15-744: Computer Networking

L-10 QoS and Security

QoS and Security

- Denial of service
- IntServ
- DiffServ
- Assigned reading
  - [SWKA00] Practical Network Support for IP Traceback
  - [MVS01] Inferring Internet Denial-of-Service Activity
  - [She95] Fundamental Design Issues for the Future Internet
  - [CSZ92] Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanisms
  - [CF98] Explicit Allocation of Best-Effort Packet Delivery Service

Overview

- Why QOS?
- Integrated services
- RSVP
- Differentiated services
- Security holes in IP stack
- Denial of service traceback
- Firewalls
- Authentication

Motivation

- Internet currently provides one single class of **“best-effort” service**
  - No assurances about delivery
- Existing applications are ***elastic***
  - Tolerate delays and losses
  - Can adapt to congestion
- Future “real-time” applications may be ***inelastic***

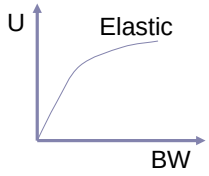
Inelastic Applications

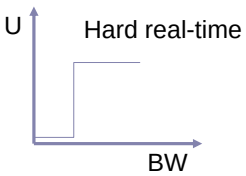
- Continuous media applications
  - **Lower and upper limit** on acceptable performance.
  - BW below which video and audio are not intelligible
  - Internet telephones, teleconferencing with high delay (200 - 300ms) impair human interaction
- Hard real-time applications
  - Require **hard limits on performance**
  - E.g. control applications

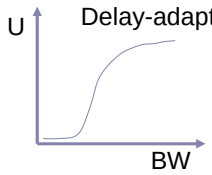
Why a New Service Model?

- What is the **basic objective** of network design?
  - Maximize total bandwidth? Minimize latency?
  - **Maximize user satisfaction** – the total **utility** given to users
- What does utility vs. bandwidth look like?
  - Must be non-decreasing function
  - Shape depends on application

Utility Curve Shapes

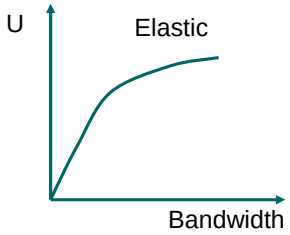






Stay to the right and you are fine for all curves

Utility curve – Elastic traffic

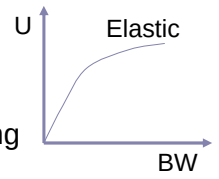


Does equal allocation of bandwidth maximize total utility?

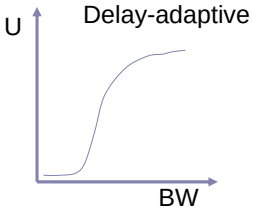
Admission Control

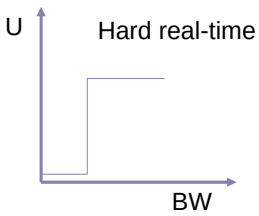
- If U(bandwidth) is concave → elastic applications
  - Incremental utility is decreasing with increasing bandwidth
  - Is always advantageous to have more flows with lower bandwidth
    - No need of admission control;

  This is why the Internet works!



Utility Curves – Inelastic traffic

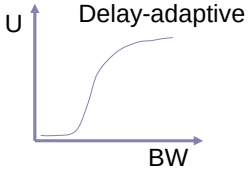




Does equal allocation of bandwidth maximize total utility?

Admission Control

- If U is convex → inelastic applications
  - U(number of flows) is no longer monotonically increasing
  - Need admission control to maximize total utility
- **Admission control** → deciding when the addition of new people would result in reduction of utility
  - Basically avoids overload



Overview

- Why QOS?
- Integrated services
- RSVP
- Differentiated services
- Security holes in IP stack
- Denial of service traceback
- Firewalls
- Authentication

Components of Integrated Services

1. **Type of commitment**
   What does the network promise?
2. Packet scheduling
   How does the network meet promises?
3. Service interface
   How does the application describe what it wants?
4. Establishing the guarantee
   How is the promise communicated to/from the network
   How is admission of new applications controlled?

1. Type of commitment

What kind of promises/services should network offer?

➡ Depends on the **characteristics of the applications** that will use the network ….

Playback Applications

- Sample signal → packetize → transmit → buffer → playback
  - Fits most multimedia applications
- Performance concern:
  - Jitter – variation in end-to-end delay
    - Delay = fixed + variable = (propagation + packetization) + queuing
- Solution:
  - Playback point – delay introduced by buffer to hide network jitter

Characteristics of Playback Applications

- In general lower delay is preferable.
- Doesn't matter when packet arrives as long as it is before playback point
- Network guarantees (e.g. bound on jitter) would make it easier to set playback point
- Applications can tolerate some loss

Applications Variations

- Rigid & adaptive applications
  - Rigid – set fixed playback point
  - Adaptive – adapt playback point
    - Gamble that network conditions will be the same as in the past
    - Are prepared to deal with errors in their estimate
    - Will have an earlier playback point than rigid applications
- Tolerant & intolerant applications
  - Tolerance to brief interruptions in service
- 4 combinations

Applications Variations

Really only two classes of applications

1) **Intolerant and rigid**
2) **Tolerant and adaptive**

Other combinations make little sense

3) Intolerant and adaptive
   - Cannot adapt without interruption
4) Tolerant and rigid
   - Missed opportunity to improve delay

So what service classes should the network offer?

