Chair for Network Architectures and Services – Prof. Carle
Department for Computer Science
TU München

Master Course Computer Networks IN2097

Prof. Dr.-Ing. Georg Carle
Christian Grothoff, Ph.D.

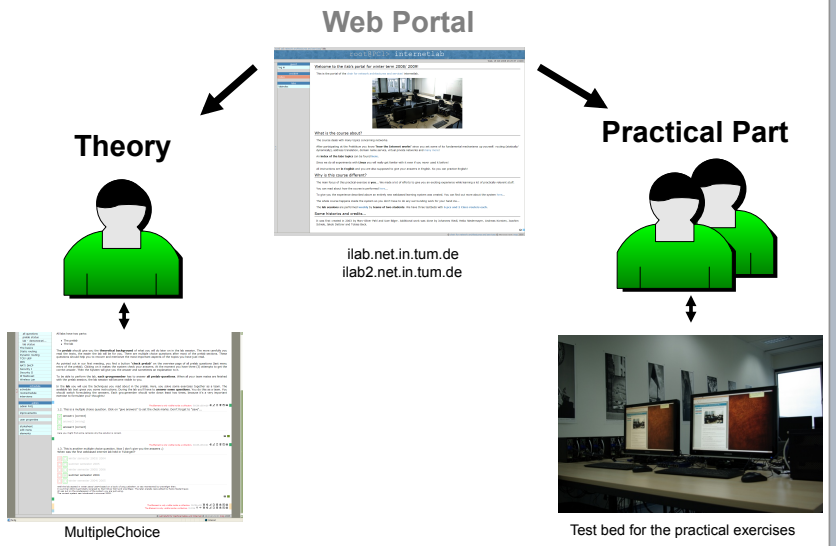
Chair for Network Architectures and Services
Institut für Informatik
Technische Universität München
http://www.net.in.tum.de

Announcement for Internet Laboratory 1 + 2

- c.f. http://www.net.in.tum.de/en/teaching/ss12/praktika/
- ilab 1
  - Andreas Müller, Holger Kinkelin
  - Information Event: Thu, 02.02.2012, 4pm, room 03.07.023
  - Kick-off with team assignment: Thursday, 19.04.2012, 16:00
    Weekly lecture: Thursdays 16:00 - 17:00, room 03.07.023
  - http://ilab.net.in.tum.de/
- ilab 2
  - Marc-Oliver Pahl, Stephan Günther
  - Information Event: Thu, 02.02.2012, 10:15, room 03.07.023
  - Kick-off (with team assignment): Thursday 19.04.2012, 10:00
    Weekly lecture: every Thursday, 10h00, Room 03.07.023
    The exercise starts with the kick-off meeting.
  - http://ilab2.net.in.tum.de/

Room 03.05.014 for the Practical Exercises

It is all web Based



ilab 1 Topics

- The **Basics**
  - Cabling and getting familiar with basic tools…
- **Static Routing**
  - Exploring static routing…
- **Dynamic Routing**
  - Exploring dynamic routing with RIP, OSPF, basic BGP
- **TCP/ UDP**
  - Watching packets travel on the transport layer (congestion control, etc.)…
- **DNS**
  - Setting up a DNS server and exploring some details of the DNS…
- **NAT/ DHCP/ IPv6**
  - Specialties behind a NAT, auto configuration, what is "new" with IPv6…
- **Security I** (**Firewall, SSL**)
  - Setting up an HTTP server with SSL and configuring an iptables firewall…
- **Security II** (**VPN**)
  - Virtual private networks with IPSec…
- **Wireless LAN** (WEP, WPA, Radius)
  - Exploring the mechanisms of wireless LANs with a focus on security…
- **Multicast**
  - Sending data from one source to multiple receivers…

ilab2.net.in.tum.de

- Modules
  - IN8903: Praktikum im Bereich Technische Informatik (10 ECTS)
  - IN2106: Master-Praktikum (10 ECTS)
- Course is in **English**
- Expected audience: Students with
  - **interest and background in computer networks**,
  - **familiar** with the **Linux Operating system at Shell level**.
- Required: you already know the basics
  - Practical experience with the ilab1 topics: Static/ dynamic Routing, TCP/ UDP, DNS, NAT, DHCP, IPv6, Firewall, TLS, VPN, Wireless LAN

ilab 2 Topics

{static, dynamic} routing
{pahl}@net.in.tum.de

BGP
{hirvi}@net.in.tum.de

Advanced NAT
{mueller} @net.in.tum.de

IPv6
{pahl}@net.in.tum.de

Stream Control Transmission Protocol
{braun, mueller}@net.in.tum.de

Multicast | SIP
{bandh}@net.in.tum.de

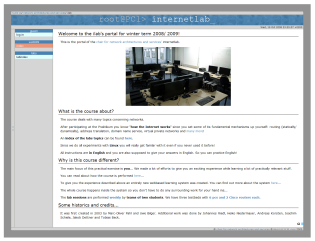
P2P-Filesharing
{haage, fessi}@net.in.tum.de

Measurement
{muenz, braun}@net.in.tum.de

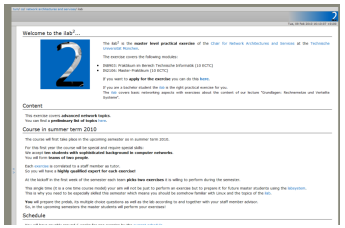
Security and Attacks
{holz}@net.in.tum.de

