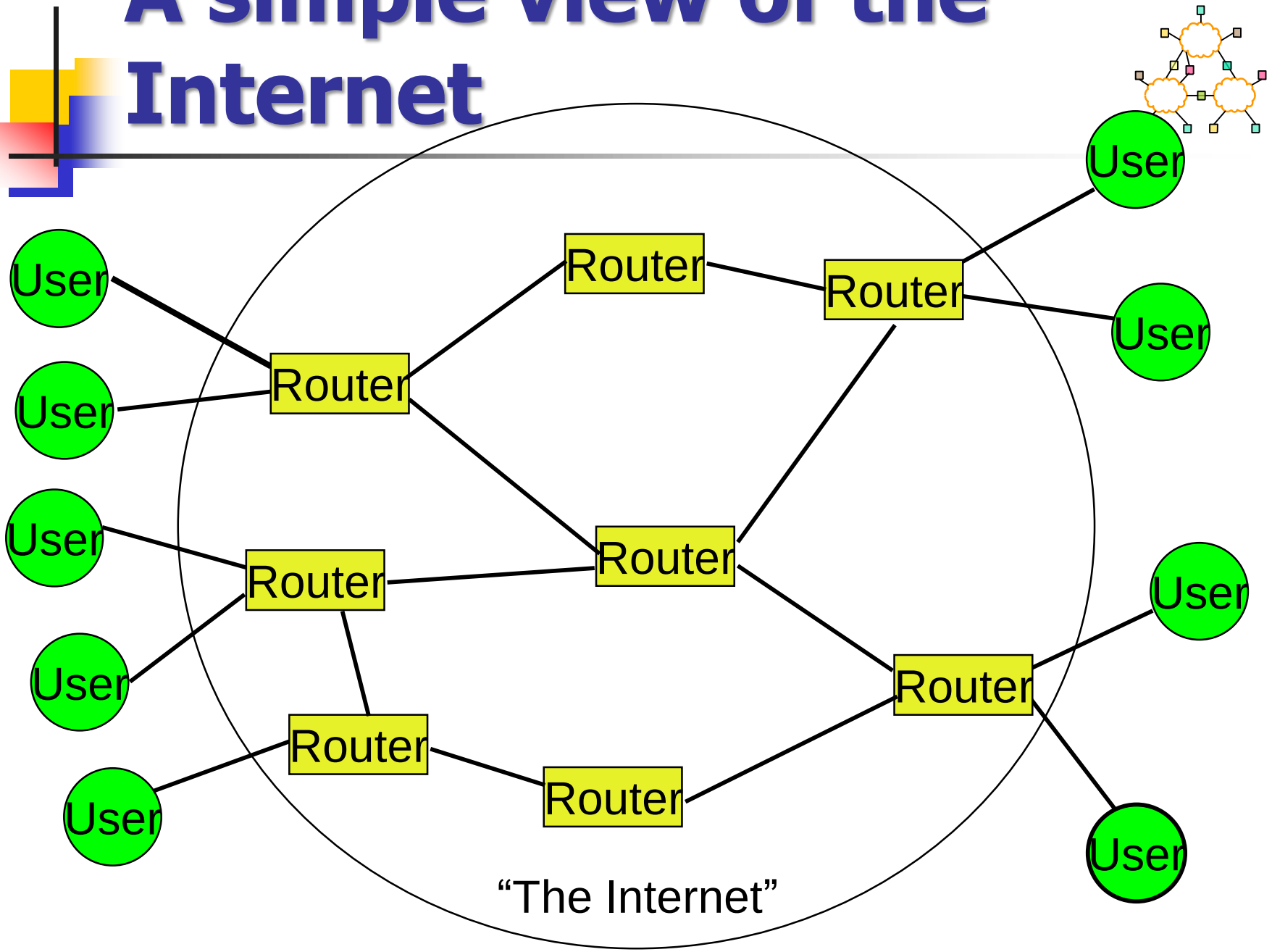
- Even if network guaranteed reliable delivery
  - Need to provide end-to-end checks
  - E.g., network card may malfunction
- If network is highly unreliable
  - Adding some level of reliability helps performance, not correctness
  - Don’t try to achieve perfect reliability!
- Is FTP like this?
  - TCP provides reliability between kernels not disks

Internet & End-to-End Argument

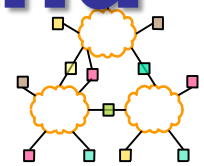


- Provides one simple service: best effort datagram (packet) delivery
- Only one higher level service implemented at transport layer: reliable data delivery (TCP)
  - Performance enhancement; used by a large variety of applications (Telnet, FTP, HTTP)
  - Does not impact other applications (can use UDP)
- Everything else implemented at application level
- Move functions "up and out".
- The network should be "as transparent as technology permits".

A simple view of the Internet

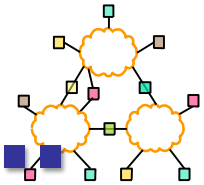


Benefits of the end to end arguments



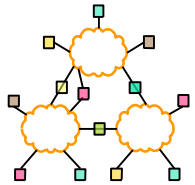
- User empowerment.
  - Run what you please.
- Flexibility in the face of unknown applications.
  - A network to hook computers together.
- Lower cost in core of network.
  - Eliminate special “features”.
  - Rely on edge-node equipment.
- More robust applications.
  - No unexpected failures of third-party nodes.