Type of Commitments

- **Guaranteed** service
  - For **intolerant and rigid** applications
  - Fixed guarantee, network meets commitment as long as clients send at match traffic agreement
- **Predicted** service
  - For **tolerant and adaptive** applications
  - Two components
    - If conditions do not change, commit to current service
    - If conditions change, take steps to deliver consistent performance (help apps minimize playback delay)
    - Implicit assumption – network does not change much over time
- **Datagram/best effort service**

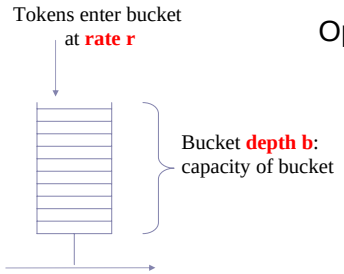
Components of Integrated Services

1. Type of commitment
   What does the network promise?
2. **Packet scheduling**
   How does the network meet promises?
3. **Service interface**
   How does the application describe what it wants?
4. Establishing the guarantee
   How is the promise communicated to/from the network
   How is admission of new applications controlled?

Scheduling for Guaranteed Traffic

- Use **token bucket filter** to characterize traffic
  - Described by rate *r* and bucket depth *b*
- Use **WFQ** at the routers
- Parekh's bound for worst case queuing delay = b/r

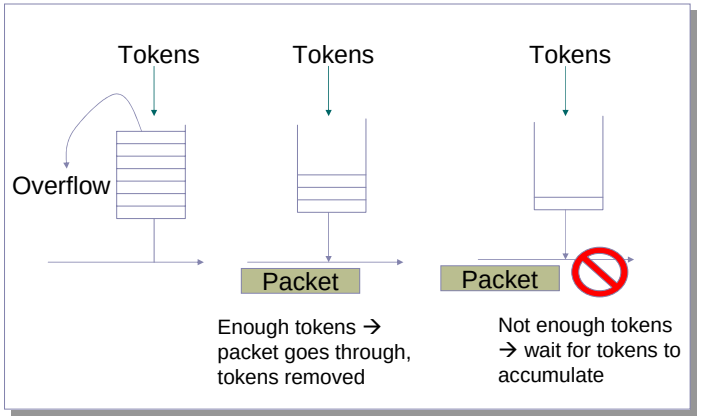
Token Bucket Filter



Operation:

- If bucket fills, tokens are discarded
- Sending a packet of size P uses P tokens
- If bucket has P tokens, packet sent at max rate, else must wait for tokens to accumulate

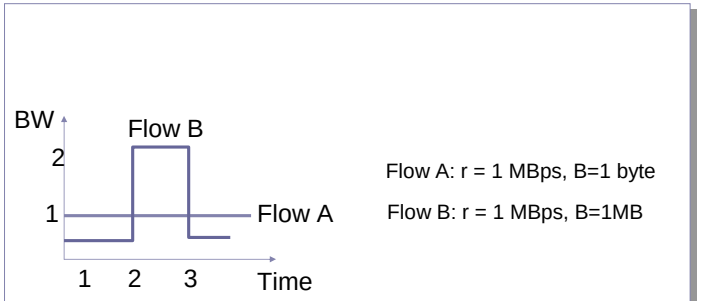
Token Bucket Operation



Token Bucket Characteristics

- On the long run, rate is limited to r
- On the short run, a burst of size b can be sent
- Amount of traffic entering at interval T is bounded by:
  - Traffic = b + r*T
- Information useful to admission algorithm

Token Bucket Specs



Possible Token Bucket Uses

- Shaping, policing, marking
  - Delay pkts from entering net (shaping)
  - Drop pkts that arrive without tokens (policing)
  - Let all pkts pass through, mark ones without tokens
    - Network drops pkts without tokens in time of congestion

Guarantee Proven by Parekh

- Given:
  - Flow *i* shaped with token bucket and leaky bucket rate control (depth *b* and rate *r*)
  - Network nodes do WFQ
- Cumulative queuing delay D_i suffered by flow *i* has upper bound
  - **D_i < b/r,** (where r may be much larger than average rate)
  - Assumes that Σr < link speed at any router
  - All sources limiting themselves to *r* will result in no network queuing

Predicted Service

Goals:

- Isolation
  - Isolates well-behaved from misbehaving sources
- Sharing
  - Mixing of different sources in a way beneficial to all

Mechanisms:

- WFQ
  - Great isolation but no sharing
- FIFO
  - Great sharing but no isolation

Predicted Service

- FIFO jitter increases with the number of hops
  - Use opportunity for sharing across hops
- FIFO+
  - At each hop: measure average delay for class at that router
  - For each packet: compute difference of average delay and delay of that packet in queue
  - Add/subtract difference in packet header
  - Packet inserted into queues expected arrival time instead of actual
    - More complex queue management!
- Slightly decreases mean delay and significantly decreases jitter

Unified Scheduling

- Assume 3 types of traffic: guaranteed, predictive, best-effort
- Scheduling: use WFQ in routers
- Each guaranteed flow gets its own queue
- All predicted service flows and best effort aggregates in single separate queue
  - Predictive traffic classes
    - Multiple FIFO+ queues
    - Worst case delay for classes separated by order of magnitude
    - When high priority needs extra bandwidth – steals it from lower class
  - Best effort traffic acts as lowest priority class

Service Interfaces

- Guaranteed Traffic
  - Host specifies rate to network
  - Why not bucket size b?
    - If delay not good, ask for higher rate
- Predicted Traffic
  - Specifies (r, b) token bucket parameters
  - Specifies delay D and loss rate L
  - Network assigns priority class
  - Policing at edges to drop or tag packets
    - Needed to provide isolation – why is this not done for guaranteed traffic?
      - WFQ provides this for guaranteed traffic

Establishing the guarantee

- Admission control
  - Don't give all bandwidth to real-time traffic
    - 90% real-time, 10% best effort
  - Very much dependent on how large fluctuations in network traffic and delay are
    - Should measure this dynamically instead of having built-in assumptions

Overview

- Why QOS?
- Integrated services
- RSVP
- Differentiated services
- Security holes in IP stack
- Denial of service traceback
- Firewalls
- Authentication

Components of Integrated Services

1. Type of commitment
   What does the network promise?
2. Packet scheduling
   How does the network meet promises?
3. Service interface
   How does the application describe what it wants?
4. **Establishing the guarantee**
   How is the promise communicated
   How is admission of new applications controlled?