Services in the web
{schlamp, haage}@net.in.tum.de

More Information



http://ilab.net.in.tum.de



http://ilab2.net.in.tum.de

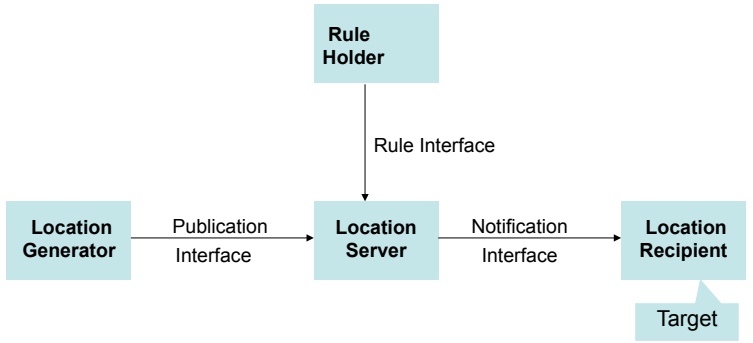
- Registration: ilab2.net.in.tum.de?address=register
- Course contents: ilab2.net.in.tum.de?address=accessableLabs
- As the practical exercises are performed in teams of two students, it is a good idea to search a team partner early on. You can give the name of your designated team partner in the registration/application form.

Chair for Network Architectures and Services – Prof. Carle
Department for Computer Science
TU München

Location Information and IETF GeoPriv Working Group

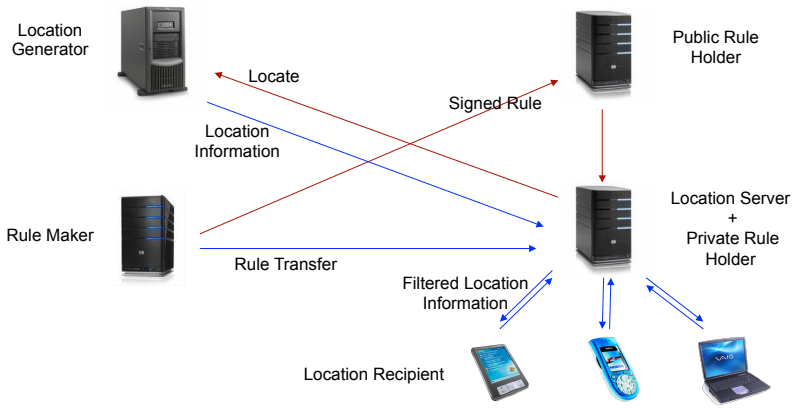
credits:
Milind Nimesh, Columbia University

Geopriv Entities



Location Recipients may request that a Location Server provide them with GEOPRIV location information concerning a particular Target.

Scenarios



Mobile Communities and Location-Based Services

Location Configuration

Configuring the location of a device, using means such as:

- DHCP extensions
  - RFC3825 : Option 123, geo-coordinate based location
  - RFC4776 : Option 99, civic address
- Link Layer Discovery Protocol - Media Endpoint Discovery
  - LLDP - a vendor-neutral Layer 2 protocol that allows a network device to advertise its identity and capabilities on the local network.
    IEEE standard 802.1AB-2005 in May 2005.
    Supersedes proprietary protocols like Cisco Discovery Protocol,
  - auto-discovery of LAN information (system id, port id, VLAN id, DiffServ settings, …) ⇨ plug & play
  - cisco discovery protocol: switch broadcasts switch/port id
    - switch → floor, port → room ⇨ room level accuracy
- HTTP Enabled Location Delivery
  - device retrieves location from Location Information Server (LIS)
  - assumption: device & LIS present in same admin domain;
    find LIS by DHCP, IPv6 anycast, …
- Applications ⇨ emergency 911, VoiP, location based applications

PIDF Elements

Baseline: RFC 3863

- entity
- contact (how to contact the person)
- timestamp
- status
- tuple (provide a way of segmenting presence information)

Extensions: RFC 4119

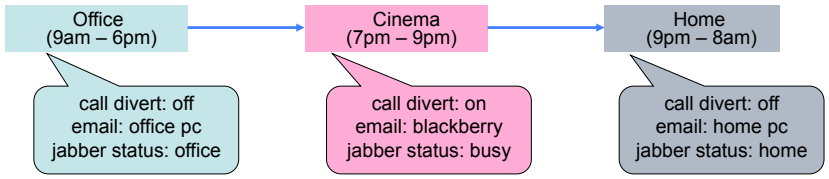
- location-info
- usage-rules
  - retransmission-allowed
  - retention-expires
  - ruleset-reference
  - note-well
- method
- provided-by

PIDF-LO Example

- PIDF-LO: RFC 4119 (RFC 5139, RFC 5491)
- c.f. http://www.voip-sos.net/tools/pidflo/

```
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
    xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
    entity="pres:sample@example.com">
<tuple id="0815">
<status>
 <gp:geopriv>
  <gp:location-info><!--  location information is inserted here --></gp:location-info>
  <gp:usage-rules>
    <gp:retransmission-allowed>no</gp:retransmission-allowed>
    <gp:retention-expiry>2010-08-10T09:00:10+02:00</gp:retention-expiry>
  </gp:usage-rules>
 </gp:geopriv>
</status>
<timestamp>2010-08-10T08:31:00+02:00</timestamp>
</tuple>
</presence>
```


Location Type Registry



- Describes places of humans or end systems
- Application
  - define location-based actions
  - e.g. if loc = “classroom” then cell phone ringer = off
  - e.g. if loc = “cinema” then call divert = on
- Location coordinate knowledge ≠ context
- airport, arena, bank, bar, bus-station, club, hospital, library….
- ⇨ Prediction:
  most communication will be presence-initiated or pre-scheduled