But in today's Internet...



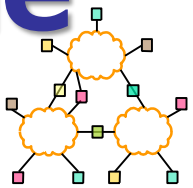
- There are "new" factors to consider:
  - Assured operation in an untrustworthy world.
  - More demanding applications.
    - Growing need for enhanced services.
  - Less sophisticated users.
    - User empowerment is a burden.
    - Does today's software really empower the user?
  - Third parties who want to intervene.
  - New sorts of end-node devices.
    - Appliances, not PCs.
  - The evolving role of the Internet Service Provider.

In the brave new world



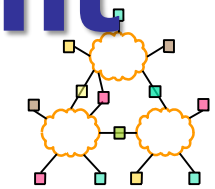
- The end to end model does not empower the ISP.
  - ISPs want to sell services, add value, and make money.
    - New network services (or not), protection, control of applications/content, accounting.
- The end to end model does not empower rights holders.
- The end to end model does not empower governments.
  - Control of content, taxation, consumer protection, law enforcement.
- The end to end model does not empower employers.
- The end to end model only empowers *certain* application makers

The E2E argument at the network level



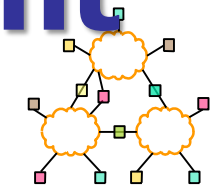
- At the network level:
  - More complex role for commercial ISPs
    - vertical integration of transport/QoS with apps.
    - control of what apps users can use.
    - filtering of unacceptable behavior.
  - Others (e.g., employers, universities) restrict activities.
    - Firewalls, filters
  - Governments propose controls in the net.
- The Internet is hard to find and control.
  - What *is* the Internet?

The end to end argument functions at two levels



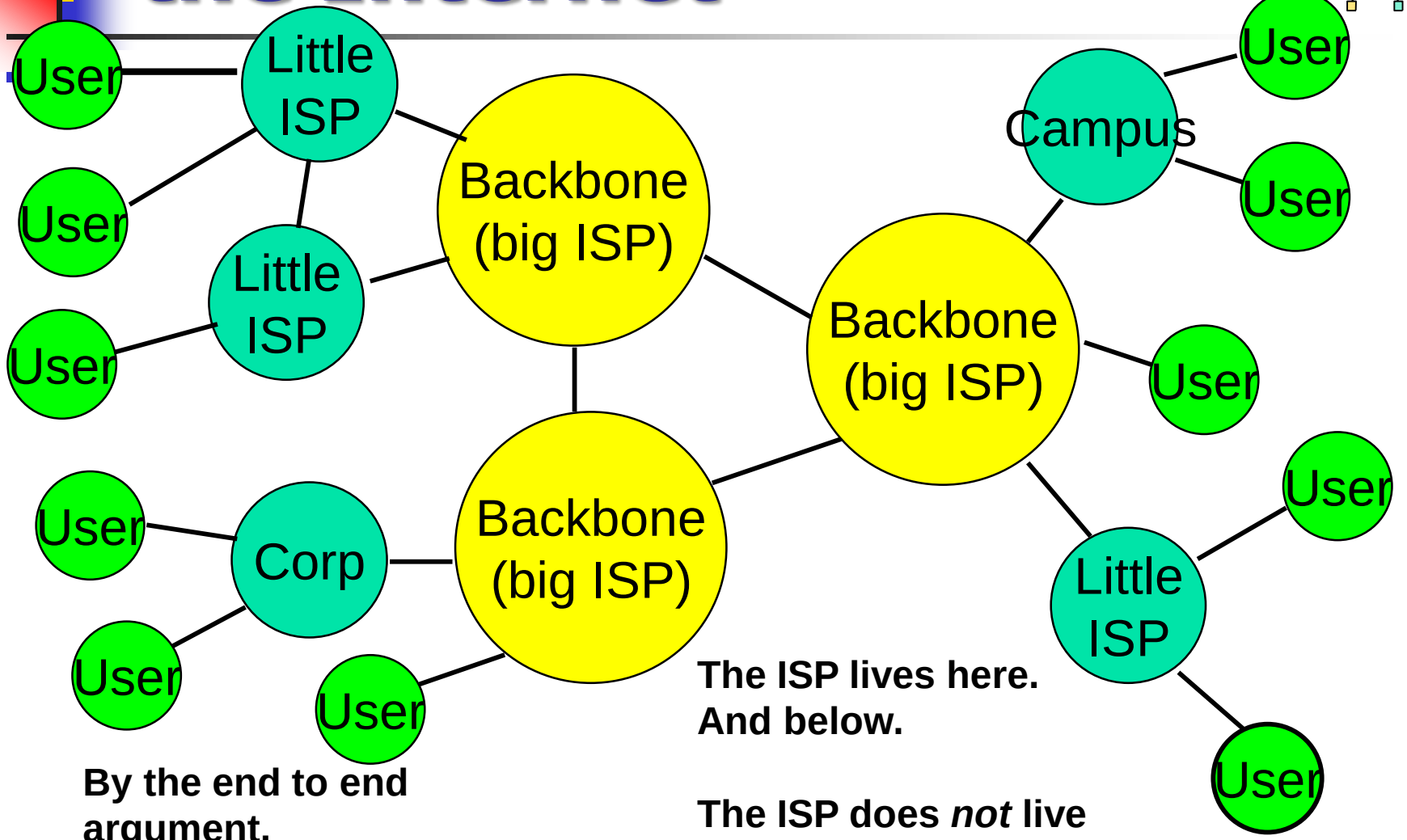
- At the "network" level:
  - Avoid putting constraining per-application functions into the core of the network.
  - Build general purpose services.
- At the "application" level:
  - Build applications in a way that makes them robust, easy to use, reliable, etc.
    - Simple approach: push software to the edge.
    - Perhaps more realistic: reason carefully about role of servers, trusted third parties, etc.
  - Increasing concern about the goal of making money?