Role of RSVP

- Rides on top of unicast/multicast routing protocols
- Carries resource requests all the way through the network
- At each hop consults admission control and sets up reservation. Informs requester if failure

RSVP Goals

- Used on connectionless networks
  - Should not replicate routing functionality
  - Should co-exist with route changes
- Support for multicast
  - Different receivers have different capabilities and want different QOS
  - Changes in group membership should not be expensive
  - Reservations should be aggregate – I.e. each receiver in group should not have to reserve
  - Should be able to switch allocated resource to different senders
- Modular design – should be generic “signaling” protocol
- Result
  - Receiver-oriented
  - Soft-state

RSVP Service Model

- Make reservations for simplex data streams
- Receiver decides whether to make reservation
- Control msgs in IP datagrams (proto #46)
- PATH/RESV sent periodically to refresh soft state
- One pass:
  - Failed requests return error messages - receiver must try again
  - No e2e ack for success

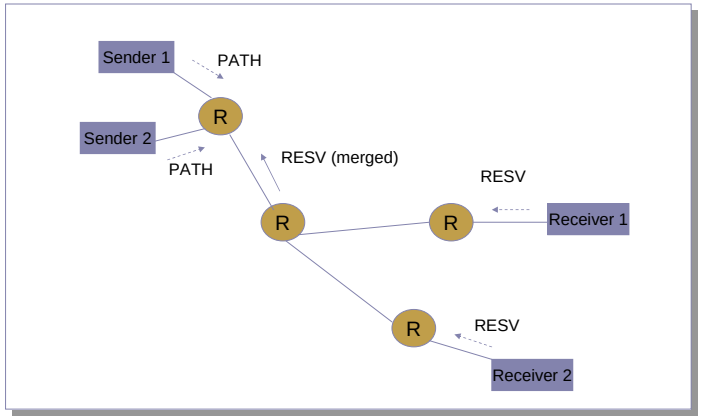
PATH Messages

- PATH messages carry sender's Tspec
  - Token bucket parameters
- Routers note the direction PATH messages arrived and set up *reverse path* to sender
- Receivers send RESV messages that follow reverse path and setup reservations
- If reservation cannot be made, user gets an error

RESV Messages

- Forwarded via reverse path of PATH
- Queuing delay and bandwidth requirements
- Source traffic characteristics (from PATH)
- Filter specification
  - Which transmissions can use the reserved resources
- Router performs admission control and reserves resources
  - If request rejected, send error message

PATH and RESV Messages



Routing Changes

- Routing protocol makes routing changes
- In absence of route or membership changes, periodic PATH and RESV msgs refresh established reservation state
- When change, new PATH msgs follow new path, new RESV msgs set reservation
- Non-refreshed state times out automatically

Announcements

- Presentations
  - 15 minute slots
    - 12 minute talk, 3 minutes for Q&A
    - Strictly enforced
  - 14 groups → 210 minutes total → class will end at 4:30!
- Writeup
  - Approx. 12 pgs single column (8 pgs double column)
  - Due Monday 12/13
- Final exam
  - 1-3pm (not 1-4pm) on Friday 12/17 in BH 136A
  - Similar to midterm in style
- Homework
  - Extension to 12/13
  - However, sample question for exam due on Friday (treat as handin)

Overview

- Why QOS?
- Integrated services
- RSVP
- Differentiated services
- Security holes in IP stack
- Denial of service traceback
- Firewalls
- Authentication

DiffServ

- Analogy:
  - Airline service, first class, coach, various restrictions on coach as a function of payment
- Best-effort expected to make up bulk of traffic, but revenue from first class important to economic base (will pay for more plentiful bandwidth overall)
- Not motivated by real-time! Motivated by economics and assurances

Basic Architecture

- Agreements/service provided within a domain
  - Service Level Agreement (SLA) with ISP
- Edge routers do traffic conditioning
  - Perform per aggregate shaping and policing
  - Mark packets with a small number of bits; each bit encoding represents a class or subclass
- Core routers
  - Process packets based on packet marking and defined per hop behavior
- More scalable than IntServ
  - No per flow state or signaling

Per-hop Behaviors (PHBs)

- Define behavior of individual routers rather than end-to-end services – there may be many more services than behaviors
- Multiple behaviors – need more than one bit in the header
- Six bits from IP TOS field are taken for Diffserv code points (DSCP)