GeoPriv RFCs

- RFC 3693: Geopriv Requirements, 2004 (Informational), Updated by RFC 6280
- RFC 3694: Threat Analysis of the Geopriv Protocol, 2004 (Informational), Updated by RFC 6280
- RFC 3825: Dynamic Host Configuration Protocol Option for Coordinate-based Location Configuration Information, 2004 (Proposed Standard), Obsoleted by RFC 6225
- RFC 4079: A Presence Architecture for the Distribution of GEOPRIV Location Objects, 2005 (Informational)
- RFC 4119: A Presence-based GEOPRIV Location Object Format, 2005 (Proposed Standard), Updated by RFC 5139, RFC 5491
- RFC 4589: Location Types Registry, 2006 (Proposed Standard)
- RFC 4676: Dynamic Host Configuration Protocol (DHCPv4 and DHCPv6) Option for Civic Addresses Configuration Information, 2006 (Proposed Standard), Obsoleted by RFC 4776
- RFC 4745, Common Policy: A Document Format for Expressing Privacy Preferences, 2007 (Proposed Standard)
- RFC 4776: Dynamic Host Configuration Protocol (DHCPv4 and DHCPv6) Option for Civic Addresses Configuration Information, 2006 (Proposed Standard), Updated by RFC 5774

GeoPriv RFCs

- RFC 5139: Revised Civic Location Format for Presence Information Data Format Location Object (PIDF-LO), 2008 (Proposed Standard)
- RFC 5491: GEOPRIV Presence Information Data Format Location Object (PIDF-LO) Usage Clarification, Considerations, and Recommendations 2009 (Proposed Standard)
- RFC 5580: Carrying Location Objects in RADIUS and Diameter, 2009 (Proposed Standard)
- RFC 5606: Implications of 'retransmission-allowed' for SIP Location Conveyance, 2009 (Informational)
- RFC 5687: GEOPRIV Layer 7 Location Configuration Protocol: Problem Statement and Requirements, 2010 (Informational)
- RFC 5774: Considerations for Civic Addresses in the Presence Information Data Format Location Object (PIDF-LO): Guidelines and IANA Registry Definition, 2010 (Best Current Practice)
- RFC 5808: Requirements for a Location-by-Reference Mechanism, 2010 (Informational)

GeoPriv RFCs

- RFC 5870: A Uniform Resource Identifier for Geographic Locations ('geo' URI), 2010 (Proposed Standard)
- RFC 5985: HTTP-Enabled Location Delivery (HELD), 2010 (Proposed Standard)
- RFC 5986: Discovering the Local Location Information Server (LIS), 2010 (Proposed Standard)
- RFC 6155: Use of Device Identity in HTTP-Enabled Location Delivery (HELD), 2011 (Proposed Standard)
- RFC 6225: Dynamic Host Configuration Protocol Options for Coordinate-Based Location Configuration Information, 2011 (Proposed Standard)
- RFC 6280: An Architecture for Location and Location Privacy in Internet Applications, 2011 (Best Current Practice)

GeoPriv Tools

c.f. http://trac.tools.ietf.org/wg/geopriv/trac/wiki/GeoprivTools

- Open Source LIS: A PHP-based HELD server with a Java-based client, http://held-location.sourceforge.net/
- The Internet Geolocation Toolkit: A multi-platform, multi-protocol C++ library for geolocation access, http://igtk.sourceforge.net/
- ECRITdroid: An emergency calling client for Android. Doesn't do GEOPRIV now (just LoST/ECRIT), but should soon, in order to be fully ECRIT-compliant, http://ecritdroid.googlecode.com/
- Online DHCP encoders: An AJAX tool for encoding location values for use in the DHCP location options; http://geopriv.dreamhosters.com/dhcloc/
- Firefox implementation of W3C Geolocation API: supports a limited profile of HELD. To enable: Go to "about:config"; set "geo.wifi.protocol" to "1"; set "geo.wifi.uri" to URL of HELD server, https://bugzilla.mozilla.org/show_bug.cgi?id=545001
- CommScope LIS: commercial LIS, http://www.commscope.com

Chair for Network Architectures and Services – Prof. Carle
Department for Computer Science
TU München

Maintaining network state

Design Principles

Goals:

- identify, study common architectural components, protocol mechanisms
- what approaches do we find in network architectures?
- *synthesis:* big picture

7 design principles:

- network virtualization: overlays
- separation of data, control ⇨signalling
- **hard state versus soft state**
- randomization
- indirection
- multiplexing
- design for scale

Maintaining network state

state: information *stored* in network nodes by network protocols

- updated when network "conditions" change
- stored in multiple nodes
- often associated with end-system generated call or session
- examples:
  - ATM switches maintain lists of VCs: bandwidth allocations, VCI/VPI input-output mappings
  - RSVP routers maintain lists of upstream sender IDs, downstream receiver reservations
  - TCP: Sequence numbers, timer values, RTT estimates

Hard-state

- state *installed* by receiver on receipt of *setup message* from sender
- state *removed* by receiver on receipt of *teardown message* from sender
- *default assumption:* state valid unless told otherwise
  - in practice: failsafe-mechanisms (to remove orphaned state) in case of sender failure e.g., receiver-to-sender "heartbeat": is this state still valid?
- examples:
  - Q.2931 (ATM Signaling)
  - ST-II (Internet hard-state signaling protocol - outdated)
  - TCP

Soft-state

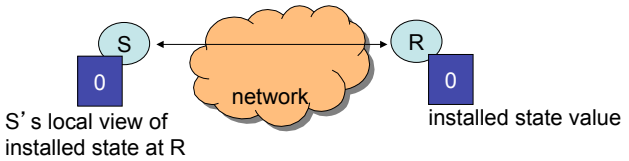
- state *installed* by receiver on receipt of setup (trigger) message from sender (typically, an endpoint)
  - sender also sends periodic *refresh* message: indicating receiver should continue to maintain state
- state *removed* by receiver via timeout, in absence of refresh message from sender
- default assumption: state becomes invalid unless refreshed
  - in practice: explicit state removal (*teardown*) messages also used
- examples:
  - RSVP, RTP/RTCP, IGMP