The end to end argument at the application level



- At the application level:
  - Ease of use, mutual distrust, multi-party interactions may motivate servers and services other than at the end-points of the users.
    - Relays for content, trusted third parties.
    - How do we make these applications robust?
    - How do the parties know who they are talking to?
  - Appliance and devices motivate new designs.
- Both ISPs and ASPs (A = Application) want to "get into the action".
- ISP involvement will imply constraints on location/structure.

A more complex view of the Internet



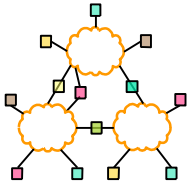
By the end to end argument, applications live at the edge.

The ISP lives here. And below.

The ISP does *not* live at the end-points.

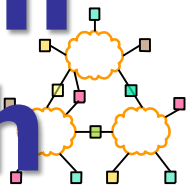
(They can try…)

Some ideas for moving forward



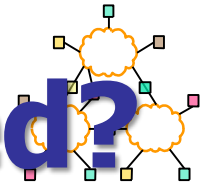
- Labels.
  - A compromise between autonomy and visibility of action.
- Distinction between private and public communication.
  - Accept that private communication is not restricted.
  - Focus on communication "to the public".
- New principles for application design.
  - Do not force an end-node implementation.
  - Allow the user to select an alternative.
  - A more sophisticated form of empowerment.
- Tolerance for experimentation.
  - That which is not forbidden is permitted?

Some "contradictions" in the end to end approach



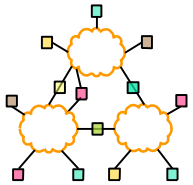
- How can we build a trustworthy network out of edge-devices that cannot be trusted?
- How can we have anonymity and accountability?
- How can the network control unacceptable behavior and still permit unknown applications?
  - Flexibility is critical if the Internet is to be a part of the computer world.
- Must find a balance among the goals.

What has really changed?



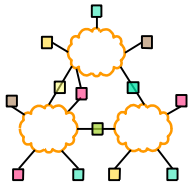
- A loss of trust among users.
  - Global communication with local trust.
- The need to factor in economic forces.
  - Role of commercial (and other) ISPs.
- The change in the nature of the user base.
  - Less sophisticated: ease of use, protection concerns
- Co-evolution of technology and law.
  - Complex cycle of evolution.
  - Geographic differences.
- Change in nature of innovation.
- Big players.

Summary: End-to-End Argument

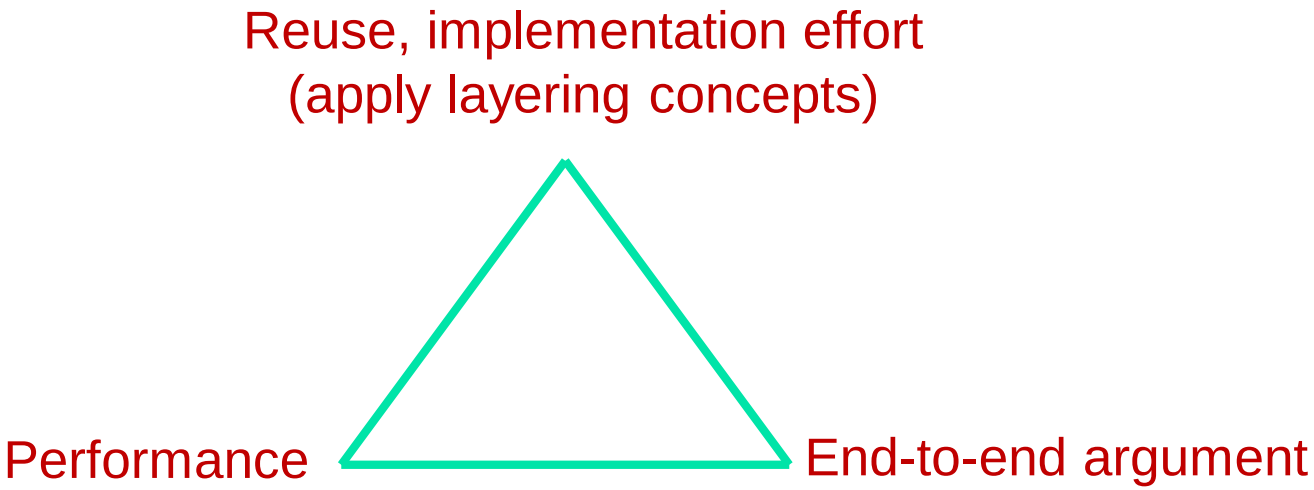


- If the application can do it, don't do it at a lower layer -- anyway the application knows the best what it needs
  - Add functionality in lower layers iff it is (1) used and improves performances of a large number of applications, and (2) does not hurt other applications
- Success story: Internet

Summary

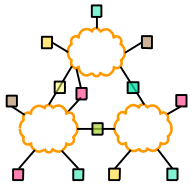


- Challenge of building a good (network) system: find the right balance between:



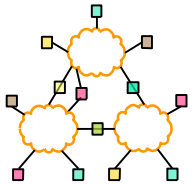
  - No universal answer: the answer depends on the goals and assumptions!