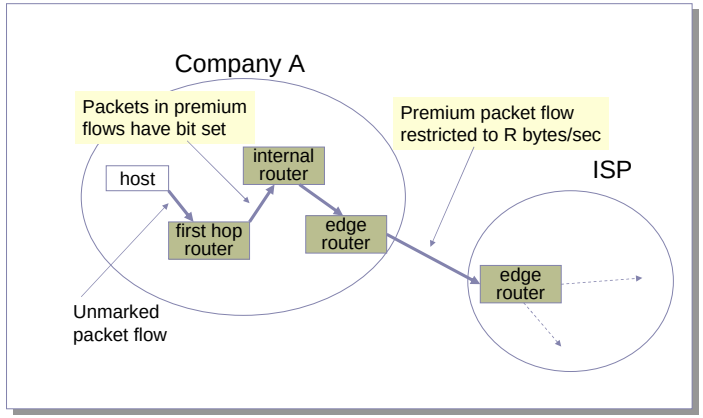
Per-hop Behaviors (PHBs)

- Two PHBs defined so far
- Expedited forwarding aka premium service (type P)
  - Possible service: providing a virtual wire
  - Admitted based on peak rate
  - Unused premium goes to best effort
- Assured forwarding (type A)
  - Possible service: strong assurance for traffic within profile & allow source to exceed profile
  - Based on expected capacity usage profiles
  - Traffic unlikely to be dropped if user maintains profile
  - Out-of-profile traffic marked

Expedited Forwarding PHB

- User sends within profile & network commits to delivery with requested profile
  - Signaling, admission control may get more elaborate in future
- Rate limiting of EF packets at edges only, using token bucket to shape transmission
- Simple forwarding: classify packet in one of two queues, use priority
  - EF packets are forwarded with minimal delay and loss (up to the capacity of the router)

Expedited Forwarding Traffic Flow



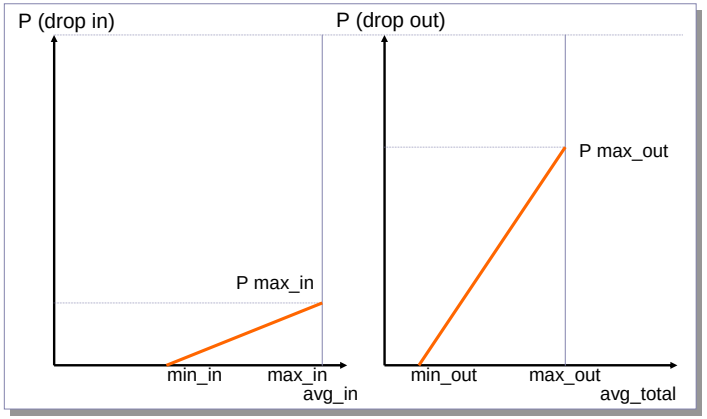
Assured Forwarding PHB

- User and network agree to some traffic profile
  - Edges mark packets up to allowed rate as “in-profile” or low drop precedence
  - Other packets are marked with one of 2 higher drop precedence values
- A congested DS node tries to protect packets with a lower drop precedence value from being lost by preferably discarding packets with a higher drop precedence value
  - Implemented using RED with In/Out bit

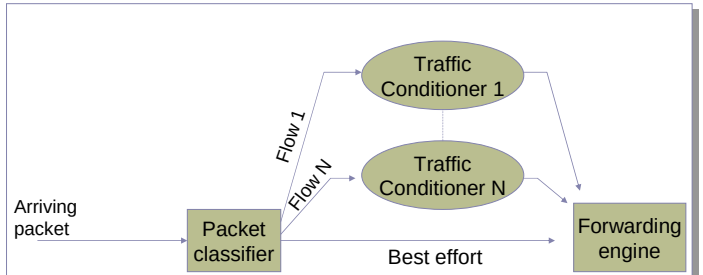
Red with In or Out (RIO)

- Similar to RED, but with two separate probability curves
- Has two classes, “In” and “Out” (of profile)
- “Out” class has lower Min_{thresh}, so packets are dropped from this class first
  - Based on queue length of all packets
- As avg queue length increases, “in” packets are also dropped
  - Based on queue length of only “in” packets