State: senders, receivers

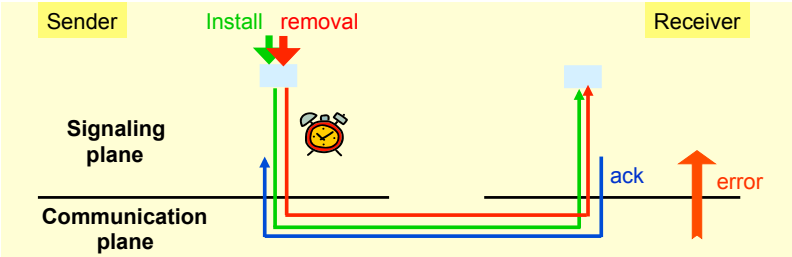
- sender: network node that *(re)generates* signaling (control) messages to install, keep-alive, remove state from other nodes
- receiver: node that creates, maintains, removes state based on signaling messages *received* from sender

Let's build a signaling protocol

- *S:* state *S*ender (state installer)
- *R:* state *R*eceiver (state holder)
- desired functionality:
  - S: set values in R to 1 when state "installed", set to 0 when state "not installed"
  - if other side is down, state is not installed (0)
  - initial condition: state not installed

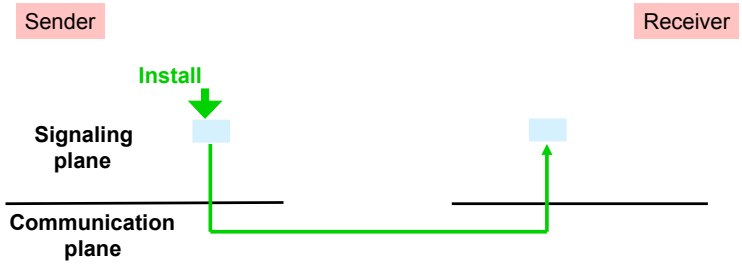


Hard-state signaling



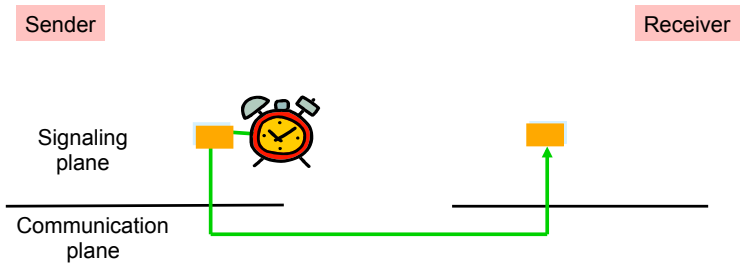
- reliable signaling
- state removal by request
- requires additional error handling
  - e.g., sender failure

Soft-state signaling



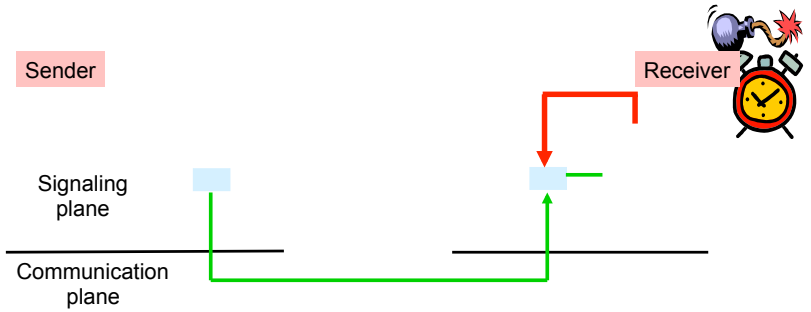
- best effort signaling

Soft-state signaling



- best effort signaling
- refresh timer, periodic refresh

Soft-state signaling



- best effort signaling
- refresh timer, periodic refresh
- state time-out timer, state removal only by time-out

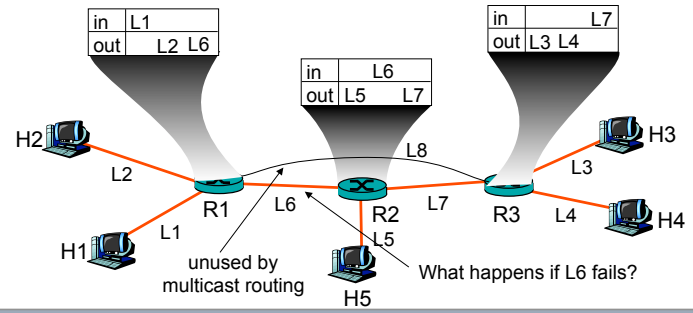
Soft-state: claims

- "Systems built on soft-state are robust" [Raman 99]
- "Soft-state protocols provide .. greater robustness to changes in the underlying network conditions…" [Sharma 97]
- "obviates the need for complex error handling software" [Balakrishnan 99]

What does this mean?

