Introduction to Communication Networks
The University of Kansas EECS 563
Network Layer

James P.G. Sterbenz
Department of Electrical Engineering & Computer Science
Information Technology & Telecommunications Research Center
The University of Kansas

jgps@eecs.ku.edu

http://www.ittc.ku.edu/~jgps/courses/intronets
Network Layer
Functions and Services

NL.1  Network layer functions and services
NL.2  Network signalling paradigms
NL.3  Switches and packet structure
NL.4  Examples: PSTN and Internet
NL.5  Fast datagram routers
Network Layer
Hybrid Layer/Plane Cube

Layer 3:
- path control in control plane and forwarding in data plane
- interaction with management plane particularly important

Layer 3: network
- path control in control plane and forwarding in data plane
- interaction with management plane particularly important

Layer 2.5: virtual link
Layer 2: link
Layer 1.5: MAC
Layer 1: physical
Layer 5: session
Layer 4: transport
Layer 7: application
Layer 8: social

L1.5, L2, L4, L5, L7, L8

social, application, session, transport, virtual link, link, MAC, physical, network
Network Layer Definitions

- **Forwards packets** along a *path* (or *route*) through the network to a destination *address*: either as individual *datagrams*, as a *flow* of datagrams, or on a *connection* across a *circuit* or *virtual circuit*.
Network Layer

Sublayer History

- Network layer may be subdivided into sublayers
  - ad hoc designations without compete agreement
- History
  - IP originally *internetworking layer* above *subnet network* layer
    - OSI model also contained this [ISO 8648]
  - IP evolved from internetworking (~3.5) layer...
  - ...to global *network layer*
Network Layer

Network Protocols and Devices

• **Network protocol**
  - responsible for determining on *which* link frame transmitted
  - moves packets on path through the network between *nodes*
Network Layer
Service and Interfaces

- Network layer 3 is above link layer 2
  - *addressing*: network layer identifier for end systems (hosts)
  - *forwarding*: transfers packets hop-by-hop
    - using link layer services
    - network layer responsible for determining *which* next hop
  - *routing*: determination of path to forward packets *Lect NR*
  - *signalling*: messages to control network layer behaviour
  - *traffic management*: management of traffic and congestion *Lecture TQ*

- Network layer service to transport layer (L4)
  - deliver TPDUs to destination transport entity
Network Layer
Forwarding vs. Routing

- **Forwarding** transfers packets through switch (data pl.)
  - each switch (router) makes decision on which link to send
  - forwarding table (generally) used to make decision
  - forwarding is *per packet* decision
    [analogy: determining which exits to take on a drive]

- **Routing** determines the path to take (control plane)
  - routing algorithm independent of forwarding
  - forwarding table entries populated by routing
  - routing is (generally) not done per packet
    [analogy: planning trip from source to destination]

*Forwarding and routing are very different*
Network Layer
Service and Interfaces

- Network layer *packet* encapsulates *TPDU*
  - *packet* = *header* + *TPDU* + opt. trailer (protocol dependent)

```
transport layer
  TPDU
  H  TPDU
network layer
  packet
  H  TPDU
link layer
transport layer
  TPDU
  H  TPDU
network layer
  packet
  H  TPDU
link layer
```

10 October 2017  KU EECS 563 – Intro Comm Nets – Network Layer  ICN-NL-9
Network Layer Service

Service Models

- Several orthogonal dimensions
  - best effort vs. QoS
  - reliability
  - ordering
Network Layer Service

Service Models: Best Effort

- Best effort
  - network attempts to:
    - deliver most packets
    - most of the time
    - eventually
  - network may attempt to be fair among users
  - contrast with best-effort applications

*This is the Internet service model*

- Differentiated service
- Guaranteed service
Network Layer Service
Service Models: Unreliable

- Reliable delivery
- Statistical reliability
- Unreliable: packet may or may not reach destination
  - if necessary, reliability provided by higher layer
    - end-to-end transport
    - application-to-application

This is the Internet service model
Network Layer Service
Service Models: Unordered Delivery

- Ordered
- Unordered: packets may be misordered
  - transport layer will reorder
  - application will reorder
  - application doesn’t care about order

This is the Internet service model
Network Architecture

Characteristics

• Network establishes paths between end systems
  – all applications (that need to) must be able to communicate

• Heterogeneity of
  – underlying links and LAN technologies
  – overlying applications
  – service providers

• Requires common addressing mechanism
  – and compatible routing and signalling
Network Architecture

Hourglass Principle

- Internet is “waist of the hourglass”
  - common addressing and forwarding (IP)
  - compatible routing (BGP) and signalling (ICMP)

- The network layer is the hardest to replace or evolve
  - even to new versions (e.g. IPv4 → IPv6)

- IP won over alternatives
  - e.g. X.25 CONS, CLNP, SNA, XNS, DECNET, ATM, ...
  - but ideas in these architectures still important
Network Layer

Network Signalling Paradigms

NL.1  Network layer functions and services
NL.2  Network signalling paradigms
NL.3  Switches and packet structure
NL.4  Examples: PSTN and Internet
NL.5  Fast datagram routers
Network Layer Service

Service Models: Signalling Paradigms

- Circuit network service
  - physical path
- Connection-oriented network service
  - virtual circuit
- Connectionless network service
  - packets

_This is the Internet signalling paradigm_
Network Circuits
State Management

- **Circuits**
  - physical path established
  - circuit state to establish and maintains path

*Examples?*
Network Circuits
State Management

• Circuits
  – physical path established
  – circuit state to establish and maintains path

• Examples
  – early PSTN
  – X.21 circuit switched networks
  – optical WDM lightpaths  
    Lecture LL
Network Circuits
Signalling and Data Transfer

- Circuit signalling characteristics
  - setup latency: RTT before data transfer
  - no multiplexing efficiency
  + negligible switch latency
  - resources must be released

Diagram showing network circuits with nodes S, 1, 2, 3, 4, and R.
Network Connections

Motivation

- Combine benefits of datagram and circuits
  - statistical multiplexing gains of datagrams
  - forwarding performance circuits
    - eliminate store-and-forward
    - high-performance switch design
  - provision of QOS
    - admission control
    - resource reservation per connection

Lecture TQ
Network Connections

State Management

- Connection-oriented
  - connection state \textit{required}
  - performance optimisations possible to reduce setup latency
    - fast reservations
    - optimistic connection establishment

\textit{Examples?}
Network Connections
State Management

• Connection-oriented or \textit{virtual circuit}
  – connection state \textit{required}
  – performance optimisations possible to reduce setup latency
    • fast reservations
    • optimistic connection establishment

• Examples
  – CONS (connection-oriented network service)
  – PSPDNs (packet-switched public data networks)
  \[\text{ISO/IEC 8878+8208 / ITU X.25}\]
  – ATM and MPLS
  – modern PSTN (wired and wireless)
Network Connections
Virtual-Circuit Signalling

- Connections (virtual circuits)
  - establish state once to reduce per packet processing
  + amortised for long flows
  - expensive for transactions
  - RTT delay before data transfer
  + high throughput possible
  + per hop messages reduce latency
Network Connections
Virtual-Circuit Forwarding

• Each packet contains a connection identifier
• Each switch does a lookup in a connection table
  – outgoing port = lookup (connection id)
• Each switch hop does a label swap
  – new label = lookup (connection id)
  – prevents the need for global connection id allocation
• Table lookup very efficient
  – connection id is index into simple table
  – can be done in fast hardware at line rate
Connectionless Network
State Management

• Connectionless
  – no per flow state \textit{required} to forward information
  – but there still is state
    \textit{what?}
Connectionless Network
State Management

• Connectionless
  – no per flow state required to forward information
  – but there still is state
    • forwarding tables
  – other state may be used to improve performance
    • per-flow queueing  Lecture TQ
    • soft state flow identification to improve performance

Examples?
Connectionless Network
State Management

• Connectionless
  – no per flow state *required* to forward information
  – but there still is state
    • forwarding tables
  – other state may be used to improve performance
    • per-flow queueing *Lecture TQ*
    • soft state flow identification to improve performance

• Examples
  – Internet IP
  – CLNP (connectionless layer network protocol)
    [ISO/IEC 8473 / ITU X.223]
Connectionless Network
Datagram Forwarding

- Each datagram contains a *destination address*
- Each hop does a *lookup* in a *forwarding table*
  - outgoing port = lookup (destination address)
- Table lookup efficiency depends on:
  - address structure (e.g. class-based IP vs. CIDR)
  - address length
  - tables length (# destinations per switch/router)
- Example: IP
  - note: IP lookup could not be done at line rate in 1980s

*more later*
Connectionless Network

Datagram Forwarding

- Connectionless signalling
  + no setup latency
  - store-and-foreword lookup delay
  + multiplexing efficiency
- large messages broken into *packets*
  + other flows interleave
- $d_r > d_t$
Network-Layer Service
Comparison of Signalling Paradigms

- Connectionless vs. connection-oriented networks
  - major debate in 1980s and 1990s
  - connectionless IP-based Internet won

- Comparison of characteristics

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Connectionless</th>
<th>Connection-oriented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup latency</td>
<td>↓ none</td>
<td>↑ round trip</td>
</tr>
<tr>
<td>Forwarding latency</td>
<td>↑ address lookup</td>
<td>↓ VC index</td>
</tr>
<tr>
<td>Forwarding information</td>
<td>↑ address per packet</td>
<td>↓ VC id per packet</td>
</tr>
<tr>
<td>Switch state</td>
<td>↑ forwarding tables</td>
<td>↓ connection id tables</td>
</tr>
<tr>
<td>Resilience to failure</td>
<td>datagrams lost</td>
<td>connections terminated</td>
</tr>
<tr>
<td>QoS</td>
<td>difficult</td>
<td>connection reservation</td>
</tr>
</tbody>
</table>
Network Layer

Switches and Packet Structure

NL.1 Network layer functions and services
NL.2 Network signalling paradigms
NL.3 Switches and packet structure
NL.4 Examples: PSTN and Internet
NL.6 Fast datagram routers
Switches
Overview

• **Switch**
  – intermediate system
  – switches packets from ingress to egress port
  – in Internet has become synonymous with non-IP switch

• **Router**
  – switch that operates on IP datagrams

• **We will use the term generically for now**
  – any network type (Internet, PSTN)
  – any technology
Switches
Representation

- Switch interconnecting $n$ inputs $i_k$ to $n$ outputs $o_k$
  - bidirectional links $i_k o_k$
  - split to simplify diagrams
Switches
Functions: Overview

- **Switch**: any switching device
  - L2, L3 (including IP)
- **Routing / signalling**: per flow or longer
- **Transfer control**: per packet control
- **Data manipulation**: per byte or packet
Router Architecture

Example: Cisco IOS

- IOS (Internetwork Operating System)
  - OS for Cisco routers (IP switches)
  - proprietary embedded OS
Router Architecture

Example: Cisco IOS Architecture

- **Processes**
  - routing protocols
  - software forwarding

- **Kernel**
  - memory management
  - process scheduling

- **Packet buffers**

- **Device drivers**
  - network interface drivers

- **Fast switching software**
  - optimised packet switching
Fast Packet Switching

Motivation

- Allow network switching at line rate
  - 155 Mb/s (OC-3) in mid 1980s
- Eliminate store-and-forward processing bottlenecks
- Eliminate blocking in switch
- Provide support for QOS
- Solution:
  - virtual connection service
Fast Packet Switch Architecture

- **Connection state**
  - simple per packet processing

- **Switch fabric**
  - eliminate contention
  - no store-and-forward
Fast Packet Switch
Connectionless vs. Connection Tradeoff

- Connection-oriented fast packet switching
  - requires round trip connection setup latency
  - achieved higher data rate due to simple label swap
    - IP lookup was a bottleneck in 1980s
- Lost out to connectionless IP datagrams
  - but became core of modern high-performance routers
Packet Size and Structure

Variability

• Fixed size (cells)
  + easier to design switches
  − difficult to predetermine the best size

• Variable size (typical in Internet)
  − more difficult to design switches
  + no need for agreement on size
  + less need for fragmentation/segmentation

• Discrete sizes: advantages of both fixed and variable
  • integral multiples, e.g. 64B, 128B, 192B...
  • power-of-2 scaling with data rate
    e.g. 128B @ OC-3, 256B @ OC-12, 512B @ OC-48
Packet Size and Structure

Size

- **Small packets**
  - efficient statistical multiplexing
  - high header/payload overhead
  - short interarrival time challenge per packet processing
    - note: this is one major reason ATM failed

- **Large packets**
  - significantly easier per packet processing
  - less efficient statistical multiplexing
  - larger queueing delays
  - efficient transport of large data blocks
  - inefficient transport of signalling and control messages
    - e.g. TCP ACKs
Packet Size and Structure

Packet Format

- **Header**
  - fields that *determine* packet processing

- **Payload**
  - TPDU transport protocol data unit

- **Trailer**
  - fields that are *dependent* on packet processing
    - e.g. checksum to allow cut-through

*version* | *type*
--- | ---
address / connection id | QoS
authentication | payload length
payload length | check
check | TPDU
Packet Size and Structure

Packet Format: Header

- **Header**
  - fields that determine packet processing
    - protocol version
    - packet type: control, data, etc.
    - connection id or protocol demux
    - QoS and authentication if needed
    - header check
      - processing even if data corrupted

- **Payload**

- **Trailer**

<table>
<thead>
<tr>
<th>version</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>address / conn. id</td>
<td>QoS</td>
</tr>
<tr>
<td>authentication</td>
<td>payload length</td>
</tr>
<tr>
<td>check</td>
<td>TPDU</td>
</tr>
</tbody>
</table>
Packet Size and Structure

Packet Format: Payload

- Header
- **Payload**
  - TPDU transport protocol data unit
- Trailer

<table>
<thead>
<tr>
<th>version</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>connection id</td>
<td></td>
</tr>
<tr>
<td>QoS</td>
<td></td>
</tr>
<tr>
<td>authentication</td>
<td></td>
</tr>
<tr>
<td>payload length</td>
<td></td>
</tr>
<tr>
<td>check</td>
<td></td>
</tr>
</tbody>
</table>

TPDU
Packet Size and Structure

Packet Format: Trailer

- Header
- Payload
- **Trailer**
  - fields that are *dependent* on packet processing
  - checksum to allow cut-through
    - compute and compare

<table>
<thead>
<tr>
<th>version</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>connection id</td>
<td></td>
</tr>
<tr>
<td>QoS</td>
<td></td>
</tr>
<tr>
<td>authentication</td>
<td></td>
</tr>
<tr>
<td>payload length</td>
<td></td>
</tr>
<tr>
<td>check</td>
<td></td>
</tr>
</tbody>
</table>

TPDU

Check
Packet Size and Structure

Control Fields

- Control field structure and encoding is critical
  - simple encoding (bit vectors vs. code points)
  - byte/octet granularity and alignment
  - field length
    - fixed when possible
    - variable length prepended with length (skip vs. hunt)
Switches
Functions: Switch Fabric