RIO Drop Probabilities

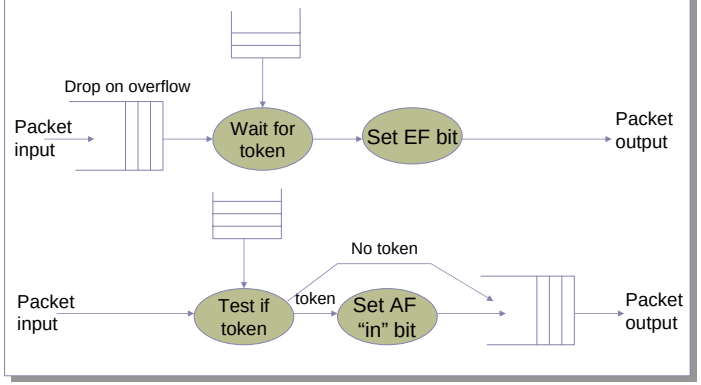


Edge Router Input Functionality



classify packets based on packet header

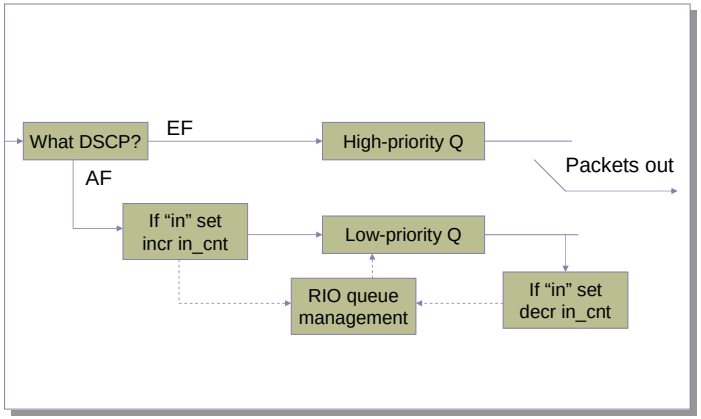
Traffic Conditioning



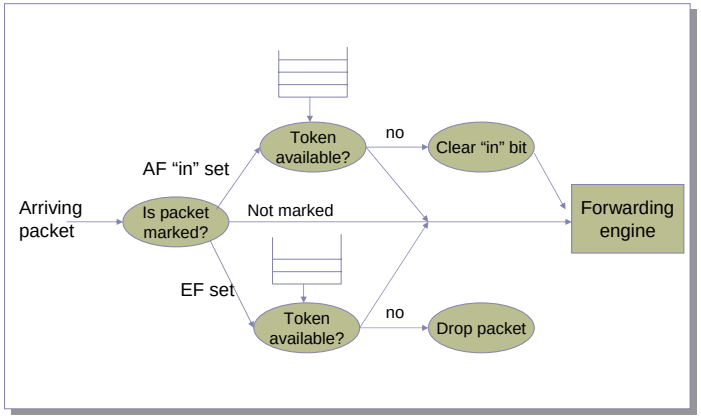
Output Forwarding

- 2 queues: EF packets on higher priority queue
- Lower priority queue implements RED "In or Out" scheme (RIO)

Router Output Processing



Edge Router Policing



Comparison

	Best-Effort	Diffserv	Intserv
Service	• Connectivity • No isolation • No guarantees	• Per aggregation isolation • Per aggregation guarantee	• Per flow isolation • Per flow guarantee
Service Scope	• End-to-end	• Domain	• End-to-end
Complexity	• No set-up	• Long term setup	• Per flow setup
Scalability	• Highly scalable • (nodes maintain only routing state)	• Scalable (edge routers maintains per aggregate state; core routers per class state)	• Not scalable (each router maintains per flow state)

Overview

- Why QOS?
- Integrated services
- RSVP
- Differentiated services
- Security holes in IP stack
- Denial of service traceback
- Firewalls
- Authentication

Basic IP

- End hosts create IP packets and routers process them purely based on destination address alone (not quite in reality)
- Problem – End host may lie about other fields and not affect delivery
  - Source address – host may trick destination into believing that packet is from trusted source
    - Many applications use IP address as a simple authentication method
    - Solution – reverse path forwarding checks, better authentication
  - Fragmentation – can consume memory resources or otherwise trick destination/firewalls
    - Solution – disallow fragments

Denial of Service

- Objective of attack: make a service unusable, usually by overloading the server or network
- Example: SYN flooding attack
  - Send SYN packets with bogus source address
  - Server responds with SYNACK keeps state about TCP half-open connection
    - Eventually server memory is exhausted with this state
  - Solution: SYN cookies – make the SYNACK contents purely a function of SYN contents, therefore, it can be recomputed on reception of next ACK
- More recent attacks have used bandwidth floods
  - How do we stop these?

Bandwidth DOS Attacks - Possible Solutions

- Ingress filtering – examine packets to identify bogus source addresses
- Link testing – have routers either explicitly identify which hops are involved in attack or use controlled flooding and a network map to perturb attack traffic
- Logging – log packets at key routers and post-process to identify attacker's path
- ICMP traceback – sample occasional packets and copy path info into special ICMP messages
- IP traceback

Routing

- Source routing
  - Destinations are expected to reverse source route for replies
  - Problem – Can force packets to be routed through convenient monitoring point
    - Solution – Disallow source routing – doesn't work well anyway!

Routing

- Routing protocol
  - Malicious hosts may advertise routes into network
  - Problem – Bogus routes may enable host to monitor traffic or deny service to others
    - Solutions
      - Use policy mechanisms to only accept routes from or to certain networks/entities
      - In link state routing, can use something like source routing to force packets onto valid route
      - Routing registries and certificates

ICMP

- Reports errors and other conditions from network to end hosts
- End hosts take actions to respond to error
- Problem
  - An entity can easily forge a variety of ICMP error messages
    - Redirect – informs end-hosts that it should be using different first hop route
    - Fragmentation – can confuse path MTU discovery
    - Destination unreachable – can cause transport connections to be dropped

TCP

- Each TCP connection has an agreed upon/negotiated set of associated state
  - Starting sequence numbers, port numbers
  - Knowing these parameters is sometimes used to provide some sense of security
- Problem
  - Easy to guess these values
    - Listening ports #'s are well known and connecting port #'s are typically allocated sequentially
    - Starting sequence number are chosen in predictable way
  - Solution – make sequence number selection more random

Sequence Number Guessing Attack

Attacker → Victim: SYN(ISN_x), SRC=Trusted Host

Victim → Trusted Host: SYN(ISN_s), ACK(ISN_x)

Attacker → Victim: ACK($ISN_{guess\ of\ s}$), SRC=Trusted Host

Attacker → Victim: ACK($ISN_{guess\ of\ s}$), SRC=T, data = "rm -r /"

- Attacker must also make sure that Trusted Host does not respond to SYNACK
- Can repeat until guess is accurate

TCP

- TCP senders assume that receivers behave in certain ways (e.g. when they send acks, etc.)
  - Congestion control is typically done on a "packet" basis while the rest of TCP is based on bytes
- Problem – misbehaving receiver can trick sender into ignoring congestion control
  - Ack every byte in packet!
  - Send extra duplicate acks
  - Ack before the data is received (needs some application level retransmission – e.g. HTTP 1.1 range requests)
  - Solutions
    - Make congestion control byte oriented
    - Add nonces to packets – acks return nonce to truly indicate reception