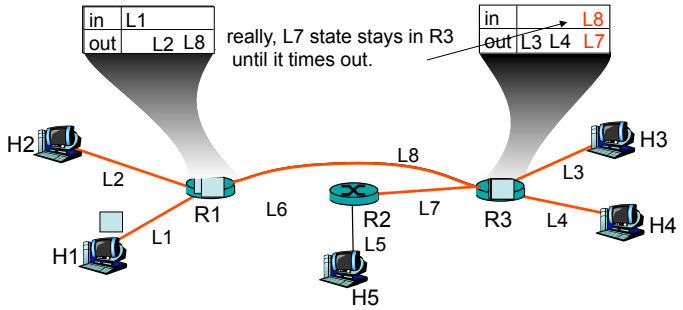
Soft-state: "easy" handling of changes

- Periodic refresh: if network "conditions" change, refresh will re-establish state under new conditions
- example: RSVP/routing interaction: if routes change (nodes fail) RSVP PATH refresh will *re-establish* state along new path



Soft-state: “easy” handling of changes

- L6 goes down, multicast routing reconfigures but…
- H1 data no longer reaches H3, H4, H5 (no sender or receiver state for L8)
- H1 refreshes PATH, establishes *new* state for L8 in R1, R3
- H4 refreshes RESV, propagates upstream to H1, establishes new receiver state for H4 in R1, R3



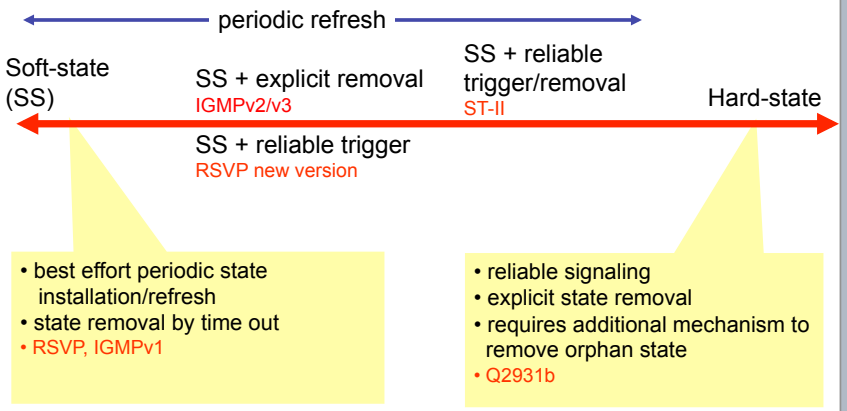
Soft-state: “easy” handling of changes

- “recovery” performed transparently to end-system by normal refresh procedures
- no need for network to signal failure/change to end system, or end system to respond to specific error
- less signaling (volume, types of messages) than hard-state from network to end-system but…
- more signaling (volume) than hard-state from end-system to network for refreshes

Soft-state: refreshes

- refresh messages serve many purposes:
  - trigger: first time state-installation
  - refresh: refresh state known to exist (“I am still here”)
  - <lack of refresh>: remove state (“I am gone”)
- challenge: all refresh messages unreliable
  - problem: what happens if first PATH message gets lost?
    - copy of PATH message only sent after refresh interval
  - would like triggers to result in state-installation a.s.a.p.
  - enhancement: add receiver-to-sender refresh_ACK for triggers
  - sender initiates retransmission if no refresh_ACK is received after short timeout
  - e.g., see paper “Staged Refresh Timers for RSVP” by Ping Pan and Henning Schulzrinne
  - approach also applicable to other soft-state protocols

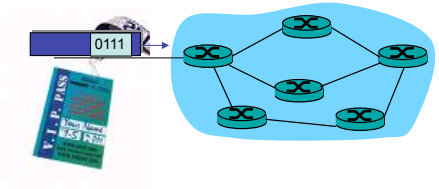
Signaling Spectrum



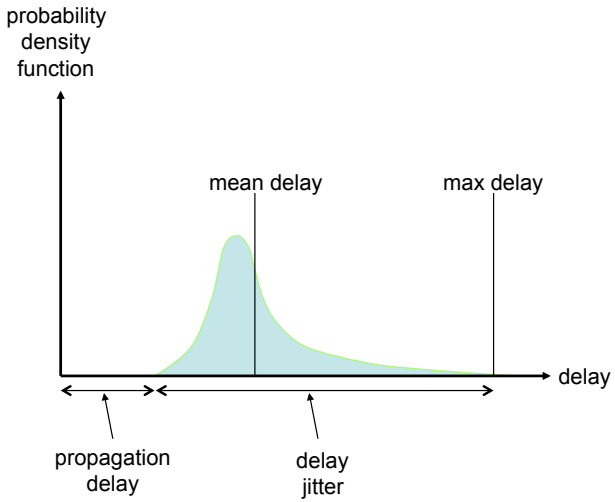
Quality of Service Support

Providing Multiple Classes of Service

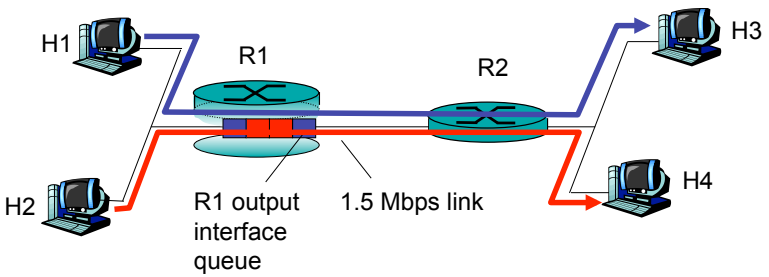
- Traditional Internet approach: making the best of best effort service
  - one-size fits all service model
- Alternative approach: multiple classes of service
  - partition traffic into classes
  - network treats different classes of traffic differently (analogy: VIP service vs regular service)
- granularity:
  differential service among multiple classes, not among individual connections
- history:
  ToS bits in IP header