Problems over time



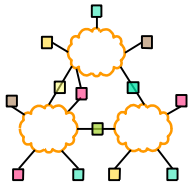
- Architecture developed in simpler times
  - Common goals, consistent vision
- • With success, the goals of different players have diverged– examples:
  - ISPs must talk to provide connectivity but are fierce competitors
  - Privacy of users vs. government's need to monitor
  - User's desire to exchange files vs. copyright owners
- • Must deal with the tussle between concerns in design

New Principles?



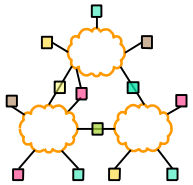
- Design for variation in outcome
  - Allow design to be flexible to different uses/results
- Isolate tussles
  - QoS designs uses separate ToS bits instead of overloading other parts of packet like port number
  - Separate QoS decisions from application/protocol design
- Provide choice: allow all parties to make choices on interactions
  - Creates competition
  - Fear between providers helps shape the tussle

Trust-to-Trust Principle



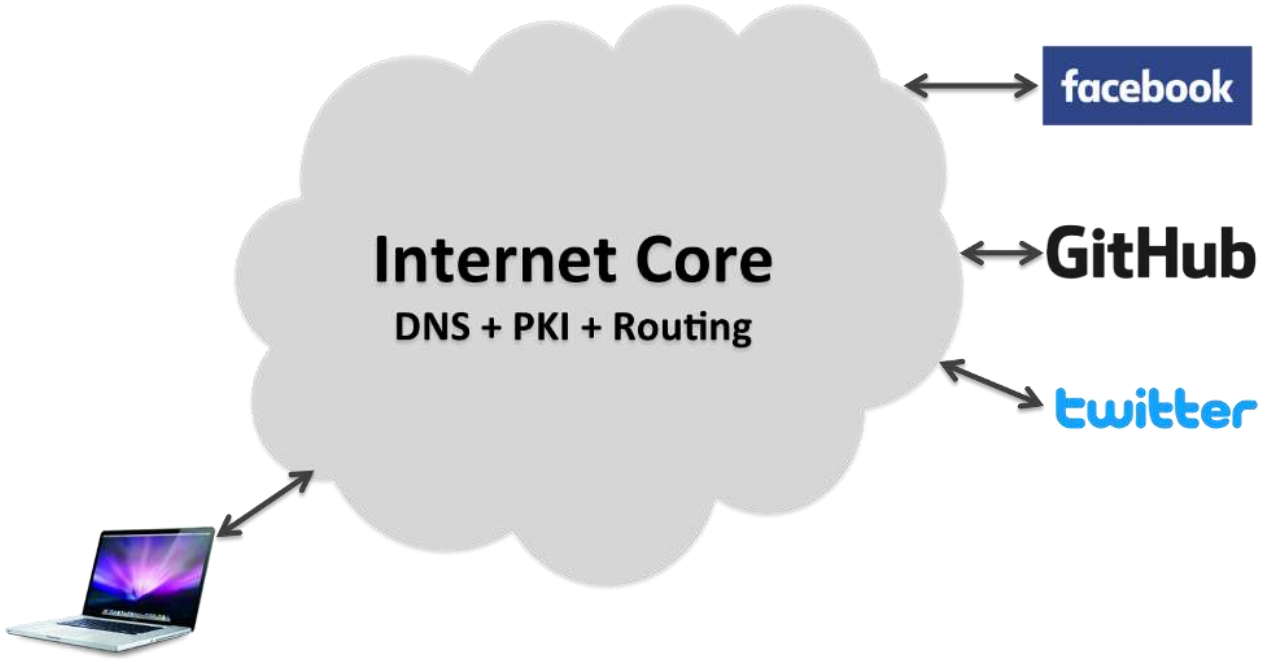
- Security and Trust are big issues today. To see a website, you have to trust 10 parties.
- Did they ignored them in the early design?
  - They thought about it. They were familiar with malware in 1970's
  - They make few mistakes
    - They did not know what the security means.
    - They assumed users can trust each others
    - And the end nodes are reliable.
  - They did not know how to model/modularize the security/trust.