DNS

- Users/hosts typically trust the host-address mapping provided by DNS
- Problems
  - Zone transfers can provide useful list of target hosts
  - Interception of requests or comprise of DNS servers can result in bogus responses
  - Solution – authenticated requests/responses

Overview

- Why QOS?
- Integrated services
- RSVP
- Differentiated services
- Security holes in IP stack
- Denial of service traceback
- Firewalls
- Authentication

Bandwidth DOS Attacks

- Possible solutions
  - Ingress filtering – examine packets to identify bogus source addresses
  - Link testing – how routers either explicitly identify which hops are involved in attack or use controlled flooding and a network map to perturb attack traffic
  - Logging – log packets at key routers and post-process to identify attacker's path
  - ICMP traceback – sample occasional packets and copy path info into special ICMP messages
  - IP traceback

IP Traceback

- Node append (record route) – high computation and space overhead
- Node sampling – each router marks its IP address with some probability *p*
  - P(receiving mark from router d hops away) = $p(1-p)^{d-1}$
  - p > 0.5 prevents any attacker from inserting false router
  - Must infer distance by marking rate → relatively slow
  - Doesn't work well with multiple routers at same distance → I.e. multiple attackers

IP Traceback

- Edge sampling
  - Solve node sampling problems by encoding edges & distance from victim in messages
  - Start router sets "start" field with probability *p* and sets distance to 0
  - If distance is 0, router sets "end" field
  - All routers increment distance
  - As before, P(receiving mark from router d hops away) = $p(1-p)^{d-1}$
- Multiple attackers can be identified since edge identifies splits in reverse path

Edge Sampling

- Major problem – need to add about 72bits (2 address + hop count) of info into packets
- Solution
  - Encode edge as xor of nodes → reduce 64 bits to 32 bits
  - Ship only 8bits at a time and 3bits to indicate offset → 32 bits to 11bits
  - Use only 5 bit for distance → 8bits to 5bits
  - Use IP fragment field to store 16 bits
    - Some backward compatibility issues
    - Fragmentation is rare so not a big problem

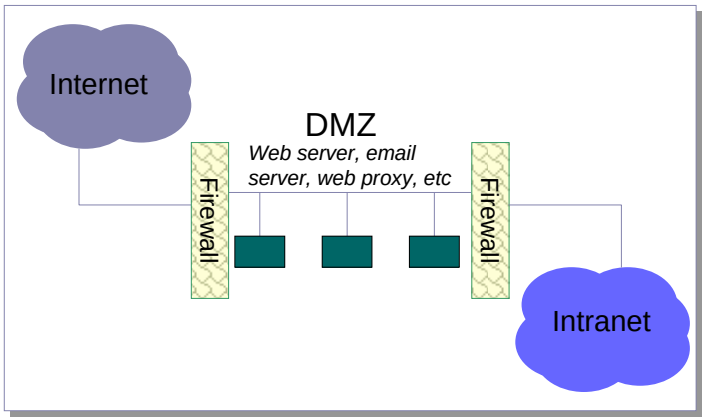
Overview

- Why QOS?
- Integrated services
- RSVP
- Differentiated services
- Security holes in IP stack
- Denial of service traceback
- Firewalls
- Authentication

Firewalls

- Basic problem – many network applications and protocols have security problems that are fixed over time
  - Difficult for users to keep up with changes and keep host secure
  - Solution
    - Administrators limit access to end hosts by using a firewall
    - Firewall and limited number of machines at site are kept up-to-date by administrators

Typical Firewall Topology



Types of Firewalls - Proxy

- End host connects to proxy and asks it to perform actions on its behalf
  - Policy determines if action is secure or insecure
- Transport level relays (SOCKS)
  - Ask proxy to create, accept TCP (or UDP) connection
  - Cannot secure against insecure application
- Application level relays (e.g. HTTP, FTP, telnet, etc.)
  - Ask proxy to perform application action (e.g. HTTP Get, FTP transfer)
  - Can use application action to determine security
- Requires applications (or dynamically linked libraries) to be modified to use the proxy
- Considered to be the most secure since it has most information to make decision

Types of Firewalls - Packet Filters

- Set of filters and associated actions that are used on a packet by packet basis
- Filters specify fields, masks and values to match against packet contents, input and output interface
- Actions are typically forward or discard
- Such systems have difficulty with things like fragments and a variety of attacks
- Typically a difficult balance between the access given and the ability to run applications
  - E.g. FTP often needs inbound connections on arbitrary port numbers – either make it difficult to use FTP or limit its use

Types of Firewalls - Stateful Packet Filters

- Typically allow richer parsing of each packet (variable length fields, application headers, etc.)
- Actions can include the addition of new rules and the creation of state to process future packets
  - Often have to parse application payload to determine "intent" and determine security considerations
- Rules can be based on packet contents and state created by past packets
- Provides many of the security benefits of proxies but without having to modify applications

Overview

- Why QOS?
- Integrated services
- RSVP
- Differentiated services
- Security holes in IP stack
- Denial of service traceback
- Firewalls
- Authentication

Trusted Intermediaries

Symmetric key problem:

- How do two entities establish shared secret key over network?