Delay Distributions

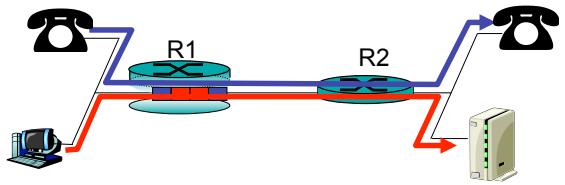


Multiple classes of service: scenario



Scenario 1: mixed FTP and audio

- Example: 1Mbps IP phone, FTP or NFS share 1.5 Mbps link.
  - bursts of FTP or NFS can congest router, cause audio loss
  - want to give priority to audio over FTP

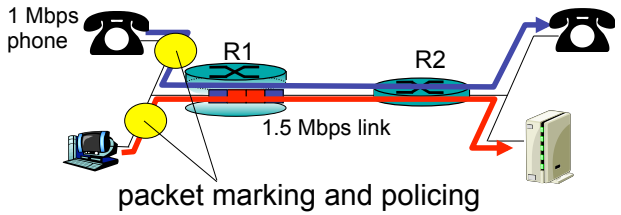


Principle 1

packet marking needed for router to distinguish between different classes; and new router policy to treat packets accordingly

Principles for QOS Guarantees (more)

- what if applications misbehave (audio sends higher than declared rate)
  - policing: force source adherence to bandwidth allocations
- marking and policing at network edge:
  - similar to ATM UNI (User Network Interface)

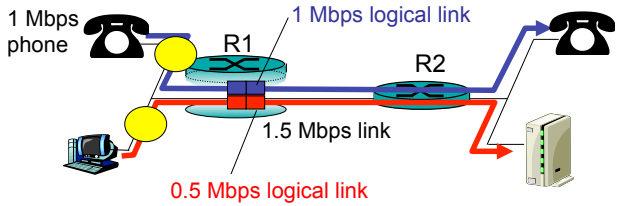


Principle 2

provide protection (*isolation*) for one class from others

Principles for QOS Guarantees (more)

- Allocating *fixed* (non-sharable) bandwidth to flow: *inefficient* use of bandwidth if flows doesn't use its allocation

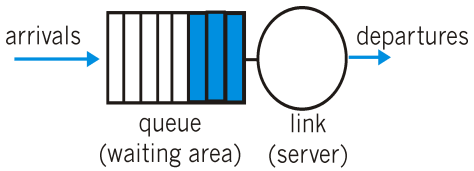


Principle 3

While providing **isolation**, it is desirable to use resources as efficiently as possible

Scheduling And Policing Mechanisms

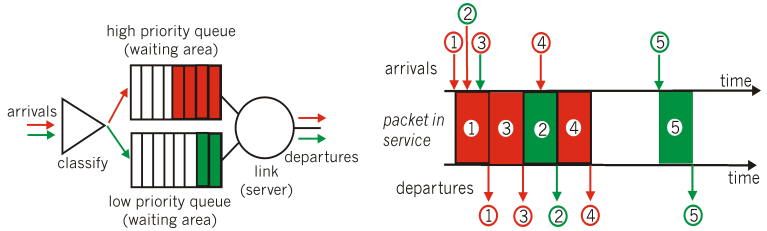
- scheduling: choose next packet to send on link
- FIFO (first in first out) scheduling: send in order of arrival to queue
  - ⇨real-world example?
  - discard policy: if packet arrives to full queue: who to discard?
    - Tail drop: drop arriving packet
    - priority: drop/remove on priority basis
    - random: drop/remove randomly



Scheduling Policies: more

Priority scheduling: transmit highest priority queued packet

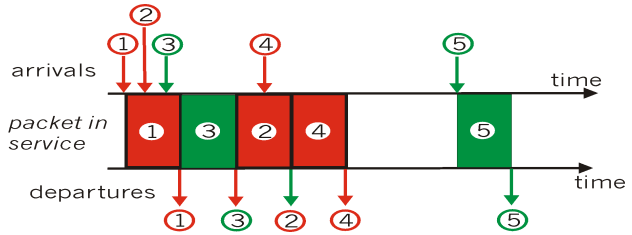
- multiple *classes*, with different priorities
  - class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc..



Scheduling Policies: still more

round robin scheduling:

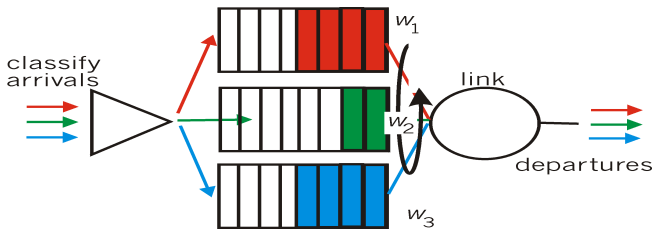
- multiple classes
- cyclically scan class queues, serving one from each class (if available)



Scheduling Policies: still more

Weighted Fair Queuing:

- generalized Round Robin
- each class gets weighted amount of service in each cycle
- when all classes have queued packets, class i will receive a bandwidht ratio of $w_i/\Sigma w_j$



Policing Mechanisms

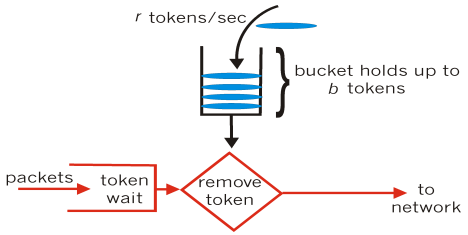
Goal: limit traffic to not exceed declared parameters

Three common-used criteria:

- *(Long term) Average Rate:* how many packets can be sent per unit time (in the long run)
  - crucial question: what is the interval length:
    100 packets per sec
    or 6000 packets per min have same average!
- *Peak Rate:* e.g., 6000 packets per min. (ppm) avg.;
  1500 pps peak rate
- *(Max.) Burst Size:* max. number of packets sent consecutively