- **Switch**: any switching device L2, L3 (including IP)
- **Routing / signalling**
  - per flow or longer
- **Transfer control**
  - per packet control
- **Data manipulation**
  - per byte or packet
Switch Fabric Architecture

Blocking

- Blocking (among \textit{different} outputs)
- Goal: \textit{nonblocking} switch fabric
  - input–output path $i_j \rightarrow o_m$ will not block a different path $i_k \rightarrow o_n$
- Some switch designs are mostly nonblocking
  - strictly nonblocking: under all conditions
  - wide-sense nonblocking: if particular algorithm is used
  - rearrangeably nonblocking: if existing paths are rearranged
  - virtually nonblocking: with extremely low probability
Switch Fabric Architecture

Contestation and Buffering

- Contention (burst collisions) in a non-blocking fabric
  - occurs when traffic destined for *same* output
  - requires buffering even for well-behaved traffic
Switch Fabric Architecture

Single Stage: Basic $2 \times 2$ Switch Element

- **States**
  - point-to-point
    - straight
    - cross
  - multicast

- **Types**
  - buffered or unbuffered
  - self routing or externally controlled
Switch Fabric Architecture

Single Stage: Crossbar Switch

- Crosspoint switch element
  - electronic
    - multicast possible
  - optical MEMS
    - rotating mirror
Switch Fabric Architecture
Single Stage: Crossbar Switch

- **Crossbar** fabric
  - square array of crosspoint elements
  - $O(n^2)$ growth complexity
  - reasonable for moderate $n$
Crossbar Switch
Path Selection

- Crossbar fabric
  - simple path routing
    - element \((o,i)\) turns

```
i0
  o0
  o1
  o2
  o3
  o4
  o5
  o6
  o7
i1
i2
i3
i4
i5
i6
i7
```

© James P.G. Sterbenz
Crossbar Switch
Path Selection

- Crossbar fabric
  - simple path routing
  - element \((o,i)\) turns
  - \(i_3 \rightarrow o_4\)

![Crossbar Switch Diagram]
Crossbar Switch

Path Selection

• Crossbar fabric
  – simple path routing
    • element \((o, i)\) turns
    • \(i_3 \rightarrow o_4\)
Crossbar Switch

Path Selection

- Crossbar fabric
  - simple path routing
    - element \((o,i)\) turns
    - \(i_3 \rightarrow o_4\)
Crossbar Switch
Strictly Nonblocking

- Crossbar fabric
  - simple path routing
    - element \((o,i)\) turns
    - \(i_3 \rightarrow o_4\)
  - strictly nonblocking
    - \(i_j \rightarrow o_n\) noblock \(i_k \rightarrow o_m\)
    - \(\forall j,k,n,m: i \neq j, n \neq m\)
    - \(i_1 \rightarrow o_1\)
Crossbar Switch
Strictly Nonblocking

- Crossbar fabric
  - simple path routing
    - element \((o, i)\) turns
    - \(i_3 \rightarrow o_4\)
  - strictly nonblocking
    - \(i_j \rightarrow o_n\) noblock \(i_k \rightarrow o_m\)
    - \(\forall j, k, n, m: i \neq j, n \neq m\)
    - \(i_1 \rightarrow o_1\)
Crossbar Switch
Strictly Nonblocking

- Crossbar fabric
  - simple path routing
    - element \((o,i)\) turns
    - \(i_3 \rightarrow o_4\)
  - strictly nonblocking
    - \(i_j \rightarrow o_n\) noblock \(i_k \rightarrow o_m\)
    - \(\forall j,k,n,m: i \neq j, n \neq m\)
    - \(i_1 \rightarrow o_1\)
Switch Fabrics
Multistage Switches

- Large switches built from single stage elements
  - $2 \times 2$ elements or $n \times n$ crossbars
  - $O(n \log n)$ growth complexity
Multistage Switch Fabrics
Delta Fabric Construction and Scalability

- Delta fabric
  - $O(n \log n)$
    - $n/2$ rows
    - $\log_2 n$ stages
    - $n = 2$
    - $2/2 \log_2 2 = 1$
Multistage Switch Fabrics
Delta Fabric Construction and Scalability

- Delta fabric
  - $O(n \log n)$
    - $n/2$ rows
    - $\log_2 n$ stages
    - $n = 4$
    - $4/2 \log_2 4 = 4$
Multistage Switch Fabrics
Delta Fabric Construction and Scalability

- Delta fabric
  - $O(n \log n)$
  - $n/2$ rows
  - $\log_2 n$ stages
  - $n = 8$
  - $8/2 \log_2 8 = 12$
Delta fabric
- $O(n \log n)$
  - $n/2$ rows
  - $\log_2 n$ stages
  - $n = 16$
  - $16/2 \log_2 16 = 32$
Multistage Switch Fabrics
Delta Fabric Construction Self-Routing

- Delta fabric
  - self-routing
  - $i^{th}$ bit of $p_{out}$ used to make routing decision in $i^{th}$ stage
Multistage Switch Fabrics
Delta Fabric Construction Self-Routing

- Delta fabric
  - self-routing
  - $i^{th}$ bit of $p_{out}$ used to make routing decision in $i^{th}$ stage
  - $i_2 \rightarrow o_{10}$
Multistage Switch Fabrics

Delta Fabric Construction Self-Routing

- Delta fabric
  - self-routing
  - $i^{th}$ bit of $p_{out}$ used to make routing decision in $i^{th}$ stage
  - $i_2 \rightarrow o_{10}$
Multistage Switch Fabrics
Delta Fabric Construction Self-Routing

- Delta fabric
  - self-routing
  - $i^{th}$ bit of $p_{out}$ used to make routing decision in $i^{th}$ stage
  - $i_2 \rightarrow o_{10}$
Multistage Switch Fabrics

Delta Fabric Construction Self-Routing

- Delta fabric
  - self-routing
  - $i^{th}$ bit of $p_{out}$ used to make routing decision in $i^{th}$ stage
  - $i_2 \rightarrow o_{10}$
Multistage Switch Fabrics
Delta Fabric Construction Self-Routing

- Delta fabric
  - self-routing
  - $i^{th}$ bit of $p_{out}$ used to make routing decision in $i^{th}$ stage
  - $i_2 \rightarrow o_{10}$
Multi-stage Switch Fabrics

Delta Fabric Construction Self-Routing

- Delta fabric
  - self-routing
  - $i^{th}$ bit of $p_{out}$ used to make routing decision in $i^{th}$ stage
  - $i_2 \rightarrow o_{10}$
Multistage Switch Fabrics
Delta Fabric Construction Self-Routing

- Delta fabric
  - self-routing
  - $i^{th}$ bit of $p_{out}$ used to make routing decision in $i^{th}$ stage
  - $i_2 \rightarrow o_{10}$
Multistage Switch Fabrics
Delta Fabric Construction Self-Routing

- Delta fabric
  - self-routing
  - $i^{th}$ bit of $p_{out}$ used to make routing decision in $i^{th}$ stage
  - $i_2 \rightarrow o_{10}$
Multistage Switch Fabrics
Delta Fabric Construction Self-Routing

- Delta fabric
  - self-routing
  - \(i^{th}\) bit of \(p_{out}\) used to make routing decision in \(i^{th}\) stage
  - \(i_2 \rightarrow o_{10}\)
Multistage Switch Fabrics
Delta Fabric Construction Self-Routing

- Delta fabric
  - self-routing
  - $i^{th}$ bit of $p_{out}$ used to make routing decision in $i^{th}$ stage
  - $i_2 \rightarrow o_{10}$
Multistage Switch Fabrics
Delta Fabric Construction Self-Routing

- Delta fabric
  - self-routing
  - $i^{th}$ bit of $p_{out}$ used to make routing decision in $i^{th}$ stage
    - $i_2 \rightarrow o_{10}$
    - $i_{13} \rightarrow o_{10}$
Multistage Switch Fabrics
Delta Fabric Construction Self-Routing

- Delta fabric
  - self-routing
  - $i^{th}$ bit of $p_{out}$ used to make routing decision in $i^{th}$ stage
    - $i_2 \rightarrow o_{10}$
    - $i_{13} \rightarrow o_{10}$
Multistage Switch Fabrics
Delta Fabric Construction Self-Routing

- Delta fabric
  - self-routing
  - $i^{th}$ bit of $p_{out}$ used to make routing decision in $i^{th}$ stage
    - $i_2 \rightarrow o_{10}$
    - $i_{13} \rightarrow o_{10}$
Multistage Switch Fabrics

Delta Fabric Construction Self-Routing

- Delta fabric
  - self-routing
  - $i^{th}$ bit of $p_{out}$ used to make routing decision in $i^{th}$ stage
    - $i_2 \rightarrow o_{10}$
    - $i_{13} \rightarrow o_{10}$
• Delta fabric
  – self-routing
  – $i^{th}$ bit of $p_{out}$ used to make routing decision in $i^{th}$ stage
  – $i_2 \rightarrow o_{10}$
  – $i_{13} \rightarrow o_{10}$
Multistage Switch Fabrics

Delta Fabric Construction Self-Routing

- Delta fabric
  - self-routing
  - $i^{th}$ bit of $p_{out}$ used to make routing decision in $i^{th}$ stage
    - $i_2 \rightarrow o_{10}$
    - $i_{13} \rightarrow o_{10}$
Multistage Switch Fabrics
Delta Fabric Construction Self-Routing

- Delta fabric
  - self-routing
  - $i^{th}$ bit of $p_{out}$ used to make routing decision in $i^{th}$ stage
    - $i_2 \rightarrow o_{10}$
    - $i_{13} \rightarrow o_{10}$
Multistage Switch Fabrics
Delta Fabric Construction Self-Routing

- Delta fabric
  - self-routing
  - $i^{th}$ bit of $p_{out}$ used to make routing decision in $i^{th}$ stage
    - $i_2 \rightarrow o_{10}$
    - $i_{13} \rightarrow o_{10}$
• Delta fabric
  – self-routing
  – $i^{th}$ bit of $p_{out}$ used to make routing decision in $i^{th}$ stage
    – $i_2 \rightarrow o_{10}$
    – $i_{13} \rightarrow o_{10}$
Network Layer

PSTN

NL.1  Network layer functions and services
NL.2  Network signalling paradigms
NL.3  Switches and packet structure
NL.4  Examples: PSTN and Internet
   NL.4.1  PSTN
   NL.4.2  IP, ICMP, and IPv6
NL.5  Fast datagram routers
PSTN
Functions and Protocols

- Addressing
- Forwarding
- Routing
- Signalling
- Traffic Management
PSTN
Functions and Protocols: Addressing

- **Addressing**: telephone number [ITU E.164]
- Forwarding
- Routing
- Signalling
- Traffic Management
PSTN
Functions and Protocols: Forwarding

- **Addressing**: E.164
- **Forwarding**:
  - traditional circuit switch: physical connection
  - modern virtual circuit
    - TDM mux/demux
    - ATM label-swap switching
  - emerging: VoIP using IP forwarding
- **Routing**
- **Signalling**
- **Traffic Management**
PSTN
Functions and Protocols: Routing

- **Addressing**: E.164
- **Forwarding**: circuit or virtual circuit switch
- **Routing**: *Lecture NR*
  - traditional: HIER
    - static hierarchy based on telephone number
  - network engineering to provide required service
    - blocking probability
  - modern: dynamic routing (DNHR, RTNR)
- **Signalling**
- **Traffic Management**
PSTN
Functions and Protocols: Signalling

- Addressing: E.164
- Forwarding: circuit or virtual circuit switch
- Routing: HIER, DNHR, RTNR, ...

**Signalling:**
- traditional: in-band audio
- modern: out-of-band (common channel signalling) SS7

- Traffic Management
PSTN
Functions and Protocols: Traffic Management

- **Addressing**: E.164
- **Forwarding**: circuit or virtual circuit switch
- **Routing**: HIER, DNHR, RTNR, ...
- **Signalling**: SS7
- **Traffic management**:
  - network engineering to provide required service
    - blocking probability
  - modern: ATM and MPLS traffic engineering
PSTN Addressing

History

• Each telco devised its own numbering plan
  – US small towns frequently had 4- or 5-digits

• Bell System standardised on 7 digits
  – 3 digit exchange represented as exchange name + number
    • intended to make numbers easier to remember
    • std. names: http://www.ourwebhome.com/TENP/Recommended.html
  – 4 digit subscriber line id
  – example evolution in Lawrence
    • UNiversity 4-7890
    • UN 4-7890
    • 864-7890
PSTN Addressing

Addressing Notation

- **Notation** [ITU E.123] symbol & icons [ITU E.121]
  - country-specific international access denoted by +
  - followed by grouped digits (no hyphens, dots)
    - grouping based on each country's numbering plan, e.g.
      - +1 785 864 7890
      - +1 508 944 3067
      - +44 1524 510302
      - +41 44 632 70 01
PSTN Addressing
Address Format

- Telephone number format ($\leq 15$ digits) [ITU E.164]
  \[(\text{country-code}) (\text{national-destination-code}) (\text{subscriber-number})\]
  - country code (1–3 digits) assigned by ITU
    * List of ITU-T Recommendation E.164 Assigned Country Codes
  - NDC: national destination code (city code or area code)
    * may be fixed length (e.g. US) or variable length (e.g. UK)
    * may be structured (e.g.) Germany or unstructured (e.g. US)
  - SN: subscriber number
    * may be structured as in US
      \[(\text{subscriber number}) = (\text{central-office-exch.}) (\text{subscriber-line-id})\]
PSTN: Addressing
Global Addressing

- Bell System IDDD
  - international direct distance dialing
- ITU Country codes
  - early numbering in 1960 ITU Red Book
  - current scheme defined 1963 ITU Blue Book
- Grouped into 10 zones
- 1 – 3 digits
  - variable length code

<table>
<thead>
<tr>
<th>zone</th>
<th>area</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>US, Canada, Caribbean</td>
</tr>
<tr>
<td>2</td>
<td>Africa</td>
</tr>
<tr>
<td>3</td>
<td>Europe</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Mexico, Central and S. America</td>
</tr>
<tr>
<td>6</td>
<td>South Pacific</td>
</tr>
<tr>
<td>7</td>
<td>Russia (former Soviet Union)</td>
</tr>
<tr>
<td>8</td>
<td>East Asia and special services</td>
</tr>
<tr>
<td>9</td>
<td>West and South Asia</td>
</tr>
<tr>
<td>0</td>
<td>spare</td>
</tr>
</tbody>
</table>
# PSTN: Addressing