Solution:

- Trusted key distribution center (KDC) acting as intermediary between entities

Public key problem:

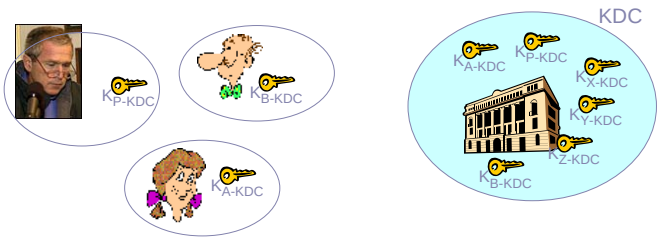
- When Alice obtains Bob's public key (from web site, e-mail, diskette), how does she know it is Bob's public key, not Trudy's?

Solution:

- Trusted certification authority (CA)

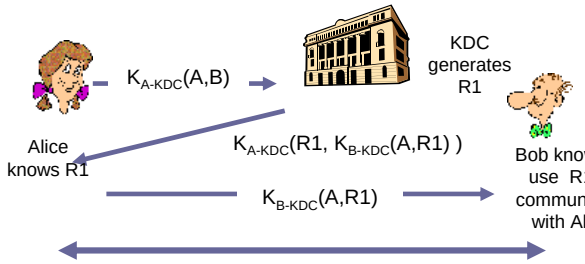
Key Distribution Center (KDC)

- Alice, Bob need shared symmetric key.
- KDC: server shares different secret key with *each* registered user (many users)
- Alice, Bob know own symmetric keys, $K_{A\text{-}KDC}$ $K_{B\text{-}KDC}$, for communicating with KDC.



Key Distribution Center (KDC)

Q: How does KDC allow Bob, Alice to determine shared symmetric secret key to communicate with each other?



Alice and Bob communicate: using R1 as *session key* for shared symmetric encryption

Kerberos

- Objective: provide a way for services to authenticate clients
- A client must present a ticket to use service
  - Ticket is obtained from Kerberos system
    - Contains client ID, session key
    - Ticket is encrypted using service's private key known only to service and Kerberos
    - Ensures that only Kerberos could have created ticket
  - Client uses authenticator to prevent replay of tickets
    - Contains client ID, network address, time of day
    - Encrypted using session key
    - Only if recent and Ids match is ticket valid
    - Forces time to be synchronized within few minutes

Kerberos

- Obtaining tickets
  - Client sends message to Kerberos
    - Contains service ID, client ID, time of day
  - Response encrypted with client's private key
    - Contains ticket, session key and timestamp
  - First ticket gotten is for Kerberos Ticket Granting Service
    - All other requests sent to the TGS using the TGS session key
    - Avoids having to provide passwd for each ticket

Kerberos

- To get the TGS ticket, the client contacts the KDS

  User → workstation (WS): N_{client}

  WS → Key Distribution Service (KDS): $\{N_{client}, N_{TGS}, T_{current}\}$

  $Ticket_{TGS} = \{K_{session}, N_{client}, N_{TGS}, T_{current}, WS, Lifetime\}K_{TGS}$

  KDS → WS: $\{K_{session}, N_{TGS}, Lifetime, T_{current}, Ticket_{TGS}\}K_{client}$

  - Above message is subject to know plaintext attack!

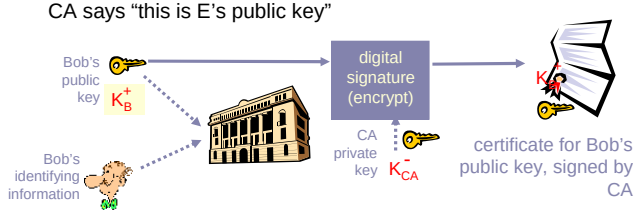
  User → WS: password == K_{client} → used to decrypt previous message
- To use a service, client creates authenticator:

  $Authenticator = \{N_{client}, T_{current}, WS\}K_{session}$

  WS → service: {Authenticator, Ticket}
- Further exchanges are similar to KDS exchange but with TGS using $K_{session}$ instead of K_{client}

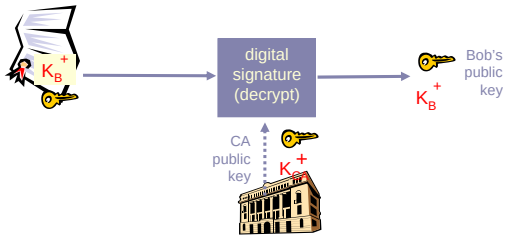
Certification Authorities

- Certification authority (CA): binds public key to particular entity, E.
- E (person, router) registers its public key with CA.
  - E provides "proof of identity" to CA.
  - CA creates certificate binding E to its public key.
  - Certificate containing E's public key digitally signed by CA – CA says "this is E's public key"



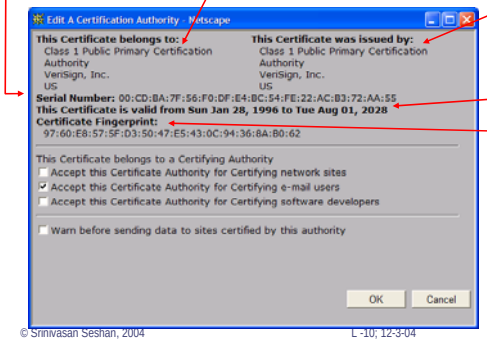
Certification Authorities

- When Alice wants Bob's public key:
  - Gets Bob's certificate (Bob or elsewhere).
  - Apply CA's public key to Bob's certificate, get Bob's public key



Certificate Contents

- Serial number (unique to issuer)
- info about certificate owner, including algorithm and key value itself (not shown)



- Info about certificate issuer
- Valid dates
- Digital signature by issuer

Secure Sockets Layer (SSL)

- Transport layer security to any TCP-based app using SSL services.
- Used between Web browsers, servers for e-commerce (shttp).
- Security services:
  - Server authentication
  - Data encryption
  - Client authentication (optional)
- Server authentication:
  - SSL-enabled browser includes public keys for trusted CAs.
  - Browser requests server certificate, issued by trusted CA.
  - Browser uses CA's public key to extract server's public key from certificate.
- Check your browser's security menu to see its trusted CAs.