Trust-to-Trust Principle



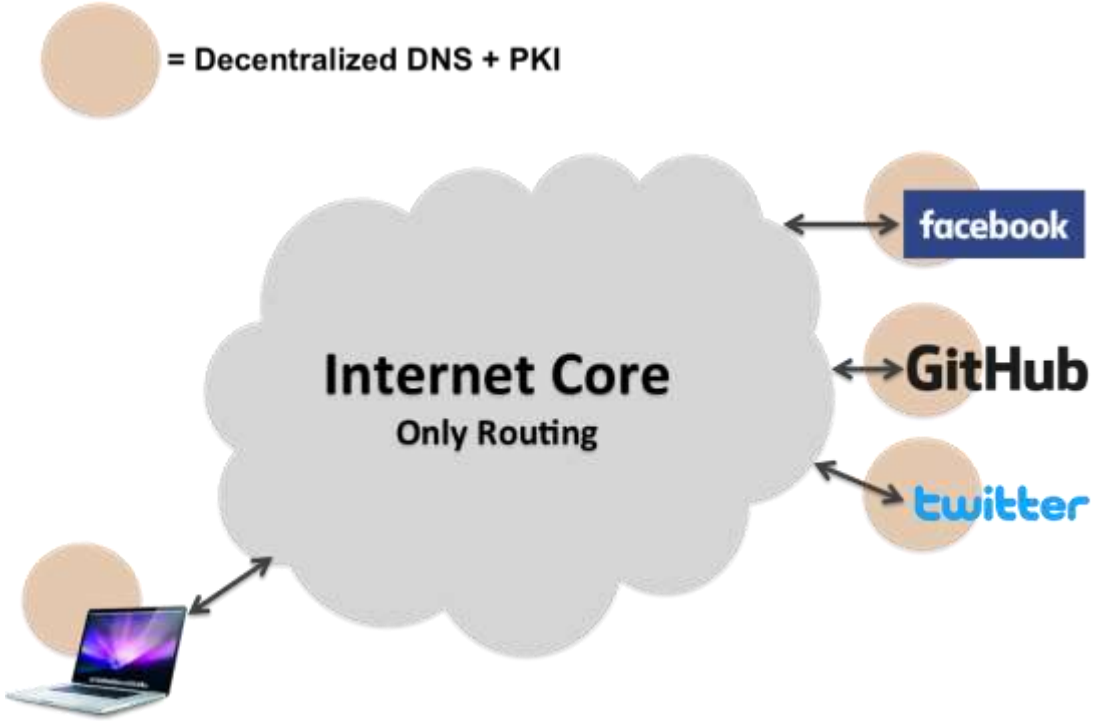
- Some basic question on the end point
  - Not everything in the end point
    - Cash server, mail server, Clouds, CDN
  - What about human in VoIP.
- Trust to Trust is a generalization of End to End principle.
  - The end user should have control over the trust decisions.
  - Move trust from the core of the network to the edges.

Trust to trust Internet



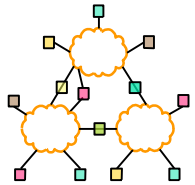
DNS and public Key (PKI) server in the network that must be trusted .

End to End Internet



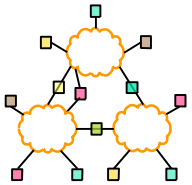
DNS and public Key (PKI) server in the points that must be trusted .

What do we face Today



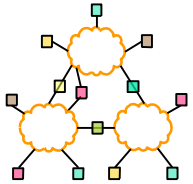
- Internet is a huge system
  - Very Complicated
    - Around 7000 RFCs
  - Difficult to modify/change
    - Not easy to fix bugs
    - Not easy to innovate
- Overwhelming demands for scale and new services
- Solution?
  - Complexity: new design?
  - Flexibility: move to software?
- Is SDN a solution?

NSF Programs



- Stagnation
  - 100x100 → Clean Slate Design
  - PlanetLab
  - Overcoming the Internet Impasse through Virtualization → GENI
- Internet architecture projects
  - Named Data Networking
  - MobilityFirst
  - eXpressive Internet Architecture

NSF FIND Project [Clark]



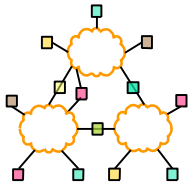
1988

1. Internet communication must continue despite loss of networks or gateways.
2. The Internet must support multiple types of communications service.
3. The Internet architecture must accommodate a variety of networks.
4. The Internet architecture must permit distributed management of its resources.
5. The Internet architecture must be cost effective.
6. The Internet architecture must permit host attachment with a low level of effort.
7. The resources used in the Internet architecture must be accountable.

2008

1. Security
2. Availability and resilience
3. Economic viability
4. Better management
5. Meet society's needs
6. Longevity
7. Support for tomorrow's computing
8. Exploit tomorrow's networking
9. Support tomorrow's applications
10. Fit for purpose (it works...)

NSF FIND Project [Clark]



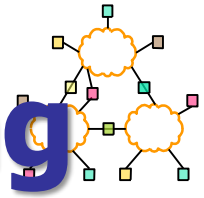
2008

1. Security
2. Availability and resilience
3. Economic viability
4. Better management
5. Meet society's needs
6. Longevity
7. Support for tomorrow's computing
8. Exploit tomorrow's networking
9. Support tomorrow's applications
10. Fit for purpose (it works...)