## Zone 1 Country Codes (NANP)

<table>
<thead>
<tr>
<th>Code</th>
<th>TLD</th>
<th>Country</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1NXX</td>
<td>.us</td>
<td>United States</td>
<td>United States</td>
</tr>
<tr>
<td>+1NXX</td>
<td>.ca</td>
<td>Canada</td>
<td>Canada</td>
</tr>
<tr>
<td>+1706</td>
<td>.mx</td>
<td>Mexico</td>
<td>Mexican (now +52 part of Mexico) was accessible in NANP before 1991</td>
</tr>
<tr>
<td>+1905</td>
<td></td>
<td>Mexico City</td>
<td></td>
</tr>
<tr>
<td>+1NXX</td>
<td></td>
<td>(Caribbean Nations)</td>
<td></td>
</tr>
</tbody>
</table>
PSTN Addressing
NANP (WZ1) Origins

- Long distance originally required operator assistance
- NANP (North American numbering plan) in 1947
  - DDD (direct distance dialing) began in 1951
PSTN Addressing
NANP (WZ1) Administration

- NANPA (NANP Administrator)  www.nanpa.org
- CNA (Canadian Number Administrator)  www.cnac.ca
- Regulation by FCC in US
  - NANC (North American Numbering Council)
    www.fcc.gov/wcb/tapd/Nanc
- Guidance from
  - ATIS INC (Industry Numbering Committee)
    www.atis.org/inc/docs.asp
  - CISC CSCN (Canadian Steering Committee on Numbering)
    www.crtc.gc.ca/cisc/eng/cisf3f.htm
PSTN Addressing
NANP (WZ1) Traditional Hierarchical Structure

+1
final trunk group
(to class 4 long distance switching)
913

local loops
local (class 5) switches
direct trunk group
tandem trunk group
tandem office
local tandem switches
direct trunk group
local office 897
tandem trunk group
local loops
local (class 5) switches
direct trunk group
local office 897
8538
PSTN Addressing
NANP (WZ1) Nomenclature and Format

• Nomenclature: $N = \{2\ldots9\}; \quad X = \{0\ldots9\}; \quad 0/1 = \{0|1\}$
  – entire state codes of form $N0X$
  – split state codes of form $N1X$
  – high-population codes generally low $N/X$
    • reduce pulse delay
    • reduce dial pullback

• NANP is ITU E.164 compliant
  $\langle \text{national-destination-code}\rangle = \text{NPA (area code)}$
  $\langle \text{subscriber number}\rangle = \langle \text{central-office-exch.}\rangle \langle \text{sub.-line-id}\rangle$
  – e.g. +1 913 897 8538
    US KC OP SLID(KU)
PSTN: Addressing
NANP (WZ1) 1947: Original Bell System

- US and Canada
  - N0X dedicated
  - N1X split state
- 86 NPAs assigned
  - 152 N0/1X poss.
  - N00 / N11 reserv.
PSTN: Addressing?
NANP Address Capacity

- **Address space fields**
  - each CO code has 10,000 SLIDs (subscriber line ID)
  - each NPA can have 640 NNX CO codes
  - 152 N°/1 X NPAs

- **Total address space**
  - 152 x 640 x 10000 = 972,800,000 ≈ 10^{10}
  - order of magnitude less than 10^{11} possible with 10 digits
  - NPA and CO geography determines distribution
  - usable number *much* smaller

*How to accommodate growth?*
PSTN: Addressing
NANP Address Capacity

• Options to accommodate growth
  – add SLIDs to approach 10,000 per CO
    • some spares needed for churn
    • reduce redirect period
  – add CO codes until 640 per NPA
    • adding SLIDs and CO codes is relatively easy
    • add and expand CO switches and trunks add NPAs to 152 maximum
    • more difficult

  why?
PSTN: Addressing
NANP NPA Growth and Capacity

- NPA capacity process:
  - NANP forecasts need for new NPAs
  - balancing act
    - too aggressive causes unnecessary number changes
    - too conservative prevents new number assignments
  - jeopardy: demand for new numbers exceed forecasts
    - new CO code assignments are restricted until relief
PSTN: Addressing
NANP NPA Growth and Capacity

• Options for relief
  – NPA split
  – NPA repartition
  – NPA overlay

• Geographic partition required for hierarchical routing
  
  Lecture NR
  
  – requires user numbers to change: disruptive and expensive
Many NPAs near capacity in 1970s
  - $N^0/_{1}X$ NPAs nearly exhausted

Solution?
PSTN: Addressing
NANP NPA Growth and Capacity

- Many NPAs near capacity in 1970s
  - \(N^0/1\times\) NPAs nearly exhausted
- Option
  - underused NPA realignment would cause *massive* disruption
PSTN: Addressing
NANP NPA Growth and Capacity

1st step solution
  - adjust NANP addressing within current framework
  - institute *interchangeable* NPA/CO codes
  - CO codes NNX → NXX
    - address space increases from 640 → 792
    - $152 \times 792 \times 10000 = 1203840000 \approx 10^{10}$ total numbers
  - implication: 10-digit local numbers
    - NPAs no longer distinguishable from CO codes
### PSTN: Addressing NANP (WZ1)

- **Nomenclature:** \( N = \{2 \ldots 9\}; \quad X = \{0 \ldots 9\}; \quad Y = \{0 \ldots 8\}; \quad \theta/1 = \{0|1\} \\

<table>
<thead>
<tr>
<th></th>
<th>1947</th>
<th>1974</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CC</strong></td>
<td></td>
<td><strong>N</strong></td>
</tr>
<tr>
<td><strong>NPA</strong></td>
<td>(N^\theta/1X) [152]</td>
<td>(N^\theta/1X) [152]</td>
</tr>
<tr>
<td><strong>Central office</strong></td>
<td><strong>NNX</strong> [640]</td>
<td><strong>NXX</strong> [792]</td>
</tr>
<tr>
<td><strong>SLID</strong></td>
<td><strong>XXXX</strong> [10000]</td>
<td><strong>XXXX</strong> [10000]</td>
</tr>
<tr>
<td><strong>free</strong></td>
<td></td>
<td><strong>800</strong> [1]</td>
</tr>
<tr>
<td><strong>IDDD + Service code</strong></td>
<td><strong>N11</strong> [8]</td>
<td><strong>N11</strong></td>
</tr>
<tr>
<td><strong>Toll center System code</strong></td>
<td>(\theta XX) [200]</td>
<td>(\theta XX) [200]</td>
</tr>
</tbody>
</table>
PSTN: Addressing NANP NPA Growth and Capacity

• 1st step solution
  – adjust NANP addressing within current framework
  – institute *interchangeable* NPA/CO codes
  – CO codes NNX → NXX
  – implication: 10-digit local numbers
  – still insufficient to meet demand explosion in late 1980s
    • fax machines
    • 2nd residential lines for home computer modems
    • mobile telephones

*solution?*
PSTN: Addressing
NANP NPA Growth and Capacity

• 2nd step solution
  – NPA $N^0/1X \rightarrow NYX$: NPA space increased from 152 $\rightarrow$ $\sim$900
    • $Y=\{0...8\}$
    • usually NYX is still written NXX with a footnote
  – address space now $\approx 712 \times 792 \times 10000 = 6272640000$
    • note: some NPAs and CO code reserved for special use

• Massive growth in 1980s and 1990s
  – $\sim$500 new NYX NPAs
  – but frequent NPA splits very disruptive for users and carriers
PSTN: Addressing NANP NPA Growth and Capacity

- 3rd step solution
  - remove restriction that NPAs be geographically unique
  - enabled by more flexible switch routing software
    - nonhierarchical routing \textit{Lecture NR}
  - \textit{overlay} NPAs: multiple NPAs shared in a given area
  - initially resisted by some PUCs (public utility commissions)
### PSTN: Addressing

**NANP (WZ1)**

- **Nomenclature:** \( N = \{2...9\}; \ X = \{0...9\}; \ Y = \{0...8\} \quad 0/1 = \{0|1\} \)

<table>
<thead>
<tr>
<th></th>
<th>1947</th>
<th>1974</th>
<th>1995</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CC</strong></td>
<td></td>
<td>( N^+ )</td>
<td>( N^+ )</td>
</tr>
<tr>
<td><strong>NPA</strong></td>
<td>( N^0/1X \ [152] )</td>
<td>( N^0/1X \ [152] )</td>
<td>( NYX \ [712] )</td>
</tr>
<tr>
<td><strong>Central office</strong></td>
<td>( NNX \ [640] )</td>
<td>( NXX \ [792] )</td>
<td>( NXX \ [792] )</td>
</tr>
<tr>
<td><strong>SLID</strong></td>
<td>( XXXX \ [10000] )</td>
<td>( XXXX \ [10000] )</td>
<td>( XXXX \ [10000] )</td>
</tr>
<tr>
<td><strong>free</strong></td>
<td></td>
<td>( 800 \ [1] )</td>
<td>( 8xx \ [8] )</td>
</tr>
<tr>
<td><strong>IDDD +</strong></td>
<td></td>
<td>( 011 )</td>
<td>( 011 )</td>
</tr>
<tr>
<td><strong>Service code</strong></td>
<td>( N11 \ [8] )</td>
<td>( N11 )</td>
<td>( N11 )</td>
</tr>
<tr>
<td><strong>Toll center</strong></td>
<td>( 0XX )</td>
<td>( 0XX )</td>
<td>( 0XX )</td>
</tr>
<tr>
<td><strong>System code</strong></td>
<td>( \theta/1XX \ [200] )</td>
<td>( \theta/1XX \ [200] )</td>
<td>( \theta/1XX \ [200] )</td>
</tr>
</tbody>
</table>
PSTN: Addressing
NANP NPA Evolution Example: KS + MO

- 1947: original NPAs [original MO boundary approximate]
  - KS: two N1X codes split between Kansas City and Wichita
  - MO: two N1X codes split between St. Louis and Kansas City
PSTN: Addressing
NANP NPA Evolution Example: KS + MO

- 1950: geographic split
  - MO: Kansas City keeps 816; St. Louis keeps 314
  - MO: southwest replaced with 417 including Springfield
PSTN: Addressing
NANP NPA Evolution Example: KS + MO

- 1995: geographic split
  - MO: St. Louis and suburbs keep 314
  - MO: rest of east replaced by 573
PSTN: Addressing
NANP NPA Evolution Example: KS + MO

- 1996: jeopardy
  - MO: 816 NPA in danger of CO code exhaustion before relief
PSTN: Addressing
NANP NPA Evolution Example: KS + MO

- 1997: geographic split
  - MO: Kansas City suburbs keep 816
  - MO: rest of northwest replaced by 660
PSTN: Addressing
NANP NPA Evolution Example: KS + MO

- **1997: jeopardy**
  - KS: 913 NPA in danger of CO code exhaustion before relief
PSTN: Addressing
NANP NPA Evolution Example: KS + MO

1997: geographic split
- KS: Kansas City suburbs keep 913
- KS: rest of north replaced by 785 including Lawrence
PSTN: Addressing
NANP NPA Evolution Example: KS + MO

- 1998: jeopardy
  - MO: 314 NPA in danger of CO code exhaustion before relief
PSTN: Addressing
NANP NPA Evolution Example: KS + MO

- 1999: geographic split
  - MO: St. Louis city and inner suburbs keeps 314
  - MO: St. Louis suburbs code replaced with 636
PSTN: Addressing
NANP NPA Evolution Example: KS + MO

• 1999: interchangeable codes
  – KS/MO: Kansas City 10 digit local 913/816 dialing
    • note that 913 and 816 are in the same LATA
PSTN: Addressing
NANP NPA Evolution Example: KS + MO

- 2001: geographic split
  - KS: Wichita keeps 316
  - KS: rest of south replaced by 620
PSTN: Addressing
NANP NPA Evolution Example: KS + MO

• 2001: first overlays planned
  – 975 for KC (1.1M metro pop.); 557 for St. Louis (2.0M pop.)
  – initial plans suspended but codes still reserved
PSTN: Addressing
NANP NPA Growth and Capacity

- NYX NXX XXXX addresses
  - was in danger of exhaustion within next decade
  - growth has slowed with FAX disuse and shift to mobile only
  - repartition of existing geographical boundaries impractical

- Expansion with additional digits: ITU E.164 allows 15

- Options
  - use reserved N9X for more NPA digits
  - increase number of SLID digits

- Fixed vs. variable length codes
  - much of world used variable length
  - US reluctant to change fixed-length tradition
PSTN: Addressing
NANP (WZ1)

- **Nomenclature:** $N = \{2...9\}; \ X = \{0...9\}; \ Y = \{0...8\} \ \ 0/1 = \{0|1\}$

<table>
<thead>
<tr>
<th></th>
<th>1947</th>
<th>1974</th>
<th>1995</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CC</strong></td>
<td></td>
<td>$N^+$</td>
<td>$N^+$</td>
<td></td>
</tr>
<tr>
<td><strong>NPA</strong></td>
<td>$N^0/1X$</td>
<td>$N^0/1X$</td>
<td>$NYX$ [712]</td>
<td>$N9XX$ [7920]</td>
</tr>
<tr>
<td><strong>Central office</strong></td>
<td>$NNX$ [640]</td>
<td>$NXX$ [792]</td>
<td>$NXX$ [792]</td>
<td></td>
</tr>
<tr>
<td><strong>SLID</strong></td>
<td>$XXXX$ [10000]</td>
<td>$XXXX$ [10000]</td>
<td>$XXXX$ [10000]</td>
<td>$XXXX^+$ [10^n]</td>
</tr>
<tr>
<td><strong>free</strong></td>
<td></td>
<td>800 [1]</td>
<td>8xx [8]</td>
<td>?</td>
</tr>
<tr>
<td><strong>IDDD +</strong></td>
<td></td>
<td>011</td>
<td>011</td>
<td>011</td>
</tr>
<tr>
<td><strong>Service code</strong></td>
<td>N11 [8]</td>
<td>N11</td>
<td>N11</td>
<td>?</td>
</tr>
<tr>
<td><strong>Toll center</strong></td>
<td>0XX</td>
<td>0XX</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>System code</strong></td>
<td>$\theta/1XX$ [200]</td>
<td>$\theta/1XX$ [200]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PSTN: Addressing
NANP (WZ1) \(^0/1\) Call Type Codes

- Trunk access and operator codes

<table>
<thead>
<tr>
<th>(^0/1X)</th>
<th>Use</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>local operator</td>
<td>requires digit timeout</td>
</tr>
<tr>
<td>00</td>
<td>toll operator</td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>IDDD access</td>
<td></td>
</tr>
<tr>
<td>1N...</td>
<td>non-local (toll) call</td>
<td>8/11 digit number</td>
</tr>
<tr>
<td>1N11</td>
<td></td>
<td>1+3 digit service code</td>
</tr>
<tr>
<td>N...</td>
<td>local call</td>
<td>7/10 digit number</td>
</tr>
<tr>
<td>N11</td>
<td></td>
<td>3 digit service code</td>
</tr>
</tbody>
</table>
PSTN: Addressing
NANP (WZ1) Service Codes