SSL (continued)

Encrypted SSL session:

- Browser generates *symmetric session key*, encrypts it with server's public key, sends encrypted key to server.
- Using private key, server decrypts session key.
- Browser, server know session key
  - All data sent into TCP socket (by client or server) encrypted with session key.
- SSL: basis of IETF Transport Layer Security (TLS).
- SSL can be used for non-Web applications, e.g., IMAP.
- Client authentication can be done with client certificates.

Announcements

- Presentations
  - 15 minute slots
    - 12 minute talk, 3 minutes for Q&A
    - Strictly enforced
  - 14 groups → 210 minutes total → class will end at 4:30!
- Writeup
  - Approx. 12 pgs single column (8 pgs double column)
  - Due Monday 12/13
- Final exam
  - 1-3pm (not 1-4pm) on Friday 12/17 in BH 136A
  - Similar to midterm in style

Why a New Service Model?

- Given the shape of different utility curves – clearly equal allocation of bandwidth does not maximize total utility
- In fact, desirable rate for some flow may be 0.

Admission Control

- Caveats
  - Admission control can only turn away new requests → sometimes it may be have been better to terminate an existing flow
  - U(0) != 0 → users tend to be very unhappy with no service – perhaps U should be discontinuous here
- Alternative → overprovision the network
  - Problem: high variability in usage patterns
  - "Leading-edge" users make it costly to overprovision
  - Having admission control seems to be a better alternative

Other QOS principles

1. Admission Control
2. Marking of packets is needed to distinguish between different classes.
3. Protection (isolation) for one class from another.
4. While providing isolation, it is desirable to use resources as efficiently as possible → sharing.

How to Choose Service – Implicit

Network could examine packets and **implicitly** determine service class

- No changes to end hosts/applications
- Fixed set of applications supported at any time
- Can't support applications in different uses/modes easily
- Violates layering/modularity

How to Choose Service – Explicit

Applications could **explicitly** request service level

- Why would an application request lower service?
  - Pricing
  - Informal social conventions
  - Problem exists in best-effort as well → congestion control
- Applications must know network service choices
  - Difficult to change over time
  - All parts of network must support this → places greater burden on portability of IP

Parekh Bound on Delay Across Net

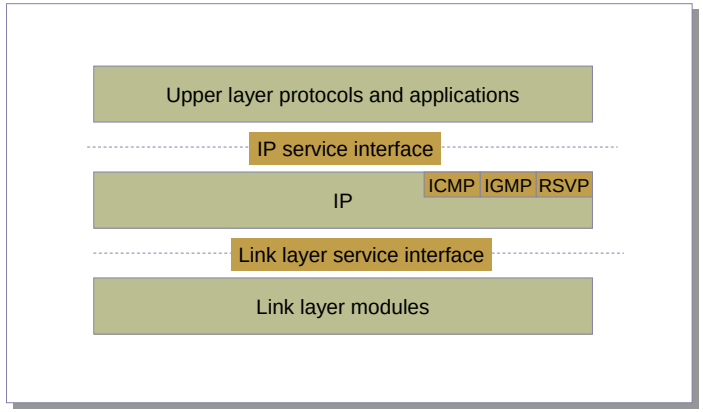
D_i = (bucket size/weighted rate allocated) + [(nhops - 1) * MaxPacketLen / weighted rate allocation] + Σ m=1 to hop_i (max packet length / outbound bw at hop)

- 1st term: delay when running at full speed
- 2nd term: packetization effects
- 3rd term: added delay due to packet approx of FQ (goes away as data rate increases)

IETF Internet Service Classes

- Guaranteed service
  - Firm bounds on e2e delays and bandwidth
- Controlled load
  - "A QoS closely approximating the QoS that same flow would receive from an unloaded network element, but uses capacity (admission) control to assure that this service is received even when the network element is overloaded"
- Best effort

Reservation Protocol: RSVP



Basic Message Types

- PATH message
- RESV message
- CONFIRMATION message
  - Generated only upon request
  - Unicast to receiver when RESV reaches node with established state
- TEARDOWN message
- ERROR message (if PATH or RESV fails)

Packet Classifying and Scheduling

- Each arriving packet must be:
  - **Classified**: associated with the application reservation
    - Fields: source + destination address, protocol number, source + destination port
  - **Scheduled**: managed in the queue so that it receives the requested service
    - Implementation not specified in the service model, left up to the implementation

RSVP and Multicast

- Reservations from multiple receivers for a single sender are merged together at branching points
- Reservations for multiple senders may not be added up:
  - Audio conference, not many talk at same time
  - Only subset of speakers (filters)
  - Mixers and translators

Reservation Styles

- How filters are used
- Three styles
  - Wildcard/No filter – does not specify a particular sender for group
  - Fixed filter – sender explicitly specified for a reservation
  - Dynamic filter – valid senders may be changed over time
- Receiver chooses but sender can force no-filter by setting F-Flag

Changing Reservation

- Receiver-oriented approach and soft state make it easy to modify reservation
- Modification sent with periodic refresh