Internet Support

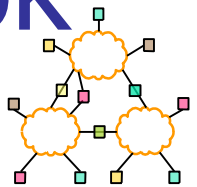
1. End2end integrity/confidentiality with encryption
2. Routing and TTL – but little else
3. None
4. None
5. None
6. Has been effective – but parts such as addressing
7. Was developed in very heterogeneous settings. Hard to predict
8. Mobile and optical have raised issues. Link failure and network management interactions could be better.
9. Done well but are important applications just ignored?
10. It does work ☺

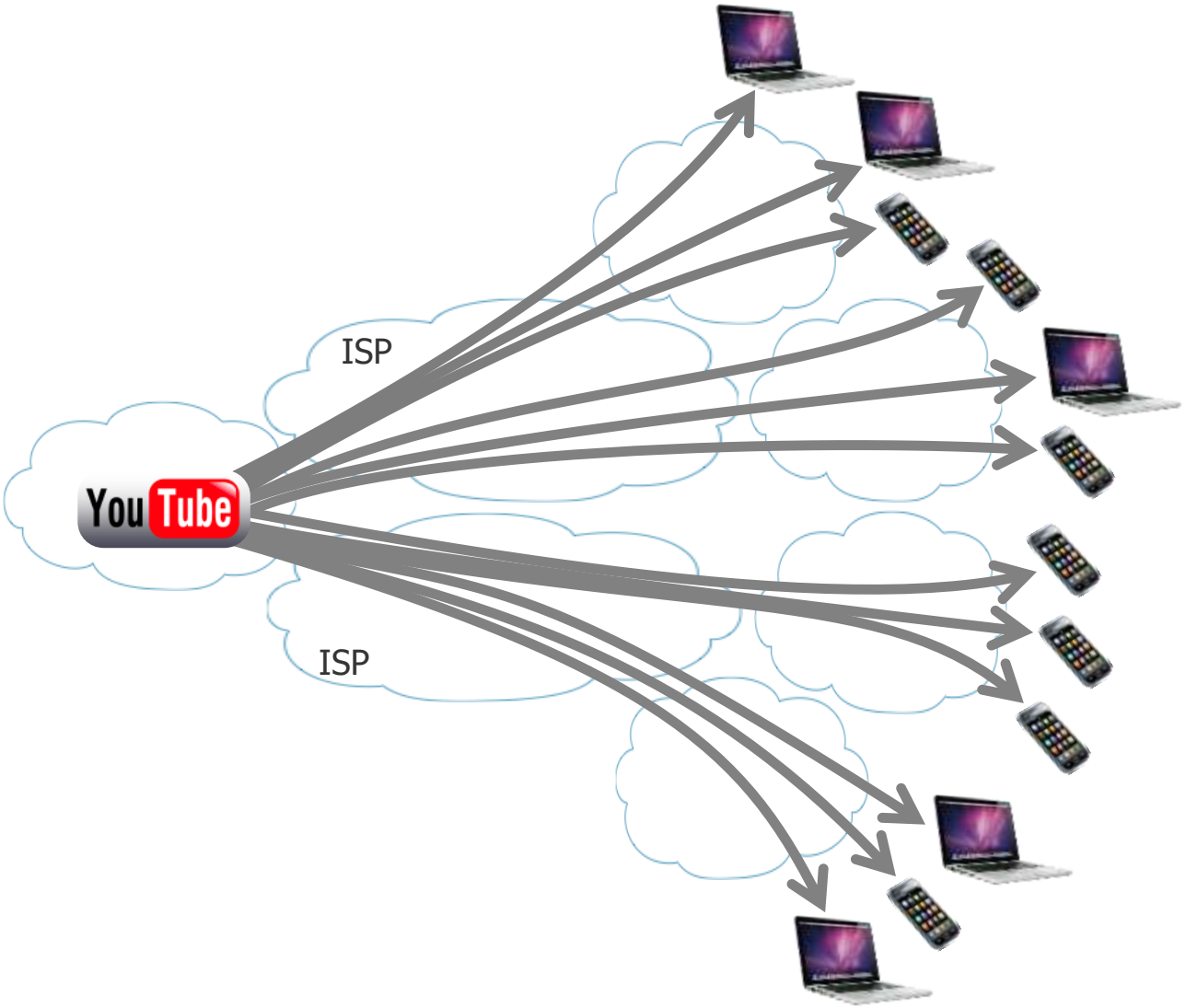
Named Data Networking



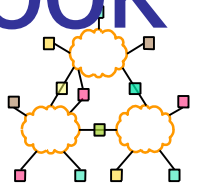
- In the beginning…
  - First applications strictly focused on host-to-host interprocess communication:
    - Remote login, file transfer, …
  - Internet was built around this host-to-host model.
  - Architecture is well-suited for communication between pairs of stationary hosts.
- … while today
  - Vast majority of Internet usage is data retrieval and service access.
  - Users care about the content and are oblivious to location. They are often oblivious as to delivery time:
    - Fetching headlines from CNN, videos from YouTube, TV from Tivo
    - Accessing a bank account at www.bank.com.