- N11 codes administered by FCC

<table>
<thead>
<tr>
<th>N11</th>
<th>Use</th>
<th>Basis</th>
</tr>
</thead>
<tbody>
<tr>
<td>211</td>
<td>community information &amp; referral services</td>
<td>FCC</td>
</tr>
<tr>
<td>311</td>
<td>non-emergency police &amp; other government services</td>
<td>FCC</td>
</tr>
<tr>
<td>411</td>
<td>local directory assistance</td>
<td>traditional</td>
</tr>
<tr>
<td>511</td>
<td>traffic and transportation info (US)</td>
<td>FCC</td>
</tr>
<tr>
<td>611</td>
<td>repair service</td>
<td>traditional</td>
</tr>
<tr>
<td>711</td>
<td>TRS (telecommunications relay service)</td>
<td>FCC</td>
</tr>
<tr>
<td>811</td>
<td>dig safe</td>
<td>FCC</td>
</tr>
<tr>
<td>911</td>
<td>emergency</td>
<td>FCC</td>
</tr>
</tbody>
</table>
PSTN: Addressing
NANP (WZ1) Reserved NPAs

- NPAs reserved for special use

<table>
<thead>
<tr>
<th>NXX</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nxx</td>
<td>ERC: easily recognisable codes (x = 2nd = 3rd digit)</td>
</tr>
<tr>
<td>37X</td>
<td>reserved for future contiguous block use</td>
</tr>
<tr>
<td>96X</td>
<td>reserved for future contiguous block use</td>
</tr>
<tr>
<td>N9X</td>
<td>reserved for future expansion to 4-digit NPA</td>
</tr>
</tbody>
</table>
PSTN: Addressing
NANP (WZ1) Non-Geographic NPAs

- NPAs not tied to geographical region

<table>
<thead>
<tr>
<th>NXX</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>456</td>
<td>inbound international carrier identification</td>
</tr>
<tr>
<td>5XX</td>
<td>personal communication service (&quot;follow-me&quot;)</td>
</tr>
<tr>
<td>600</td>
<td>Canadian services</td>
</tr>
<tr>
<td>700</td>
<td>IXC (interexchange) carrier services</td>
</tr>
<tr>
<td>710</td>
<td>GETS: US government emergency telecom service</td>
</tr>
<tr>
<td>8xx</td>
<td>TRS (telecommunications relay service)</td>
</tr>
<tr>
<td>900</td>
<td>premium services (additional billing)</td>
</tr>
</tbody>
</table>
### PSTN: Addressing
NANP (WZ1) Special CO Codes

- Restricted and special use CO codes

<table>
<thead>
<tr>
<th>NXX-XXXX</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>555-XXXX</td>
<td>NANP-wide assignment</td>
</tr>
<tr>
<td>555-01XX</td>
<td>fictitious use (media and advertising)</td>
</tr>
<tr>
<td>555-1212</td>
<td>directory assistance</td>
</tr>
<tr>
<td>555-1313</td>
<td>Bell Canada</td>
</tr>
<tr>
<td>700-4141</td>
<td>intraLATA carrier verification</td>
</tr>
<tr>
<td>950-XXXX</td>
<td>CAC: carrier access code (CIC xxxx)</td>
</tr>
<tr>
<td>958-XXXX</td>
<td>test code</td>
</tr>
<tr>
<td>959-XXXX</td>
<td>test code</td>
</tr>
<tr>
<td>976-XXXX</td>
<td>information delivery services (additional billing)</td>
</tr>
</tbody>
</table>
PSTN: Addressing
NANP (WZ1) Local Number Portability

- Problem: users must change phone number
  - when they move
  - when they change providers (ILEC and new CLECs)
  - when they convert to mobile telephones for home use
  - when they change mobile providers

Solution?
PSTN: Addressing NANP (WZ1) Local Number Portability

- Problem: users must change phone number
- Solution: local number portability (LNP)
  - *strongly* resisted by providers
    - ILEC monopolistic customer base
    - mobile providers want to avoid customer churn
      - alternative: signup deals with long contracts and heavy penalties
- Mandated by FCC
  - 2003
    - LNP for wireline providers *within* an NPA
    - WLNP (wireless LNP) 100 largest MSAs (metro statistical areas)
  - 2004 WLNP in all areas (NPA not required to be portable)
PSTN Signalling

Overview

• Signalling needed for
  – call setup and routing
  – call modification
  – call release

• Signalling between
  – telephone and central office (local loop)
  – inter-office (trunks)
PSTN Signalling

In-Channel Signalling

• In-channel (per-trunk) signalling
  – same transmission links for voice and signalling
  – in-band: uses voice frequencies
  – out-of-band: non-voice frequencies

• Advantages
  – does not require distinct infrastructure
  – compatible with simple POTS telephones

• Disadvantages
  – user can hack into network
    • blue boxes, etc.
  – security by obscurity
PSTN Signalling
Common Channel Signalling and SS7

- **Common channel signalling**
  - distinct data network for signalling (control plane)
    - *out-of-band* signalling
  - connects control processing of switches

- **SS7: signalling system no. 7**
  - evolution of SS6 for capabilities beyond voice telephony
    - support for ISDN (integrated services data network)
    - variable message lengths and higher link rates
  - deployed beginning in the 1980s
    - initially for 800 freephone number translation in the US
PSTN Common Channel Signalling

SS7 Network Architecture

- SS7 signalling network components and architecture
  - telephone switch
  - **SSP**: service switching point
    - adjunct signalling processor
  - **STP**: signalling transfer point
    - signalling message switch
  - **SCP**: service control points
    - interface to databases
    - e.g. 800 number translation

Diagram:
- SSP
- STP
- SCP
- Associated signalling channel
- Non-associated signalling links
- Redundant paths
- Voice trunks
PSTN Common Channel Signalling
SS7 Network Protocol Stack

- SS7 protocol stack
  - ITU-T Q.700
  - TUP: telephone user part
  - ISUP: ISDN user part
  - TC: transaction capabilities
  - SSCP: signalling connection control part
  - MTP: message transfer part
    - level 3
    - level 2
    - level 1
PSTN Common Channel Signalling
SS7 Protocols: ISUP

- **ISUP**: ISDN user part
  - ITU Q.761–766  ANSI T1.113
    - signalling for ISDN: integrated services data network
      - data, voice, video
    - uses MTP for message transport
    - uses SSCP for E2E signalling
PSTN Common Channel Signalling

SS7 Protocols: ISUP Messages

- **ISUP signalling message examples**
  - call establishment
    - IAM: initial address message (call SETUP)
    - ACM: address complete message (call PROCEEDING)
    - ANM: answer message (call CONNECT)
  - call modification
  - call teardown
    - REL: release (call RELEASE)
    - RLC: release complete (call RELEASE ack)
PSTN Common Channel Signalling

SS7 Protocols: Telephony Call Example

- Telephony signalling
PSTN Common Channel Signalling

SS7 Protocols: Telephony Call Example

- Telephony signalling
  - caller goes off-hook
PSTN Common Channel Signalling
SS7 Protocols: Telephony Call Example

- Telephony signalling
  - caller goes off-hook
  - caller receives dial tone
PSTN Common Channel Signalling
SS7 Protocols: Telephony Call Example

- Telephony signalling
  - caller goes off-hook
  - caller receives dial tone
  - caller enters digits
PSTN Common Channel Signalling
SS7 Protocols: Telephony Call Example

- Telephony signalling
  - caller goes off-hook
  - caller receives dial tone
  - caller enters digits
  - calling switch sends IAM
PSTN Common Channel Signalling
SS7 Protocols: Telephony Call Example

- Telephony signalling
  - caller goes off-hook
  - caller receives dial tone
  - caller enters digits
  - calling switch sends **IAM**
  - called switch **rings** and returns **ACM**
PSTN Common Channel Signalling
SS7 Protocols: Telephony Call Example

- Telephony signalling
  - caller goes off-hook
  - caller receives dial tone
  - caller enters digits
  - calling switch sends IAM
  - called switch rings and returns ACM
  - caller hears ring tone
PSTN Common Channel Signalling

SS7 Protocols: Telephony Call Example

- Telephony signalling
  - caller goes off-hook
  - caller receives dial tone
  - caller enters digits
  - calling switch sends IAM
  - called switch rings and returns ACM
  - caller hears ring tone
  - callee answers and goes off-hook
PSTN Common Channel Signalling
SS7 Protocols: Telephony Call Example

- Telephony signalling
  - caller goes off-hook
  - caller receives dial tone
  - caller enters digits
  - calling switch sends IAM
  - called switch rings and returns ACM
  - caller hears ring tone
  - callee answers and goes off-hook
  - called switch returns ANM; connected
PSTN Common Channel Signalling

SS7 Protocols: Telephony Call Example

- Telephony signalling
  - caller goes off-hook
  - caller receives dial tone
  - caller enters digits
  - calling switch sends IAM
  - called switch rings and returns ACM
  - caller hears ring tone
  - callee answers and goes off-hook
  - called switch returns ANM; connected
  - parties talk
PSTN Common Channel Signalling

SS7 Protocols: Telephony Call Example

- Telephony signalling
  - caller goes off-hook
  - caller receives dial tone
  - caller enters digits
  - calling switch sends IAM
  - called switch rings and returns ACM
  - caller hears ring tone
  - callee answers and goes off-hook
  - called switch returns ANM; connected
  - parties talk
  - parties hang up by going on-hook
PSTN Common Channel Signalling
SS7 Protocols: Telephony Call Example

- Telephony signalling
  - caller goes off-hook
  - caller receives dial tone
  - caller enters digits
  - calling switch sends IAM
  - called switch rings and returns ACM
  - caller hears ring tone
  - callee answers and goes off-hook
  - called switch returns ANM; connected
  - parties talk
  - parties hang up by going on-hook
  - REL and RLC messages terminate
Network Layer

Internet

NL.1 Network layer functions and services
NL.2 Network signalling paradigms
NL.3 Switches and packet structure
NL.4 Examples
   NL.4.1 PSTN
   NL.4.2 Internet: DNS, IP, ICMP and IPv6
NL.5 Fast datagram routers
Internet Functions and Protocols

- Addressing
- Forwarding
- Routing
- Signalling
- Traffic Management
Internet
Functions and Protocols

- *Addressing?*
- Forwarding
- Routing
- Signalling
- Traffic Management
Internet
Functions and Protocols: Addressing

• **Addressing**: IP
  – IPv4 [RFC 0791 / STD 0005]
  – subnetting [RFC 0950 / STD 0005]
  – CIDR [RFC 1519]
  – MAC layer address resolution: ARP [RFC 0826 / STD 0037]
  – IPv6 [RFC 2460, 3513]

• **Forwarding**?

• Routing
• Signalling
• Traffic Management
Internet Functions and Protocols: Forwarding

- **Addressing**: IP
- **Forwarding**: IP
  - IP address lookup in routers
- **Routing?**
- Signalling
- Traffic Management
Internet
Functions and Protocols: Routing

- Addressing: IP
- Forwarding: IP
- **Routing**: *Lecture NR*
  - interdomain EGP (exterior gateway protocol): BGP
  - intradomain IGP (interior gateway protocol): RIP, OSPF, ...
- **Signalling**?
- Traffic Management
Internet
Functions and Protocols: Signalling

• Addressing: IP
• Forwarding: IP
• Routing: BGP + IGPs
• **Signalling**: ICMP
  – ICMPv4 [RFC 0792 / STD 0005]
  – ICMPv6 [RFC 2463]
• **Traffic Management?**
Internet
Functions and Protocols: Traffic Management

- Addressing: IP
- Forwarding: IP
- Routing: BGP + IGPs
- Signalling: ICMP
- Traffic management: Lecture TQ
  - congestion avoidance and control (e.g. RED)
  - fair queuing
  - DiffServ
  - IntServ
Internet Architecture
Overview

• Interconnection of service provider networks
  – tier 1 service providers,
  – lower tier (2 and 3) service provider networks
  – access networks
  – enterprise, campus, home

• No organised structure
  – since end of NSFNET

• Service model
  – best effort: no performance guarantees
  – providers may offer SLAs (service level agreements)
    • generally by over-provisioning of network infrastructure
Internet Architecture
Overview: Regulation

• Minimal regulation
  – ICANN [www.icann.org]
    Internet Corporation for Assigned Names and Numbers
    • administers DNS TLDs (top level domains) & IP address blocks
    • registrars administer second level domains and subnets
  – IANA [www.iana.org]
    Internet Assigned Numbers Authority
    • administers various number- and name-spaces

• Repeated attempts to regulate
  – by government and traditional PSTN carriers
  – largely unsuccessful so far
  – ultimately futile due to global scope of Internet
Internet Architecture
Overview: Standards

- **IAB (Internet Architecture Board)** [www.iab.org]
  - advisory role to ISoc (Internet Society) [www.isoc.org]
  - committee of Internet Engineering Task Force (IETF)
  - Internet Research Task Force (IRTF) [www.irtf.org]

- **IETF protocol standards** [www.ietf.org]
  - but vendors implement what they wish...
    subject to ...
  - service providers who offer what they wish...
    subject to ...
  - customer demand
Internet Architecture
Design Principles

• ARPANET design principles
  – simple (relatively) stateless core for resiliency
  – most functionality at end systems
  – end-to-end addressing transparency
  – hourglass model (actually a bit later than original design)
    • any transport protocol over IP over any link layer

• End-to-end arguments
  – what functionality must be located on end systems
  – what functionality should be in the network for performance

Recall: these two are not the same thing
Names and Addresses

Overview

• **Address**: identifier of a node
  - may only be machine readable (binary address)
    • e.g. 10000001 11101101 01010111 00010010
  - may be represented by human readable number
    • e.g. 129.237.87.18 or 148.88.3.47
  - may be indirection by human friendly form (e.g. DNS name)
    • e.g. www.eecs.ku.edu or www.comp.lancs.ac.uk

• **Name**: *global persistent* identifier of an entity
  - e.g. James Philip Guenther Sterbenz

• Unfortunately “name” is commonly used for both
Network Layer

Internet: DNS

NL.1  Network layer functions and services
NL.2  Network signalling paradigms
NL.3  Switches and packet structure
NL.4  Examples
   NL.4.1  PSTN
   NL.4.2  Internet: DNS, IP, ICMP and IPv6
NL.5  Fast datagram routers
Domain Name System Overview

• DNS: domain name system [RFC 1034/1035 / STD 0013]
  – directory service for the Internet
  – resolves *hostnames* to IP addresses
    • hostname is merely a human friendly address redirection

• DNS is
  – technically an application layer protocol
    • runs over TCP or UDP
  – an essential network infrastructure
    *why?*
Domain Name System
Overview

- DNS: domain name system [RFC 1034/1035 / STD 0013]
  - directory service for the Internet
  - resolves *hostnames* to IP addresses
    - hostname is merely a human friendly address redirection

- DNS is
  - technically an application layer protocol
    - runs over TCP or UDP
  - in reality an essential network infrastructure service
    - http://www.eecs.ku.edu
      - rather than
      - http://129.237.87.18
Domain Name System Services

- Hostname to IP addresses *resolution*
  - e.g. www.eecs.ku.edu to 129.237.87.18
Domain Name System
Services