Policing Mechanisms

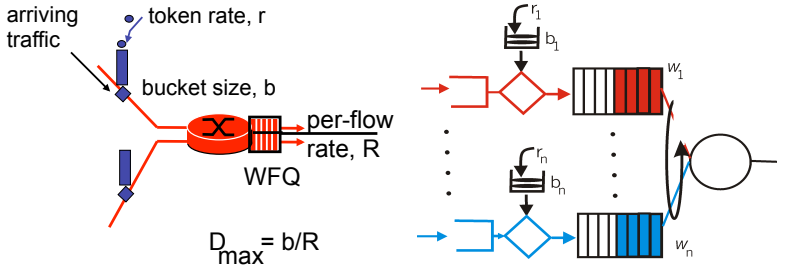
Token Bucket: limit input to specified Burst Size and Average Rate.



- bucket can hold b tokens ⇒ limits maximum burst size
- tokens generated at rate *r token/sec* unless bucket full
- *over interval of length t: number of packets admitted less than or equal to (r t + b).*

Policing Mechanisms (more)

- token bucket, WFQ combined provide guaranteed upper bound on delay, i.e., *QoS guarantee*



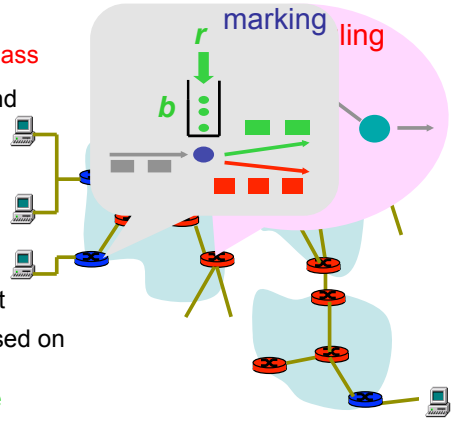
IETF Differentiated Services

- want "qualitative" service classes
  - "behaves like a wire"
  - relative service distinction: Platinum, Gold, Silver
- *scalability:* simple functions in network core, relatively complex functions at edge routers (or hosts)
  - in contrast to IETF Integrated Services: signaling, maintaining per-flow router state difficult with large number of flows
- don't define define service classes, provide functional components to build service classes

Diffserv Architecture

Edge router:

- per-flow traffic management
- marks packets according to class
- marks packets as in-profile and out-profile

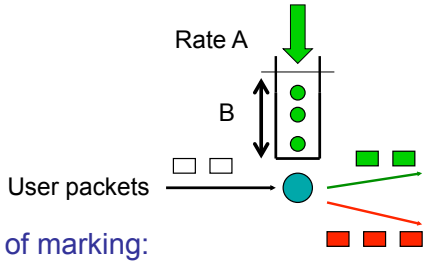


Core router:

- per class traffic management
- buffering and scheduling based on marking at edge
- preference given to in-profile packets

Edge-router Packet Marking

- profile: pre-negotiated rate A, bucket size B
- packet marking at edge based on per-flow profile



Possible usage of marking:

- class-based marking: packets of different classes marked differently
- intra-class marking: conforming portion of flow marked differently than non-conforming one

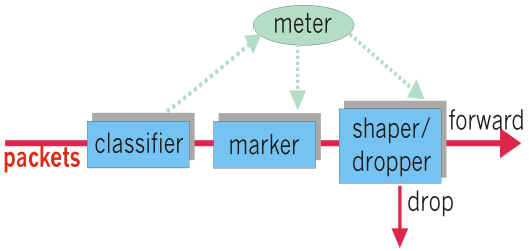
Classification and Conditioning

- Packet is marked in the Type of Service (TOS) in IPv4, and Traffic Class in IPv6
- 6 bits used for Differentiated Service Code Point (DSCP) and determine PHB that the packet will receive
- 2 bits can be used for congestion notification: Explicit Congestion Notification (ECN), RFC 3168

Classification and Conditioning

May be desirable to limit traffic injection rate of some class:

- user declares traffic profile (e.g., rate, burst size)
- traffic metered, shaped or dropped if non-conforming



Forwarding (PHB)

- PHB result in a different observable (measurable) forwarding performance behavior
- PHB does not specify what mechanisms to use to ensure required PHB performance behavior
- Examples:
  - Class A gets x% of outgoing link bandwidth over time intervals of a specified length
  - Class A packets leave first before packets from class B

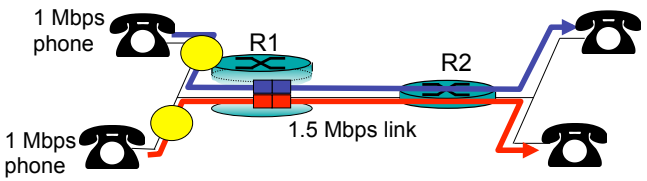
Forwarding (PHB)

PHBs being developed:

- Expedited Forwarding: packet departure rate of a class equals or exceeds specified rate
  - logical link with a minimum guaranteed rate
- Assured Forwarding: e.g. 4 classes of traffic
  - each class guaranteed minimum amount of bandwidth and a minimum of buffering
  - packets each class have one of three possible drop preferences; in case of congestion routers discard packets based on drop preference values

Principles for QOS Guarantees (more)