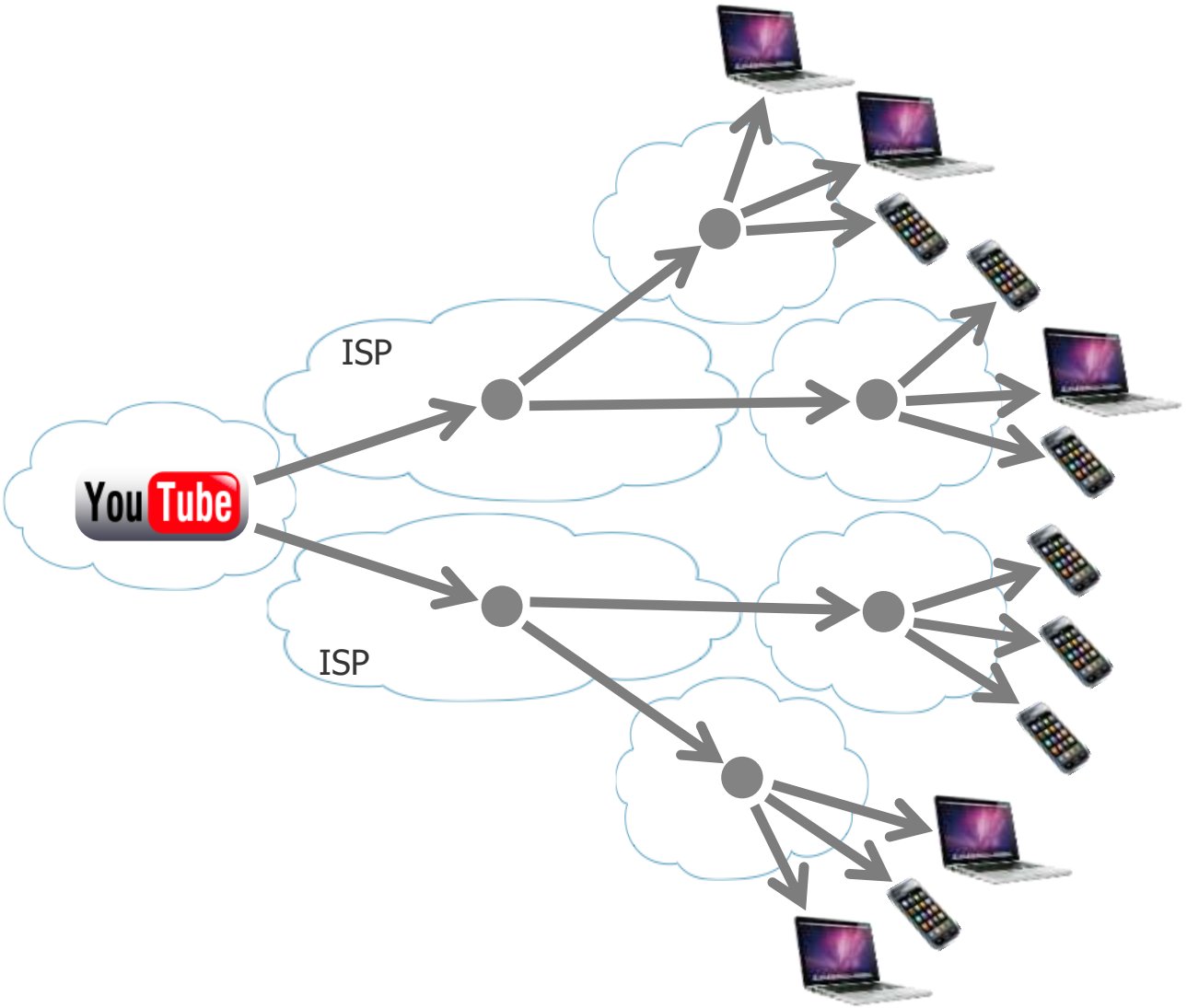
What does the network look like…



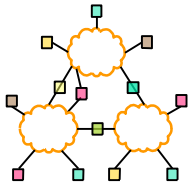


What should the network look like…

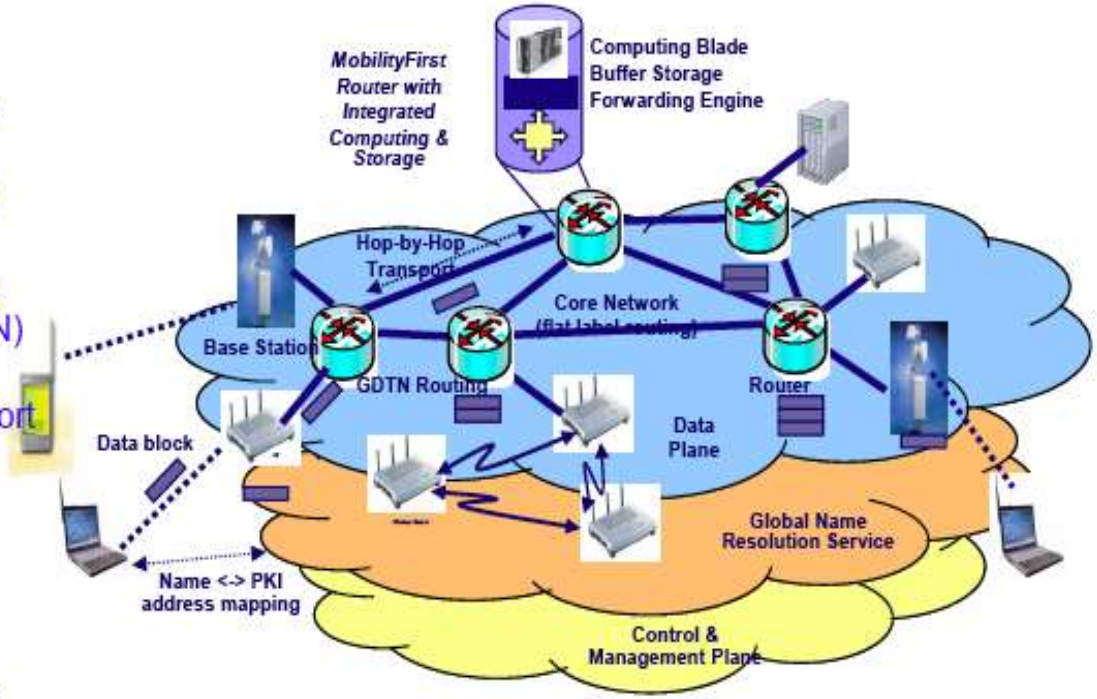




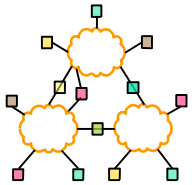
MobilityFirst Architecture

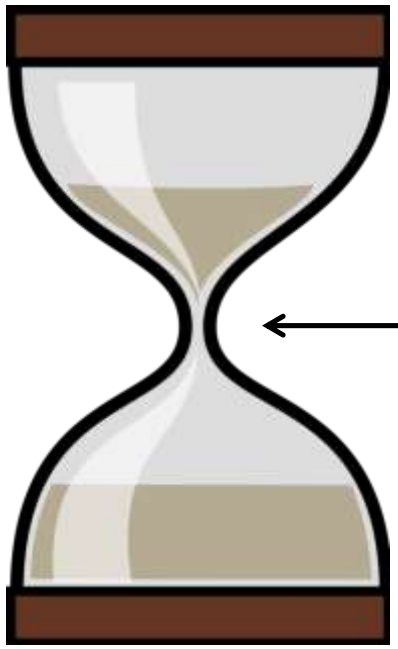


- MobilityFirst key protocol features:
  - Fast global naming service
  - Self-certifying public key names
  - Dynamic mapping of name to topological network address(es)
  - Routers support both flat name and hierarchical address routing
  - Storage-aware (generalized DTN) routing in access
  - Hop-by-hop (segmented) transport
  - Programmable computing layer
  - Support for content/context/location
  - Separate network mgmt plane
- New components, very distinct from IP, intended to achieve key mobility and trust goals



A History of Internet Evolution





Applications

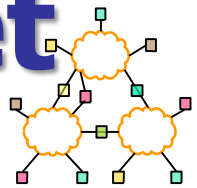
IP