- Hostname to IP addresses *resolution*
- Hostname aliasing to canonical name
  - allows stable DNS names for Web and mail servers, e.g.
    - `www.sterbenz.org` to `abell.lunarpages.com:80`
    - `www.ku.edu` to `raven.cc.ku.edu:80`
    - `mail.ittc.ku.edu` to `stephens.ittc.ku.edu:25`
  - allows proper default behavior on incoming port 80
    - `http://example.com` to `http://www.example.com`
    - many servers not properly configured to do this
Domain Name System

Services

- Hostname to IP addresses *resolution*
- Hostname aliasing to canonical name
- Load distribution
  - set of IP addresses for one canonical name
  - typically used for replicated Web servers
  - e.g. `www.cnn.com` to `264.236.{255|224|226}.n`
Domain Name System Implementation

- Distributed database implemented in a hierarchy
  - many *name servers*
  - no relationship to IP addressing structure!
- Distributed implementation
  - improves scalability
  - decentralises administration
Domain Name System

Name Structure

• Fully qualified domain name (FQDN): … . ⟨SLD⟩ . ⟨TLD⟩
• TLD: top level domain
  – originally assigned by IANA
  – now assigned by ICANN [www.icann.org/tlds]
  – gTLD: generic TLD
  – ccTLD: country-code TLD
• SLD: second level domain
  – may be defined by TLD policy
  – may be assigned by TLD registrar to domain owner
• nLD: n-th level domain
  – may be defined by TLD or SLD policy
  – may be assigned by registrar/sub-registrar to domain owner
Domain Name System
Name Structure: Generic TLDs

- **gTLD**: generic top level domain
  - assigned by ICANN [www.icann.org/tlds](http://www.icann.org/tlds)
  - gTLD types
    - reserved
    - infrastructure
    - unrestricted and restricted
    - unsponsored and sponsored
    - unrestricted
    - pseudo

- **SLD**: second level domain
  - policy based on each TLD
  - direct indicates that SLD assigned to domain owner
Domain Name System

Name Structure: Reserved gTLDs

- DNS names reserved for special use [RFC 2606]

<table>
<thead>
<tr>
<th>TLD</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>.test</td>
<td>reserved for DNS code testing</td>
</tr>
<tr>
<td>.example</td>
<td>online and documentation examples for valid DNS name</td>
</tr>
<tr>
<td>.invalid</td>
<td>online and documentation examples for invalid DNS name</td>
</tr>
<tr>
<td>.localhost</td>
<td>resolves to loopback address (typically 127.0.0.1)</td>
</tr>
</tbody>
</table>
Domain Name System
Name Structure: Unrestricted gTLDs

• Among original seven 1980 TLDs [RFC 0920]
• Originally partitioned among
  – commercial entities and businesses
  – network service providers
  – non-profit organisations
• NSI mismanaged assignments & ignored IANA intent
  – registered to *anyone* willing to pay annual fee
  – distinction is now almost meaningless

<table>
<thead>
<tr>
<th>TLD</th>
<th>Original Use</th>
<th>Registrar</th>
</tr>
</thead>
<tbody>
<tr>
<td>.com</td>
<td>commercial</td>
<td>many</td>
</tr>
<tr>
<td>.net</td>
<td>network service provider</td>
<td>many</td>
</tr>
<tr>
<td>.org</td>
<td>non-profit organisation</td>
<td>ISOC Public Interest Reg.</td>
</tr>
</tbody>
</table>
Domain Name System
Name Structure: Restricted gTLDs

- Among original
  - .edu .gov .mil among seven 1980 TLDs [RFC 0920]
  - .nato created in late 1980s but replaced by .int
  - .int created in late 1980s for international use

- Registrar sets policy and restricts use

<table>
<thead>
<tr>
<th>TLD</th>
<th>Use</th>
<th>Registrar</th>
</tr>
</thead>
<tbody>
<tr>
<td>.edu</td>
<td>higher educational institution</td>
<td>EDUCAUSE</td>
</tr>
<tr>
<td>.gov</td>
<td>US government</td>
<td>US GSA</td>
</tr>
<tr>
<td>.mil</td>
<td>US DOD (military)</td>
<td>US DOD DISA</td>
</tr>
<tr>
<td>.int</td>
<td>international</td>
<td>IANA</td>
</tr>
<tr>
<td>.nato</td>
<td>NATO originally</td>
<td>replaced by .int</td>
</tr>
</tbody>
</table>
Domain Name System
Name Structure: Un-sponsored gTLDs

- Among second batch of seven new TLDs in 2000
- ICANN responsible for TLD policy
  - registry delegated under ICANN contract
  - .biz and .info use and policies loosely applied, at best

<table>
<thead>
<tr>
<th>TLD</th>
<th>SLD</th>
<th>Use</th>
<th>Registrar</th>
</tr>
</thead>
<tbody>
<tr>
<td>.biz</td>
<td>direct</td>
<td>business</td>
<td>Neulevel (Neustar)</td>
</tr>
<tr>
<td>.info</td>
<td>direct</td>
<td>information</td>
<td>Afilias</td>
</tr>
<tr>
<td>.name</td>
<td>&lt;first&gt;.&lt;last&gt;</td>
<td>individuals</td>
<td>Global Name Registry</td>
</tr>
<tr>
<td>.pro</td>
<td>&lt;3-letter-prof-code&gt; direct</td>
<td>professions</td>
<td>RegistryPro</td>
</tr>
</tbody>
</table>
Domain Name System
Name Structure: Unrestricted gTLDs

- ICANN accepted recommendation to open TLDs
  - applicants can choose (almost) any TLD
  - will accommodate non-Latin TLD names
  - implementation has begun; over 1000 new

- Mechanisms will be used for trademarks, e.g.
  - **sunrise**: applications only from trademark holders
  - **land-rush**: applications from anyone meeting requirements

- This is widely recognised as a **terrible** idea
Domain Name System
Name Structure: Unrestricted gTLD Problems

• Expensive: $185,000 application fee
  – biased against third world
  – biased against small businesses
    • can’t afford trademark protection

• Multiple registrations for trademark protection
  – make registrars wealthy; ICN estimates $200M/yr for .xxx
  – e.g. needing apple.sucks, apple.xxx

• Legal nightmare
  – reopen disputes long settled in SLDs
    • e.g. apple.com vs. applemusic.com
  – will make the lawyers wealthy
Domain Name System
Name Structure: Country Code TLD

- ccTLD: country code top level domain
  - based on [ISO 3166]
  - administered per nation
    - some with profit potential, e.g. .tv (Tuvalu)
    - many *domain hacks* possible e.g. jam.es
  - examples
    - .us .ca .uk .de .ch .jp
Domain Name System

Name Structure: Examples

- Example DNS names
  - deep-thought . eecs . ku . edu
    user 4LD       dept 3LD   inst SLD  gTLD
  - wopr . labs . gte . com
    user 4LD       div 3LD   corp SLD  gTLD
  - www . tik . ee . ethz . ch
    host 5LD      group 4LD  dept 3LD  inst SLD  Swiss ccTLD
  - www . comp . lancs . ac . uk
    host 5LD      dept 4LD   inst 3LD  UK-SLD  UK ccTLD
  - jam.es
    SLD  ccTLC     domain hack for name “James”
  - pbs.tv
    ccTLD with 2nd meaning (e.g. television)
Domain Name System

Name Server Structure

- **Example:** resolve `www.amazon.com`
  - client queries a root server to find .com TLD DNS server
  - client queries .com TLD server to get `amazon.com` server
  - client queries `amazon.com` DNS server to get IP address
Domain Name System

Name Servers: Root

- Root name servers  www.root-servers.org
  - responsible for resolution to proper TLD server
  - 13 root servers contracted identified as A–M

- Configuration file named.root
  - used by other DNS servers to locate root servers
  - contain list of A–M servers and their IP addresses
  - ftp://ftp.internic.net/domain/named.root
Domain Name System

Root Name Servers

A. VeriSign, Dulles VA
B. ISI, Marina Del Rey CA
C. Cogent, Herndon VA + 3
D. UMd, College Park MD
E. NASA Ames, Mountain View CA
F. ISC, 37 sites
G. US DOD NIC
H. ARL, Aberdeen MD
I. Autonomica/NORDUnet, Stockholm + 28 mirrors
J. VeriSign, Dulles VA + 16 mirrors
K. RIPE, London + 16 mirrors
L. ICANN, Los Angeles Ca
M. WIDE, Tokyo

Selected mirrors shown
Domain Name System

Name Servers: TLD

- Root name server
- Top-level domain (TLD) server
  - responsible for all gTLD and ccTLD resolution
  - ICANN contracts each gTLD name server
  - nations responsible for administering or delegating ccTLD
Domain Name System

Name Servers: Authoritative

- Root name server
- Top-level domain (TLD) server
- Authoritative DNS server
  - DNS servers for organisation or corporate entity
  - provide *authoritative* hostname resolution
    - hosts within its own domain
    - e.g. *ku.edu* Web and mail servers
  - can be maintained by
    - organization itself (e.g. *ku.edu*)
    - outsourced to service provider (e.g. *sterbenz.org*)
Domain Name System

Name Servers: Local

- Root name server
- Top-level domain (TLD) server
- Authoritative DNS server
- Local name server (default name server)
  - maintained by each service provider or enterprise
  - resolves host DNS queries
  - acts as a proxy, forwards query into hierarchy
Domain Name System
Updating and Caching Records

• Root name server
  – contacted by local name server that can not resolve name
  – contacts authoritative name server if mapping not known
  – returns mapping to local name server

• When name server learns mapping it is \textit{cached}
  – cache entries timeout (disappear) after some time

• TLD servers typically cached in local servers
  – thus root name servers not often visited

• Update/notify mechanisms [RFC 2136]
Domain Name System

Essential Tool: whois

• whois
  – client–server protocol
  – server listens to port 43
    • uses whois database (attempts underway to standardise)
    • originally stored information on people (Internet white pages)
    • now stores domain registry information

• Client usage (CLI)
  – whois (SLD).(TLD)
  – whois (3LD).(SLD).(ccTLD) some structured ccTLD: .uk
  – Web interface: http://www.internic.net/whois
    • only for gTLDs (but not .edu .mil .gov)

  try it whois ku.edu
       whois lancs.ac.uk
Domain Name System

Essential Tool: dig

- **dig** (domain information grouper)
  - client program to query DNS server
  - more comprehensive information than **nslookup**

- **Usage (CLI)**
  - `dig <DNS-name>`
    - returns DNS records and IP address(es) of `<DNS-name>`
  - `dig -x <IP-address>`
    - reverse lookup: returns DNS name and records of `<IP-address>`

*try it*

- `dig www.ku.edu`
- `dig -x 129.237.33.3`
- `dig www.cnn.com`
Domain Name System
Essential Tool: nslookup

- **nslookup** (name server lookup)
  - client program to query DNS server
  - use when dig not available (e.g. Windows boxes)

- **Usage (CLI)**
  - `nslookup`
    - returns DNS server of local host and enters interactive mode
  - `nslookup <DNS-name>`
    - returns IP address(es) of `<DNS-name>`
  - `nslookup <IP-address>`
    - reverse lookup: returns DNS name of `<IP-address>`

**try it**
- `nslookup www.ku.edu`
- `nslookup 129.237.33.3`
- `nslookup www.cnn.com`
Domain Name System
Web-Based Tools

• Many DNS and IP utilities on the Web
  – http://dnsstuff.com is particularly comprehensive
  – http://kloth.net/services

• Caveats
  – services aren’t being invoked locally

implication?
Domain Name System
Web-Based Tools

• Many DNS and IP utilities on the Web
  – http://dnsstuff.com is particularly comprehensive
  – http://kloth.net/services

• Caveats
  – services aren’t being invoked locally
    • e.g. ping will still test liveness but delay number is useless
    • e.g. traceroute will *not* trace route from local machine
    • local DNS information will not be obtained
  – pages come and go on the Web
Domain Name System
DNS Lookup Iterated Query Example

- DNS iterated lookup example
  - jpgs.ittc.ku.edu needs IP address of gaia.cs.umass.edu
  - ns1.ittc.ku.edu is local DNS server
    129.237.125.220 obtained via DHCP

jpgs.ittc.ku.edu

\[\text{gaia.cs.umass.edu}\]
Domain Name System

DNS Lookup Iterated Query Example

- DNS iterated lookup example
  1Q. A? gaia.cs.umass.edu to local DNS server
  ns1.ittc.ku.edu 129.237.125.220

1Q. A? gaia.cs.umass.edu to
local DNS server
ns1.ittc.ku.edu 129.237.125.220

jpgs.ittc.ku.edu  gaia.cs.umass.edu
Domain Name System
Iterated Query Example

- DNS lookup example
  1Q. local DNS query
  2Q. A? gaia.cs.umass.edu to root nameserver in named.root
  h.rootservers.net 128.63.2.53

- Diagram:
  - Local DNS query
  - Root nameserver: h.rootservers.net 128.63.2.53
  - Queries:
    - 1Q. local DNS query
    - 2Q. A? gaia.cs.umass.edu
  - Responses:
    - jpgs.ittc.ku.edu
    - gaia.cs.umass.edu
Domain Name System
Iterated Query Example

- DNS lookup example
  1Q. local DNS query
  2Q. A? gaia.cs.umass.edu to root nameserver in named.root
  2R. list of .edu NSs including NS a3.nstld.com
      A 192.5.6.32

- DNS diagram:
  - root
  - .edu
  - local
  - a3.nstld.com 192.5.6.32
  - jpgs.ittc.ku.edu
  - gaia.cs.umass.edu

1Q. local DNS query
2Q. A? gaia.cs.umass.edu to root nameserver in named.root
2R. list of .edu NSs including
    NS a3.nstld.com
    A 192.5.6.32
Domain Name System
Iterated Query Example

- DNS lookup example
  1Q. local DNS query
  2QR. root DNS query/response
  3Q. A? gaia.cs.umass.edu to TLD a3.nstld.com 192.5.6.32
Domain Name System
Iterated Query Example

• DNS lookup example
  1Q. local DNS query
  2QR. root DNS query/response
  3Q. A? gaia.cs.umass.edu to TLD a3.nstld.com 192.5.6.32
  3R. list of .edu NSs including
      NS ns1.umass.edu
      A 128.119.100.21
      NS unix1.cs.umass.edu
      A 128.119.40.22
      note: contains L3 & L4 entries use most specific & authoritative entry
      jpgs.ittc.ku.edu
      gaia.cs.umass.edu
Domain Name System
Iterated Query Example

- DNS lookup example
  1Q. local DNS query
  2QR. root DNS query/response
  3QR. TLD DNS query/response
  4Q. A? gaia.cs.umass.edu to auth unix1.cs.umass.edu
      128.119.40.22
Domain Name System
Iterated Query Example