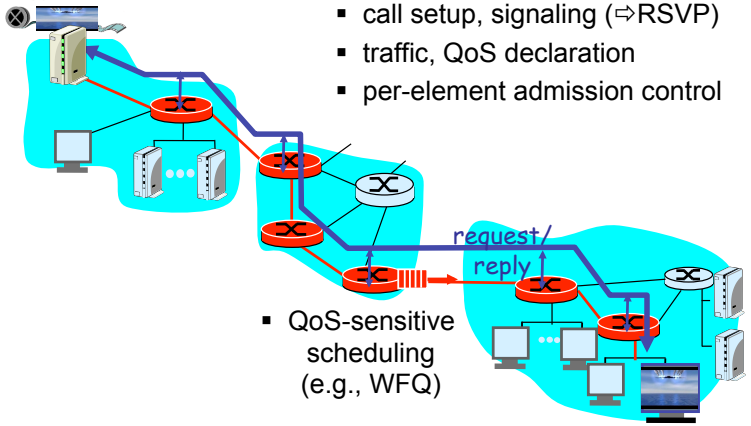
- *Basic fact of life:* can not support traffic demands beyond link capacity



Principle

Call Admission: flow declares its needs, network may block call (e.g., busy signal) if it cannot meet needs

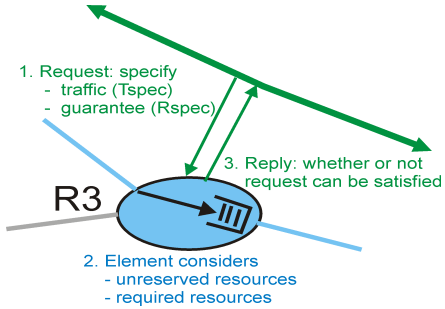
QoS Guarantee Scenario



- Resource reservation
  - call setup, signaling (⇨RSVP)
  - traffic, QoS declaration
  - per-element admission control
- QoS-sensitive scheduling (e.g., WFQ)

Call Admission

- Routers will admit calls based on:
- Flow behavior:
  - R-spec and T-spec
- the current resource allocated at the router to other calls.



IETF Integrated Services

- architecture for providing QOS guarantees in IP networks for individual application sessions
- resource reservation: routers maintain state info (as for VCs) of allocated resources, QoS requests
- admit/deny new call setup requests:

Question: can newly arriving flow be admitted with performance guarantees while not violated QoS guarantees made to already admitted flows?

Call Admission

Arriving session must :

- declare its QoS requirement
  - R-spec: defines the QoS being requested
- characterize traffic it will send into network
  - T-spec: defines traffic characteristics
- signaling protocol: needed to carry R-spec and T-spec to routers (where reservation is required)
  - RSVP

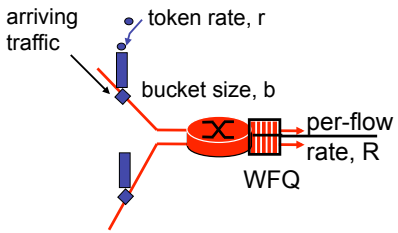
Intserv QoS: Service models [RFC 2211, RFC 2212]

Guaranteed service:

- worst case traffic arrival: leaky-bucket-policed source
- simple (mathematically provable) *bound* on delay [Parekh 1992, Cruz 1988]

Controlled load service:

- "a quality of service closely approximating the QoS that same flow would receive from an unloaded network element."



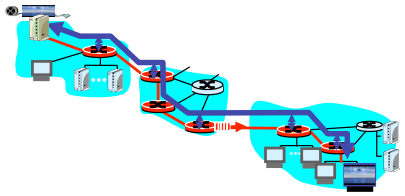
Signaling in the Internet

connectionless (stateless) forwarding by IP routers + best effort service = no network signaling protocols in initial IP design

- New requirement: reserve resources along end-to-end path (end system, routers) for QoS for multimedia applications
- RSVP: Resource Reservation Protocol [RFC 2205]
  - " … allow users to communicate requirements to network in robust and efficient way." i.e., signaling !
- earlier Internet Signaling protocol: ST-II [RFC 1819]

RSVP Design Goals

1. accommodate heterogeneous receivers (different bandwidth along paths)
2. accommodate different applications with different resource requirements
3. make multicast a first class service, with adaptation to multicast group membership
4. leverage existing multicast/unicast routing, with adaptation to changes in underlying unicast, multicast routes
5. control protocol overhead to grow (at worst) linear in # receivers
6. modular design for heterogeneous underlying technologies



RSVP: does not…

- specify *how* resources are to be reserved
  - rather: a mechanism for *communicating needs*
- determine routes packets will take
  - that's the job of routing protocols
  - signaling decoupled from routing
- interact with forwarding of packets
  - separation of control (signaling) and data (forwarding) planes

RSVP: overview of operation

- senders, receiver join a multicast group
  - done outside of RSVP
  - senders need not join group
- sender-to-network signaling
  - *path message:* make sender presence known to routers
  - path teardown: delete sender's path state from routers
- receiver-to-network signaling
  - *reservation message:* reserve resources from sender(s) to receiver
  - reservation teardown: remove receiver reservations
- network-to-end-system signaling
  - path error
  - reservation error

RSVP Messages

- There are two primary types of messages:
- Path messages (*path*)
  - The *path* message is sent from the sender host along the data path and stores the *path state* in each node along the path.
  - The *path state* includes the IP address of the previous node, and some data objects:
    - *sender template* to describe the format of the sender data
    - *sender tspec* to describe the traffic characteristics of the data flow
    - *adspec* that carries advertising data (see RFC 2210 for more details).
- Reservation messages (*resv*)
  - The *resv* message is sent from the receiver to the sender host along the reverse data path. At each node the IP destination address of the *resv* message will change to the address of the next node on the reverse path and the IP source address to the address of the previous node address on the reverse path.
  - The *resv* message includes the *flowspec* data object that identifies the resources that the flow needs.