Innovation both above and below IP

Technology

- Hard to change IP
  - …especially after 1990

XIA: An Evolvable Internet Architecture



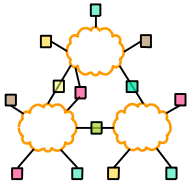
Network design

How do we design a network in which we can gracefully introduce new functionality?

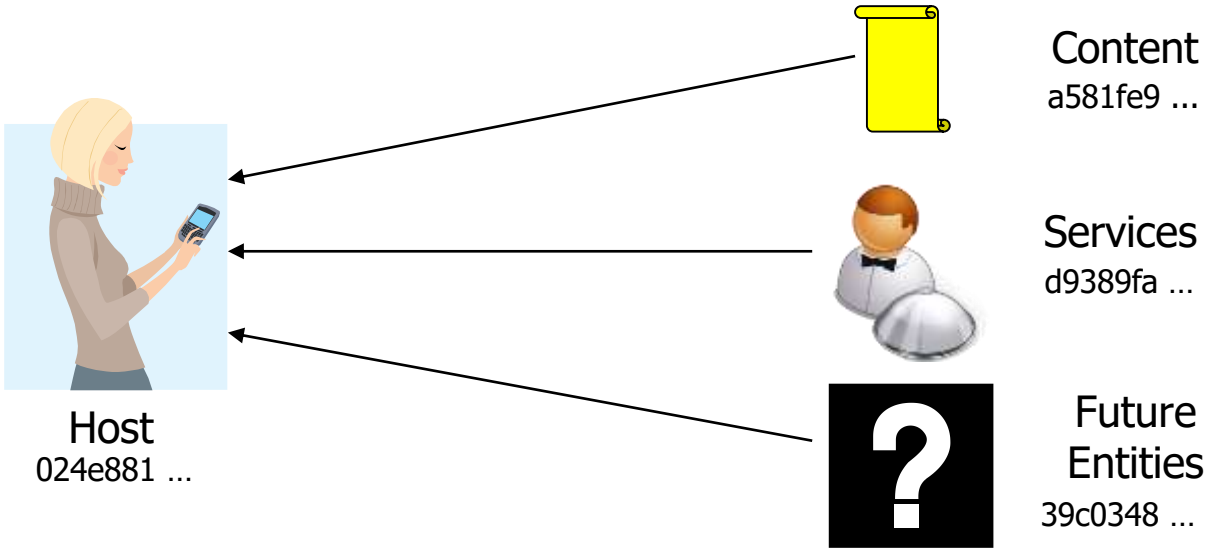
End-point design

How should the end-points change when the networks provide new functionality?

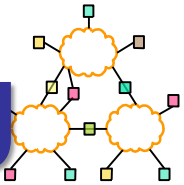
XIA: Evolvable Set of Principals



- Identifying the intended communicating entities reduces complexity and overhead
  - No need to force all communication at a lower level (hosts), as in today's Internet
- Allows the network to *evolve*



Next: Packet forwarding



- IP lookup
  - How router find packets' output ports.
- References
  - **DMP-tree: A dynamic M-way prefix tree data structure for strings matching**, Nasser **Yazdani**, Hossein Mohammadi http://www.sciencedirect.com