• DNS lookup example
  1Q. local DNS query
  2QR. root DNS query/response
  3QR. TLD DNS query/response
  4Q. A? gaia.cs.umass.edu to auth unix1.cs.umass.edu
      128.119.40.22
  4R. A 128.119.245.12
Domain Name System
Iterated Query Example

- DNS lookup example
  1Q. local DNS query
  2QR. root DNS query/response
  3QR. TLD DNS query/response
  4QR. auth DNS query/response
  1R. A 128.119.245.12
Domain Name System
Iterated Query Example

- DNS lookup example
  1Q. local DNS query
  2QR. root DNS query/response
  3QR. TLD DNS query/response
  4QR. auth DNS query/response
  1R. A 128.119.245.12
  5Q. ping gaia.cs.umass.edu
Domain Name System
Iterated Query Example

• DNS lookup example
  1Q. local DNS query
  2QR. root DNS query/response
  3QR. TLD DNS query/response
  4QR. auth DNS query/response
  1R. A 128.119.245.12
  5Q. ping gaia.cs.umass.edu
  5R. ...reply from 128.119.245.12...
Domain Name System
Iterated Query Example

Iterated query performance?
Domain Name System
Iterated Query Performance

- Iterated query performance problems
  - multiple round trips per host query
  - at least 4, perhaps more

Can we do better?
Domain Name System

Caching

- Iterated query performance
- Caching
  - each name server caches queries
  - significantly reduces
    - load on root and TLD servers
    - round trip latency typically 1 or 2
  - hosts cache mapping
    - no resolution for recent mappings
Domain Name System

Reverse Lookup

- DNS provides mapping function
  - hostname to IP address

Problem: how to get hostname given IP address?
Domain Name System
Reverse Lookup

- DNS provides mapping function
  - hostname to IP address
- Reverse lookup:
  - map IP address to hostname
Domain Name System
Reverse Lookup Example

- Reverse lookup example
  - `jpgs.ittc.ku.edu` needs hostname of `128.119.245.12`

```
jpgs.ittc.ku.edu
128.119.245.12
```
Domain Name System
Reverse Lookup Example

- DNS iterated lookup example
  1Q. PNTR? 12.245.119.128.in-addr.arpa to local DNS server
  ns1.ittc.ku.edu 129.237.125.220

```
local
129.237.125.220

jpgs.ittc.ku.edu 128.119.245.12
```

```java
ns1.ittc.ku.edu 129.237.125.220
```
Domain Name System
Reverse Lookup Example

- DNS lookup example
  1Q. local reverse DNS query
  2Q. PNTR? 12.245.119.128.in-addr.arpa to
      root nameserver in named.root

root

local

jpgs.ittc.ku.edu  128.119.245.12
Domain Name System
Reverse Lookup Example

- DNS lookup example
  1Q. local reverse DNS query
  2Q. PNTR? 12.245.119.128.in-addr.arpa to root nameserver in named.root
  2R. NS figwort.arin.net A 192.42.93.32 zone 128.in-addr.arpa

1. local reverse DNS query
2. PNTR? 12.245.119.128.in-addr.arpa to root nameserver in named.root
3. NS figwort.arin.net A 192.42.93.32 zone 128.in-addr.arpa
Domain Name System
Reverse Lookup Example

- DNS lookup example
  1Q. local reverse DNS query
  2QR. root reverse DNS query
  3Q. PNTR? 12.245.119.128.in-addr.arpa to RIR figwort.arin.net 192.5.6.32

- figwort.arin.net 192.42.93.32
- jpgs.ittc.ku.edu 128.119.245.12
Domain Name System
Reverse Lookup Example

- DNS lookup example
  1Q. local reverse DNS query
  2QR. root reverse DNS query
  3Q. PNTR? 12.245.119.128.in-addr.arpa to
      RIR figwort.arin.net 192.5.6.32
  3R. NS ns1.umass.edu
      A 128.119.100.21
      zone 119.128.in-addr.arpa
Domain Name System
Reverse Lookup Example

• DNS lookup example
  1Q. local reverse DNS query
  2QR. root reverse DNS query
  3QR. RIR reverse DNS query
  4Q. PNTR? 12.245.119.128.in-addr.arpa to
       ns1.umass.edu 128.119.100.21
       jpgs.ittc.ku.edu 128.119.245.12
Domain Name System
Reverse Lookup Example

- DNS lookup example
  1Q. local reverse DNS query
  2QR. root reverse DNS query
  3QR. RIR reverse DNS query
  4Q. PNTR? 12.245.119.128.in-addr.arpa to ns1.umass.edu 128.119.100.21
  4R. PTR gaia.cs.umass.edu A 128.119.245.112

jnps.ittc.ku.edu 128.119.245.12
Domain Name System
Reverse Lookup Example

- DNS lookup example
  1Q. local reverse DNS query
  2QR. root reverse DNS query
  3QR. RIR reverse DNS query
  4QR auth reverse DNS query
  1R. PTR gaia.cs.umass.edu
      A 128.119.245.112
Domain Name System

UDP vs. TCP

• DNS uses UDP or TCP for transport
  why?
Domain Name System

UDP vs. TCP

- DNS uses UDP or TCP for transport
- **UDP**
  - generally used for small queries
    - name resolution and reverse name lookup
  - avoids delay of TCP 3-way handshake
  - automatically retries with TCP if answer too long
- **TCP**
  - generally used for large queries
    - e.g. zone transfers
Domain Name System
DNS Message Header Format

- Header [12b]
- Query/response fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>16b</td>
</tr>
<tr>
<td>flags</td>
<td>8b</td>
</tr>
<tr>
<td>QDCOUNT</td>
<td>16b</td>
</tr>
<tr>
<td>ANCOUNT</td>
<td>16b</td>
</tr>
<tr>
<td>NSCOUNT</td>
<td>16b</td>
</tr>
<tr>
<td>ARCOUNT</td>
<td>16b</td>
</tr>
</tbody>
</table>

- question (variable number, typ=1)
- answer RRs (variable number)
- authority RRs (variable number)
- additional RRs (variable number)
Domain Name System

DNS Query Format

- **QDCOUNT typically = 1**
- **Questions:** DNS queries
  - **QNAME**
  - **QTYPE [16b]**
    - 1=A IP address
    - 2=NS name server
    - 5=CNAME canonical name
    - 12=PTR pointer record
    - 13=HINFO host info
    - 15=MX mail exch. record
    - 252=AXFR req for zone xfer
    - 255=ANY req all records
  - **QCLASS**

<table>
<thead>
<tr>
<th>ID</th>
<th>flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>QDCOUNT</td>
<td>ANCOUNT = 0</td>
</tr>
<tr>
<td>NSCOUNT</td>
<td>ARCOUNT = 0</td>
</tr>
<tr>
<td>QNAME</td>
<td></td>
</tr>
<tr>
<td>QTYPE</td>
<td>QCLASS = 1</td>
</tr>
</tbody>
</table>

© James P.G. Sterbenz
Domain Name System

DNS Records

- DNS records: \langle name, value, type, TTL \rangle
- A record for DNS resolution
  - name: DNS hostname
  - type: IP address
- NS record for resolution to authoritative name server
  - name: domain
  - type: IP address of authoritative name server for domain
- CNAME record for DNS alias resolution
  - name: alias DNS name, e.g. www.ku.edu
  - type: canonical (real) name, e.g. raven.cc.ku.edu
Domain Name System

DNS Records

- DNS records: \langle name, value, type, TTL \rangle
- MX record for mail server alias resolution
  - name: DNS hostname
  - type: IP address
- PTR record for pointer queries (reverse lookup)
  - name: reverse IP DNS name under .in-addr.arpa
  - type: DNS name
- HINFO record
  - name: DNS hostname
  - type: CPU type and operating system
### Domain Name System
#### DNS Response Format

- **3 types of responses**
  - multiple per response

- **Answers**
  - **ANCOUNT**: # answer RRs
  - answer RR(s)

- **Authority NS answers**
  - **NSCOUNT**: # authority RRs
  - authority RRs

- **Additional RRs**
  - **ARCOUNT**: # additional RR
  - additional RR(s)

<table>
<thead>
<tr>
<th>ID</th>
<th>flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>QDCOUNT</td>
<td>ANCOUNT</td>
</tr>
<tr>
<td>NSCOUNT</td>
<td>ARCOUNT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>question</th>
</tr>
</thead>
<tbody>
<tr>
<td>(variable number, typ=1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>answer RRs</th>
</tr>
</thead>
<tbody>
<tr>
<td>(variable number)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>authority RRs</th>
</tr>
</thead>
<tbody>
<tr>
<td>(variable number)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>additional RRs</th>
</tr>
</thead>
<tbody>
<tr>
<td>(variable number)</td>
</tr>
</tbody>
</table>
**Domain Name System**

**DNS Response Format**

- **RRs**: resource records
  - **NAME**: DNS name
    - same format as QNAME
  - **TYPE** (same as QTYPE)
  - **CLASS** 1 = Internet
  - **TTL**: time to live
    - #sec. RR to be cached
    - 0 = do not cache
  - **RDLENGTH**: 
    - RDATA length in bytes
  - **RDATA**: resource data
    - response to query
    - depends on **TYPE**

---

<table>
<thead>
<tr>
<th>ID</th>
<th>flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>QDCOUNT = 0</td>
<td>ANCOUNT</td>
</tr>
<tr>
<td>NSCOUNT</td>
<td>ARCOUNT</td>
</tr>
<tr>
<td>CLASS = 1</td>
<td></td>
</tr>
<tr>
<td>TYPE</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td></td>
</tr>
<tr>
<td>RDLENGTH</td>
<td></td>
</tr>
<tr>
<td>RDATA</td>
<td></td>
</tr>
</tbody>
</table>
Domain Name System
Record Insertion Example

- Example: just created startup “Example Networks, Inc.”
  - register name example.com at a .com registrar
  - provide registrar with auth name server information
    - primary: ns1.example.com 192.0.2.1
    - secondary: ns2.example.com 192.0.2.2
  - registrar inserts RR resource recs. into the .com TLD server:
    example.com, ns1.example.com, NS
    example.com, ns2.example.com, NS
    ns1.example.com, 192.0.2.1, A
    ns2.example.com, 192.0.2.2, A
  - to get email and Web servers to resolve add:
    www.example.com, 192.0.2.10, A
    mail.example.com, 192.0.2.20, MX
Network Layer

Internet: IP, ICMP, and IPv6

NL.1  Network layer functions and services
NL.2  Network signalling paradigms
NL.3  Switches and packet structure
NL.4  Examples
   NL.4.1  PSTN
   NL.4.2  Internet: DNS, IP, ICMP and IPv6
NL.5  Fast datagram routers
Internet Protocol
Overview

• IP (Internet Protocol)
  – waist of the Global Internet
  – addressing and forwarding
  – version 4 (IPv4) [RFC 0791 / STD 0005]

• ICMP (Internet control message protocol)
  – signalling for IP
  – version 4 (ICMPv4) [RFC 0792 / STD 0005]
Internet Protocol
IP Packet Format

- **IP version number**
  - rest of packet
    - *version dependent*

- **Maintained by IANA**
  - $00 = \text{reserved}$
  - $01 = \text{TCP1 [RFC 675]}$
  - $02 = \text{TCP2 [IEN 5]}$
  - $03 = \text{TCP3 [IEN 21] (before TCP/IP split)}$
  - $04 = \text{IPv4}$
  - $05 = \text{ST [IEN 119] (experimental)}$
  - $06 = \text{IPv6}$
  - $07 = \text{TP/IX [RFC 1475] (proposed IPng)}$
  - $08 = \text{PIP [RFC 1621] (proposed IPng)}$
  - $09 = \text{TUBA [RFC 1347] (proposed IPng)}$
  - $15 = \text{reserved}$
Internet Protocol
IPv4 Packet Format: Header Control Fields

- **IP version number** = 04
- **IHL**: header length
  - [32-b words]
- **TOS**: type of service
  - not generally used *Lec TQ*
- **TTL**: time to live *Lec NR*
- **Protocol** to demux
  - TCP, UDP, etc.
- **Header checksum**
  - 1’s comp of 1’s comp \( \Sigma \)
### Internet Protocol

**IPv4 Packet Format: Payload Related Fields**

- **Total length** of datagram
  - header and data [B]
- **Payload**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IHL</td>
<td>Internet Header Length</td>
</tr>
<tr>
<td>TOS</td>
<td>Type of Service</td>
</tr>
<tr>
<td>total length</td>
<td>Total length of the packet</td>
</tr>
<tr>
<td>fragment id</td>
<td>Fragment Identification Number</td>
</tr>
<tr>
<td>flag</td>
<td>Fragment Offset</td>
</tr>
<tr>
<td>frag offset</td>
<td>Flag and Offset of Fragment</td>
</tr>
<tr>
<td>TTL</td>
<td>Time to Live</td>
</tr>
<tr>
<td>protocol</td>
<td>Protocol Number</td>
</tr>
<tr>
<td>header checksum</td>
<td>Header Checksum</td>
</tr>
<tr>
<td>source address</td>
<td>Source Address</td>
</tr>
<tr>
<td>destination address</td>
<td>Destination Address</td>
</tr>
<tr>
<td>options</td>
<td>Options</td>
</tr>
<tr>
<td>(= hl – 20B)</td>
<td>Options Length</td>
</tr>
<tr>
<td>payload</td>
<td>Payload</td>
</tr>
<tr>
<td>(= length – hl – 20B)</td>
<td>Payload Length</td>
</tr>
</tbody>
</table>
Internet Protocol
IPv4 Packet Format: Addresses

- **32 bit IP addresses**
  - host or router interface
- **Destination address**
  - used by forwarding
- **Source address**
  - why?

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>IHL</td>
</tr>
<tr>
<td></td>
<td>TOS</td>
</tr>
<tr>
<td></td>
<td>total length</td>
</tr>
<tr>
<td></td>
<td>fragment id</td>
</tr>
<tr>
<td></td>
<td>flag</td>
</tr>
<tr>
<td></td>
<td>frag offset</td>
</tr>
<tr>
<td></td>
<td>TTL</td>
</tr>
<tr>
<td></td>
<td>protocol</td>
</tr>
<tr>
<td></td>
<td>header checksum</td>
</tr>
<tr>
<td></td>
<td>source address</td>
</tr>
<tr>
<td></td>
<td>destination address</td>
</tr>
<tr>
<td></td>
<td>options</td>
</tr>
<tr>
<td></td>
<td>(= hl – 20B)</td>
</tr>
<tr>
<td></td>
<td>payload</td>
</tr>
<tr>
<td></td>
<td>(= length – hl – 20B)</td>
</tr>
</tbody>
</table>
### Internet Protocol

### IPv4 Packet Format: Addresses

- **32 bit IP addresses**
  - host or router interface
- **Destination address**
  - used by forwarding
- **Source address**
  - may be needed for:
    - network to record senders
    - destination to reply

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>IHL</td>
</tr>
<tr>
<td></td>
<td>TOS</td>
</tr>
<tr>
<td></td>
<td>total length</td>
</tr>
<tr>
<td></td>
<td>fragment id</td>
</tr>
<tr>
<td></td>
<td>flag</td>
</tr>
<tr>
<td></td>
<td>frag offset</td>
</tr>
<tr>
<td>TTL</td>
<td>protocol</td>
</tr>
<tr>
<td></td>
<td>header checksum</td>
</tr>
</tbody>
</table>

- **source address**
- **destination address**

- **options**
  - \(\text{= hl – 20B}\)

- **payload**
  - \(\text{= length – hl – 20B}\)
Internet Protocol

IP Addresses

• All interfaces that use IP have an address
  – host–network interfaces; most hosts have more than one
    • try ifconfig (or ipconfig for Windows) on your laptop
  – router ports

• 32-bit addresses
  – e.g. www.eecs.ku.edu to 129.237.87.18
  – example: www.eecs.ku.edu (resolved via DNS to)
    10000001 111101101 01010111 00010010

• Dotted decimal notation:
  – $b_7b_6 . b_5b_4 . b_3b_2 . b_1b_0$ converted to decimal in $4 \times 8$ b chunks
  – example: 129.237.87.18
Internet Protocol

IP Special Addresses

- **Localhost**
  - 0.0.0.0 during boot
  - 127/8 (127.x.x.x) ; generally 127.0.0.1

- **Link local communication** [RFC 3927]
  - 169.254/16 (169.254.X.X)

- **Broadcast**
  - 255.255.255.255 broadcast for this subnet
  - subnet address followed by all 1s: broadcast on subnet
  - generally restricted in use *why?*

- **Example for documentation**
  - 192.0.2/24 (192.0.2.X)
Internet Protocol
IPv4 Address Assignment

- IP addresses not randomly assigned to hosts
  
  why?
Internet Protocol
IPv4 Address Assignment

- IP addresses not randomly assigned to hosts
  - *every* table would have to contain *every* Internet host
  - *trillions* of entries
Internet Protocol
IPv4 Address Hierarchy

- IP addresses assigned *hierarchically*
  - address aggregation dramatically improves scalability
  - forwarding table only needs to contain *network address*
  - routing advertisements only contain network address prefix

<table>
<thead>
<tr>
<th>ISP_A</th>
<th>ISP_B</th>
<th>Tier1_X</th>
</tr>
</thead>
<tbody>
<tr>
<td>199.31.0</td>
<td>38.66.126.2</td>
<td>200.23.16</td>
</tr>
<tr>
<td>200.23.16.4</td>
<td>200.23.16.12</td>
<td>200.23.16</td>
</tr>
<tr>
<td>200.23.16.45</td>
<td>200.23.16.45</td>
<td>199.31.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B</td>
</tr>
</tbody>
</table>

IPv4 Address Hierarchy:

- IP addresses assigned hierarchically
- Address aggregation dramatically improves scalability
- Forwarding table only needs to contain network address
- Routing advertisements only contain network address prefix
IP Addressing
Class-Based Addressing Hierarchy

<table>
<thead>
<tr>
<th>Class</th>
<th>Networks</th>
<th>Hosts</th>
<th>Net</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>128</td>
<td>16M</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>16K</td>
<td>64K</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>2M</td>
<td>256</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td>1110</td>
<td>multicast address</td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td>1111</td>
<td>reserved</td>
</tr>
</tbody>
</table>

- Divide IP address into 3 level hierarchy
  - class, network address, host address
  - byte aligned
  - simple IP address lookup (3 major cases)
  - class D for multicast addresses *Lecture NR*
IP Addressing
Class-Based Addressing Problems

- Principle behind division
  - A: very large network providers
  - B: large organisations
  - C: LANs

- Reality: rigid structure
  - doesn’t match all organisations perfectly
  - doesn’t match many organisations well
    - especially class B: “three bears problem”

- Inefficient partitioning of address space
  - large fraction of unusable addresses
  - imminent exhaustion of IP address space led to...
IP Addressing

Subnets

16K networks $\times$ 64 subnets $\times$ 1024 hosts

<table>
<thead>
<tr>
<th>net</th>
<th>subnet</th>
<th>host</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Subnets** [RFC 0950 / STD 0005]
  - originally way to divide address class within organisation
  - example: 6b subnet to class B
  - subnet mask

- **Hosts in subnet** share upper IP address bits
  - natural to cluster similar IP addresses
  - efficient IP routing to subnet
  - switched layer 2 LAN with no layer 3 routing

*Lecture LL*
IP Addressing
Classless Addressing (CIDR)

- CIDR: classless interdomain routing [RFC 1519]
  - eliminate assignment of IP address blocks by class
  - $b_7b_6.b_5b_4.b_3b_2.b_1b_0/x$
    - x-bit prefix = arbitrary number of network bits
  - example: 11001000 00010111 00010000 00000000
    - 200.23.16.0/23

- Service providers get variable IP block
  - based on need from RIR (or NIR)

- Significant improvement in IP address use
  - at the cost of significant increase in complexity of IP lookup
IP Address Assignment

Administrative Delegation

- IP address blocks originally assigned directly by IANA
  - as class A, B, or C
  - later managed by InterNIC operated by NSI
- Address assignment now delegated [RFC 2050]
  - IANA manages and allocates
    - http://www.iana.org/assignments/ipv4-address-space
  - regional internet registries (RIR) allocate within their range
    - ARIN, RIPE, APNIC, LACNIC, AfriNIC
    - Number Resource Organization www.nro.net
  - national internet registries operate within APNIC
  - local internet registries: typically ISPs
## IP Address Assignment

### Organisational Subnetworks

- Service provider assigns sub-blocks to subscribers
  - CIDR enables arbitrary subnetting at multiple levels
- Example
  - ISP assigned by 200.23.16.0/20 RIR (regional Internet registry)
  - ISP assigned subnets base on each organisation need

<table>
<thead>
<tr>
<th>ISP&lt;sub&gt;A&lt;/sub&gt;</th>
<th>11001000 00010111 00010000 00000000</th>
<th>200.23.16.0/20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Org&lt;sub&gt;0&lt;/sub&gt;</td>
<td>11001000 00010111 00010000 00000000</td>
<td>200.23.16.0/23</td>
</tr>
<tr>
<td>Org&lt;sub&gt;1&lt;/sub&gt;</td>
<td>11001000 00010111 00010010 00000000</td>
<td>200.23.18.0/23</td>
</tr>
<tr>
<td>Org&lt;sub&gt;2&lt;/sub&gt;</td>
<td>11001000 00010111 00010100 00000000</td>
<td>200.23.20.0/23</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Org&lt;sub&gt;7&lt;/sub&gt;</td>
<td>11001000 00010111 00011110 00000000</td>
<td>200.23.30.0/23</td>
</tr>
</tbody>
</table>

[Kurose–Ross p.336]
IP Address Assignment
Static Host IP Addresses

- Static IP address assignment
  - configuration file (e.g. Unix /etc/rc.config)

Advantages and disadvantages?
IP Address Assignment

Static Host IP Addresses

- Static IP address assignment
  - configuration file (e.g. Unix /etc/rc.config)

- Advantages
  - relative stable IP addresses better for incoming requests
  - allows systematic addresses for use by network admins

- Disadvantages
  - network administrators must manually assign addresses
  - users must manually configure their computers
IP Address Assignment

DHCP

- DHCP: dynamic host configuration protocol
  - [RFC 2131] – DHCPv6 [RFC 3315]
  - dynamic IP address assignment
  - automatic configuration of DNS servers and default routers
  - allows user to “plug into” network and it just works
    - most of the time
IP Address Assignment

DHCP Address Assignment

• DHCP address assignment procedure
  – client broadcasts discover message
    • UDP datagram to 255.255.255.255 port 67
  – DHCP server broadcasts offer message
  – client broadcasts request message
  – DHCP server ACKs

todo: details and figure
IP Address Assignment

Strict Hierarchy with CIDR

- Forwarding table entries unique to networks

issue

[Kurose–Ross p.347]
IP Address Assignment

Strict Hierarchy

- Forwarding table entries unique to networks
  - all organisations *must* change IP address with ISP change
IP Address Assignment

Loose Hierarchy

- Forwarding table entries not unique to networks
  - longest prefix is used for forwarding (most specific)

** org_0 200.23.16.0/23
** org_1 200.23.18.0/23
** org_2 200.23.20.0/23
** org_7 200.23.30.0/23
** org_9 199.31.0.0/23

ISP_A 200.23.30.0/20

ISP_B 199.31.0.0/16
  200.23.20.0/23

Tier1_x

| 199.31.0.0/16 | B |
| 200.23.30.0/20 | A |
| 200.23.20.0/23 | B |
IP Addressing
Network Address Translation

- **NAT**: network address translation  [RFC 2663, 3022]
  - translates public Internet address ↔ *private addresses*

*Benefits?*

```
+-----------------+   +-----------------+   +-----------------+
| org₁      |   | ISPₐ  |   | Tier₁ₓ  |
| IP: 200.23.18.0/23 |   | IP: 200.23.30.0/20 |   | 199.31.0.0/16 |
| NAT       |   |       |   | B        |
| IP: 172.16/12 |   | IP: 200.23.30.0/20 |   | 200.23.30.0/20 |
| NAT       |   |       |   | A        |
| IP: 172.16/12 |   |       |   |          |
```

- **NAT**: network address translation  [RFC 2663, 3022]
  - translates public Internet address ↔ *private addresses*
IP Addressing
Network Address Translation

• NAT: network address translation  [RFC 2663, 3022]
  – translates public Internet address ↔ private addresses

• Benefits
  – provide multiple private addresses for one public address
    • e.g. home networks without buying multiple IP addresses
  – change private addresses without impacting ISP assignment
  – change ISP without impacting private addresses
  – private addresses not explicitly addressable
    • helps resist attacks
IP NAT
Private Address Space

- IP addresses reserved for private Internets [RFC 1918]
  - reserved by IANA

why?
IP NAT
Private Address Space

• IP addresses reserved for private Internets [RFC 1918]
  – reserved by IANA
  – private addresses must not conflict with public addresses
    • would prevent routing toward public Internet if conflict

<table>
<thead>
<tr>
<th>CIDR Prefix</th>
<th>Class</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/8</td>
<td>A</td>
<td>10.000.0.0 – 10.255.255.255</td>
</tr>
<tr>
<td>172.16/12</td>
<td>B</td>
<td>172.016.0.0 – 172.031.255.255</td>
</tr>
<tr>
<td>192.168/16</td>
<td>C</td>
<td>192.168.0.0 – 192.168.255.255</td>
</tr>
</tbody>
</table>
IP NAT Implementation

- NAT Implementation

*how to translate multiple private addresses to one public Internet address?*
IP NAT
Implementation

• NAT Implementation
  – translate *multiple* private addr. to *one* public Internet addr.

• Hack using TCP ports
  – 16-bit port field allows almost 64K flows

• Outgoing translation
  – choose unused high port number new-port_q
  – \(<\text{private-addr}_i, \text{port}_p> \rightarrow <\text{public-addr}, \text{new-port}_q>\)
  – store in NAT translation table

• Incoming translation
  – \(<\text{public-addr}, \text{new-port}_q> \rightarrow <\text{private-addr}_i, \text{port}_p>\)

*Disadvantages?*
IP NAT Example

- Private 10.0.0/24 network NATed behind 138.76.27.9

[Kurose–Ross p.354]
IP NAT
Example

- Private 10.0.0/24 network NATed behind 138.76.27.9
  1: 10.0.0.1 generates TCP seg. to server 128.119.40.186:80

<table>
<thead>
<tr>
<th>Public</th>
<th>Private</th>
</tr>
</thead>
<tbody>
<tr>
<td>5001</td>
<td>10.0.0.2:3345</td>
</tr>
<tr>
<td>5007</td>
<td>10.0.0.1:21</td>
</tr>
<tr>
<td>5042</td>
<td>128.119.40.186</td>
</tr>
</tbody>
</table>
IP NAT Example

- Private 10.0.0/24 network NATed behind 138.76.27.9
  1: 10.0.0.1 generates TCP seg. to server 128.119.40.186:80
  2: NAT chooses unused port, translates, adds to table
IP NAT Example

- Private 10.0.0/24 network NATed behind 138.76.27.9
  1: 10.0.0.1 generates TCP seg. to server 128.119.40.186:80
  2: NAT chooses unused port, translates, adds to table
  3: normal interaction with server
IP NAT Example

- Private 10.0.0/24 network NATed behind 138.76.27.9
  1: 10.0.0.1 generates TCP seg. to server 128.119.40.186:80
  2: NAT chooses unused port, translates, adds to table
  4: normal interaction with server

ISP 128.119.40.186

<table>
<thead>
<tr>
<th></th>
<th>138.76.27.9</th>
<th>128.119.40.186</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>128.119.40.186:80</td>
<td>128.119.40.186</td>
</tr>
</tbody>
</table>

ISP NAT 138.76.27.9

<table>
<thead>
<tr>
<th>Public</th>
<th>Private</th>
</tr>
</thead>
<tbody>
<tr>
<td>5001</td>
<td>10.0.0.1:3345</td>
</tr>
<tr>
<td>5007</td>
<td>10.0.0.2:3345</td>
</tr>
<tr>
<td>5042</td>
<td>10.0.0.1:21</td>
</tr>
</tbody>
</table>
**IP NAT Example**

- Private 10.0.0/24 network NATed behind 138.76.27.9
  1: 10.0.0.1 generates TCP seg. to server 128.119.40.186:80
  2: NAT chooses unused port, translates, adds to table
  5: normal interaction with server

<table>
<thead>
<tr>
<th>Public</th>
<th>Private</th>
</tr>
</thead>
<tbody>
<tr>
<td>5001</td>
<td>10.0.0.1:3345</td>
</tr>
<tr>
<td>5007</td>
<td>10.0.0.2:3345</td>
</tr>
<tr>
<td>5042</td>
<td>10.0.0.1:21</td>
</tr>
</tbody>
</table>

ISP: 128.119.40.186

<table>
<thead>
<tr>
<th>Public</th>
<th>Private</th>
</tr>
</thead>
<tbody>
<tr>
<td>138.76.27.9</td>
<td>1</td>
</tr>
<tr>
<td>128.119.40.186</td>
<td>2</td>
</tr>
</tbody>
</table>
IP NAT Example

- Private 10.0.0/24 network NATed behind 138.76.27.9

1: 10.0.0.1 generates TCP seg. to server 128.119.40.186:80
2: NAT chooses unused port, translates, adds to table
3: normal interaction with server
6: NAT does reverse translation

<table>
<thead>
<tr>
<th>Public</th>
<th>Private</th>
</tr>
</thead>
<tbody>
<tr>
<td>5001</td>
<td>10.0.0.1:3345</td>
</tr>
<tr>
<td>5007</td>
<td>10.0.0.2:3345</td>
</tr>
<tr>
<td>5042</td>
<td>10.0.0.1:21</td>
</tr>
</tbody>
</table>
IP NAT
Disadvantages

- NAT disadvantages
  - eliminates end-to-end address transparency
    - NATed devices not addressable from outside
  - difficult to be a server or peer
    - ugly out-of-band hacks exist
  - violates protocol layer semantics
    - restricts to transport protocols using TCP/UDP socket semantic
IP NAT
Advantages

• NAT advantages
  – helped slow IP address exhaustion
  – provide security against inbound network attacks

• NATs were controversial
  – but have been widely deployed
  – and aren’t going to disappear
Internet Protocol
IPv4 Packet Format: Option Fields

- **Options**
  - optional packet processing
  - not typically in fast path
  - many service providers *ignore* options

- **Examples [IANA]**
  - security
  - source route
  - record route taken
  - router alert
  - timestamp

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IHL</td>
<td>Internet Header Length</td>
</tr>
<tr>
<td>TOS</td>
<td>Type of Service</td>
</tr>
<tr>
<td>total length</td>
<td></td>
</tr>
<tr>
<td>fragment id</td>
<td></td>
</tr>
<tr>
<td>flag</td>
<td></td>
</tr>
<tr>
<td>frag offset</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>Time to Live</td>
</tr>
<tr>
<td>protocol</td>
<td>Protocol Type</td>
</tr>
<tr>
<td>header checksum</td>
<td></td>
</tr>
<tr>
<td>source address</td>
<td></td>
</tr>
<tr>
<td>destination address</td>
<td></td>
</tr>
<tr>
<td>options</td>
<td>(<em>hl – 20B</em>)</td>
</tr>
<tr>
<td>payload</td>
<td>(<em>length – hl – 20B</em>)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
</tr>
<tr>
<td>IHL</td>
</tr>
<tr>
<td>TOS</td>
</tr>
<tr>
<td>total length</td>
</tr>
<tr>
<td>fragment id</td>
</tr>
<tr>
<td>flag</td>
</tr>
<tr>
<td>frag offset</td>
</tr>
<tr>
<td>TTL</td>
</tr>
<tr>
<td>protocol</td>
</tr>
<tr>
<td>header checksum</td>
</tr>
<tr>
<td>source address</td>
</tr>
<tr>
<td>destination address</td>
</tr>
<tr>
<td>options</td>
</tr>
<tr>
<td>payload</td>
</tr>
</tbody>
</table>
Internet Protocol
IPv4 Fragmentation: Performance

• Packet too big for (sub)network must be fragmented
  – generally dictated by link layer MTU (maximum transfer unit)
    • e.g. Ethernet MTU = 1500B

• Significant performance penalty
  – delay in fragmentation and reassembly
  – buffer space to hold partially fragmented/reassembled

• Most service providers disable fragmentation

*Alternative*?
IPv4 Fragmentation: Path MTU Discovery

  - determine MTU of entire path
  - transport protocol uses to limit packet size
- Done automatically
  - sender sets DF flag (don’t fragment)
  - sender uses local link MTU or min(local-link-MTU,576)
- IP router at each hop
  - forwards if no fragmentation needed
  - discards and returns ICMP (3,4) message
- Sender retries with smaller MTU
Internet Control

ICMP

- Control messages for the Internet
  - carried in IP datagrams
  - not a well organised control plane
    - unlike the PSTN

- ICMP message
  - type, code plus first 8 bytes of IP datagram causing error
  - http://www.iana.org/assignments/icmp-parameters
  - http://www.iana.org/assignments/icmpv6-parameters

- Not a well-organised control plane (unlike the PSTN)
  - rather a set of messages
  - used by other protocols and tools
## Internet Control

### ICMP Selected Message Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>echo reply (used by ping)</td>
</tr>
<tr>
<td>3</td>
<td>x</td>
<td>destination unreachable</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>source quench (intended for congestion control)</td>
</tr>
<tr>
<td>5</td>
<td>x</td>
<td>redirect</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>echo (used by ping)</td>
</tr>
<tr>
<td>9</td>
<td>x</td>
<td>router advertisement [RFC 1256]</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>router solicitation</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>TTL exceeded</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>timestamp</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>timestamp reply</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>information request</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>information reply</td>
</tr>
</tbody>
</table>
# Internet Control

## ICMP Selected Unreachable Codes

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>network unreachable</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>host unreachable</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>protocol unreachable</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>port unreachable</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>fragmentation needed and don’t fragment was set</td>
</tr>
</tbody>
</table>
Internet Control

Essential Tool: ping

• Ping used to test liveness of remote host
  – sends ICMP (8,0) echo message
  – destination replies with ICMP (0,0) echo reply message

• Client usage (CLI)
  – ping ⟨dnsname⟩
    • DNS first resolves hostname to IP address
  – ping ⟨address⟩

  try it
  ping www.eecs.ku.edu
  ping www.comp.lancs.ac.uk
Internet Control
Essential Tool: traceroute

- Traceroute used to map path HBH through network
  - sends UDP datagrams with increasing TTL and unlikely port
  - each hop drops and returns ICMP (11,0) TTL exceeded
    - many network service provider do not reply
      traceroute times out and shows “*” for these hops
  - done 3 times per hop; source computes and averages RTT
  - destination replies with ICMP (3,3) port unreachable

- Client usage (CLI)
  - traceroute ⟨dnsname⟩
    - DNS first resolves hostname to IP address
  - traceroute ⟨address⟩
    - try it traceroute www.eecs.ku.edu
      traceroute www.comp.lancs.ac.uk
Internet Protocol

IPv6 Motivation

• IPv4 address space exhaustion
  – CIDR reduced problem with more efficient allocation
  – NATs reuse addresses in edge networks
    • perhaps eliminating the address-motivation for IPv6 completely
    • IoT may significantly accelerate IP address demand

• IPv4 didn’t provide explicit support for QoS
  – other than TOS field

• IPv4 was not designed for efficient processing
  – options
  – variable length header

• IPv6 chosen among a number of proposals
Internet Protocol

IPv6 Overview

- IPv6 (Internet Protocol version 6)
  - [RFC 8200] standards track, but not yet standard
- ICMPv6 (Internet control message protocol vers. 6)
  - signalling for IPv6 [RFC 4443]
- Support for QoS with flow label
- More efficient structure for high-speed processing
  - fixed 40B header
  - no options
    - but next header can contain options rather than L4 header
  - fragmentation not allowed
### Internet Protocol: IPv6

**Packet Format: Header Control Fields**

- **IP version number** = 06
- **Traffic class** [8b]
- **Flow label** [20b]
  - flow identifier for soft state
  
  *Lecture TQ*

- **Next header**
  - protocol # to demux
  - extension header

- **Hop limit**
  - used as IPv4 TTL

- **note:** *no header checksum*

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>06</strong></td>
<td>1</td>
</tr>
<tr>
<td><strong>class</strong></td>
<td>3</td>
</tr>
<tr>
<td><strong>flow label</strong></td>
<td>20</td>
</tr>
<tr>
<td><strong>payload length</strong></td>
<td>32</td>
</tr>
<tr>
<td><strong>next hdr</strong></td>
<td>1</td>
</tr>
<tr>
<td><strong>hop limit</strong></td>
<td>1</td>
</tr>
</tbody>
</table>

- **source address**
- **destination address**
- **payload**
  - (= payload length)
Internet Protocol: IPv6
Packet Format: Payload Related Fields

- IP version number = 06
- Payload length [B]
- Payload
Internet Protocol: IPv6
Packet Format: Addresses

• 128 bit IP addresses
  \[ 2^{128} = 3 \times 10^{38} = 7 \times 10^{23} \text{ addr/m}^2 \text{ on earth} \]
  \[ \approx 6.022 \times 10^{23} \text{ (Avagadro)} \]
• Structure in allocation
  – reduces usable number
  – IANA administers

<table>
<thead>
<tr>
<th>06</th>
<th>class</th>
<th>flow label</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>payload length</td>
<td>next hdrl</td>
</tr>
</tbody>
</table>

- source address
- destination address
- payload
  (= payload length)
Internet Protocol
IPv4 → IPv6 Transition

*If IPv6 replaces IPv4, how to transition?*
Internet Protocol
IPv4 → IPv6 Transition

- If IPv6 replaces IPv4, how to transition?  [RFC 4213]
  - “flag day”: all systems change to IPv6 at same time

problem?
Internet Protocol
IPv4 → IPv6 Transition

- If IPv6 replaces IPv4, how to transition?
  - “flag day”: all systems change to IPv6 at same time
  - too disruptive
    - not possible to coordinate and synchronise change
    - many running systems get no regular attention
  - won’t happen
Internet Protocol
IPv4 → IPv6 Transition

- If IPv6 replaces IPv4, how to transition?
  - “flag day”: won’t happen
  - tunneling: IPv6 encapsulated in IPv4 packets [RFC 7059]
    *how?*
**Internet Protocol**

IPv4 → IPv6 Transition

- If IPv6 replaces IPv4, how to transition?
  - "flag day": won’t happen
  - tunneling: IPv6 encapsulated in IPv4 packets [RFC 7059]
- 6in4: IPv6 is protocol ID 41 for IPv4
Internet Protocol

IPv4 → IPv6 Transition

- If IPv6 replaces IPv4, how to transition?
  - "flag day": won’t happen
  - tunneling: IPv6 encapsulated in IPv4 packets [RFC 7059]
  - dual stack
    - most IP routers now support both IPv4 and IPv6
    - permits slow transition with dual addressing
Internet Protocol
IPv4 → IPv6 Transition

- If IPv6 replaces IPv4, how to transition?
  - “flag day”: won’t happen
  - tunneling: IPv6 encapsulated in IPv4 packets [RFC 7059]
  - dual stack
- Many deployment alternative details
Network Layer

NL.5 Fast Datagram Routers

NL.1 Network layer functions and services
NL.2 Network signalling paradigms
NL.3 Switches and packet structure
NL.4 Examples
NL.5 Router architecture
NL.6 Programmable networks and OpenFlow
Fast Datagram Routers

Motivation

• Connection-oriented fast packet switching
  – emerged in ATM standards, but ATM failed
• IP became waist of global network infrastructure
  – increased processing capability enabled fast IP lookups
  – apply fast packet switching to IP datagram forwarding
Fast Datagram Switches

Architecture

- Fast packet switch core
- Input processing
  - IP lookup
  - Packet classification
- Output processing
  - Packet scheduling
    - Fair queueing

Diagram:
- Input processing
  - Prefixes
  - Input processor
  - Header update
  - Classify
- Output processing
  - Prefixes
  - Input processor
  - Output scheduling
  - Link
- Switch fabric control
- Management routing and signalling
Fast Datagram Switches

Throughput

- Packet processing rate critical [packets/s]
  - packet processing must sustain at least average rate
  - critical path must sustain peak line rate for min size packets
Fast Datagram Switches
Software IP Lookup

- Longest prefix match
- Critical parameters
  - worst case lookup time
    - brute force: $O(\log_2 n)$
    - $n$ hundred thousands
  - memory required
  - forwarding table update time
Fast Datagram Switches
Software IP Lookup Example: Trie

- Many algorithms
- Example: trie
  - sparse binary tree
  - valid prefixes are root
  - lookup time $O(a)$
    - $a$ = number of address bits
Fast Datagram Switches

Hardware IP Lookup

- Ternary CAM
  - 1, 0, X (don’t care)
  - expensive and complex
  - relative to RAM
- Simultaneous match
  - lookup time constant
  - $O(1)$

<table>
<thead>
<tr>
<th>prefix</th>
<th>$p_{out}$</th>
<th>$f_{state}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00XXXX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001XXX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0001XX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0101XX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>101XX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10100X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11XXXX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>111XXX</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

payload 101 011 01 $p_{out}$ 101 011 01

- hop count
- checksum fix
- priority mux
Network Layer

NL.6 Programmable Networks

NL.1 Network layer functions and services
NL.2 Network signalling paradigms
NL.3 Switches and packet structure
NL.4 Examples
NL.5 Router architecture
NL.6 Programmable networks and OpenFlow
Programmable Networking

Motivation

• Programmable networking motivation
  – service providers rapidly deploy and manage new services
  – without waiting and dependencies on router design cycle

• Significant benefit to research community
  – experiment on new router protocols and mechanisms
  – on high-performance device
    • software routers limited in capabilities
Programmable Networking

History

• Programmable networking is not a new paradigm
  – SDN (software defined networking) is the current buzzword

• Active Networks
  – 1990s large DARPA and EU-funded initiatives
  – mobile code in capsules
  – can inject new services into the network
    • control and data plane programmability

• IN (Intellignet networks) ITI-T Q.1200
  – 1990s enhancements to SS7-based PSTN
  – provides programmability to call state machines

• Open commodity network processors (Intel and IBM)
Programmable Networking
Requirements

- Programmable network elements
  - with *open interfaces*
  - network processors enabled this ~2000
    - Intel i1200/i2XXX and IBM NP4GS3
    - sadly Cisco developed their own proprietary NP
    - open commodity NP market is not dead

- Data and control plane separation
  - with *open interface*
  - IETF ForCES (early 2000s) [RFC 3654]
    - *Forwarding and Control Element Separation*
Programmable Networking
SDN and OpenFlow

- **SDN**: software-defined networking
  - current programmable networking paradigm
  - control plane separation and programmability

  - first and most widely deployed SDN instantiation
    - *southbound* interface
  - ONF (Open Networking Foundation)
    - www.opennetworking.org
    - non-profit industry-funded consortium
    - creates and maintains OpenFlow standards
SDN
Service Model

• **Components:**
  – SDN controller
    • e.g. OpenDaylight (ODL)
  – SDN-enabled L2 switch or IP router
    • ≥ 1 per controller
    • e.g. OpenFlow router

• **Interfaces:**
  – **northbound**: controller ↔ services
    • e.g. ODL API
  – **southbound**: controller ↔ switch
    • e.g. OpenFlow
OpenFlow
Network Architecture

- OF-enabled switches
  - contains *flow table*
- OFC: OpenFlow controller
  - logically central control plane
  - significant issues in scaling
    - multiple controllers
OpenFlow
Data Plane Abstraction

- Flow table is computed and installed from OFC
- Flow: defined by header fields
  - corresponds to conventional notion of related packets in flow
- Generalized forwarding: simple packet-handling rules
  - pattern: match values in packet header fields
  - actions: for matched packet, e.g.
    - drop, forward, or modify
    - send matched packet to controller
  - priority: disambiguate conflicting patterns
  - counters: #bytes and #packets
OpenFlow
Flow Tables

• Rule
  – switch port; VLAN ID; MAC src, dest; Eth type,
    IP src, dest; prot; src, dest port

• Action

• Statistics
Some material in these foils comes from the textbook supplementary materials:

  http://wps.aw.com/aw_kurose_network_4

- Sterbenz & Touch, *High-Speed Networking: A Systematic Approach to High-Bandwidth Low-Latency Communication*
  http://hsn-book.sterbenz.org