Notes

Protocol = An agreed format for messages, expressed by a packet header, an optional message component, and a set of rules for the exchange of messages between computers.

Messages must

- happen in an agreed to order
- travel from the sender to the correct receiver(s),
- contain the correct, unambiguous, data (what arrives must be what was sent)

Different levels of responsibility = different protocols

eg, at the lowest level concern about correctly transmitting bits (0s and 1s) at another level concerned about transferring files

One way to separate responsibilities = **OSI/ISO reference model** = 7 layer model.

layers Application<->Presentation<->Session<->Transport<->Network<->Data Link<->Physical<->

Mnemonic: "All People Seem To Need Data Processing."

Why 7?

- each layer corresponds to a different layer of abstraction
- Each layer provides a well defined, independent function
- With each layer unique protocol standard should be enforceable
- There should be a minimum of traffic between layers
- Distinct functions should be in separate layers

Each layer

Physical

- transmits the raw bit steam over the physical communication medium

Data-Link Layer

- takes the bit stream and construct logical chunks of data called *frames*

Network Layer

- provides the connection between "end systems" across a network.
- functions include: routings, relaying, flow control, sequencing

Transport Layer

- provides a reliable end-to-end service independent of the network topology. This is achieved by splitting messages into network sized *packets* and joining them back together again at the other end

Session Layer

Is the first upper layer crucial to internetworking and manages the dialogue between end systems, Typically the session layer provides:

- establishment and closing of connections.
- synchronization to allow checking and recovery of data.
- negotiation of full and half duplex communication

Presentation Layer

- provides a standard format for transferred information by overcoming compatibility problems between systems using dissimilar data encoding rules and possible different output and input technologies

Application Layer

- provides the interface between the application processes. Functions such as file transfer, remote job execution and application independent virtual terminal support are provided

Responsibilities of the physical and Data link layers

Physical Layer = provides the topology of the network, provides the transfer medium. Encodes/Decodes (depending on sending or receiving) data into a transmission signal supported by the medium, monitors transmission errors, detects arriving signal levels to synchronize speed and timing

Data Link Layer = Constructs data frames using the format of the physical layer, calculates Cyclic Redundancy Codes information, Examines arriving and passing device addresses in data. Acknowledges receipt of a frame.

Metrics Of Network Measurement

There are two components to performance - latency and bandwidth.

Latency/propagation delay = amount of time it takes for the first bit to reach its destination. Round trip time is latency * 2. The time to send and then receiver

Bandwidth = the number of bits that can be fit through a network connection, per unit time.

Throughput = the measured number of bits per second (achieved) whilst bandwidth is nominal (peak) number of bits per second possible.

Examples:

> A car can carry 4 people to Bunbury in 2 hours. Thus, the bandwidth is 2 people/hour

> A bus can take 60 passengers, and arrive in 2 hours. Thus, the bandwidth is 30 people/hour, but the latency is still 2 hours.

The latency of a message is the total time for the whole message to arrive.

TLatency = TPropagation + TTransmit + TQueue TPropagation = distance / speed-of-propagation-in-medium TTransmit = message-size / bandwidth TQueue = time-spent-in-local-and-remote-operating-system-and-switch-queues

The Physical Layer and Transmission Errors

Rates of transmission errors are described by their probability of occurring, or by their expected bit-error-rate (BER)

Transmission Errors usually occur in bursts and are caused by:

- Thermal background noise
- Impluse noise, electrical arcing
- distorted frequency
- crosstalk between adjacent wires

A burst begins at the first bit that is corrupted (inverted) and ends at the last bit corrupted - the bits between are not necessarily all modified.

How data is placed in Frames

Simple timing gaps = bad = hardware devices run at slightly different speeds. Resulting in skewing is time is relied upon

A first attempt at a solution involves counting the size of a frame and placing this count in the data link header

To overcome synchronization problems, special byte sequences are often used to prefix and suffix the data (enveloping it)

As these special bytes may themselves be required data we must “escape” their special meaning.

This is called **byte stuffing**

Phase encoding of signals

A digital signal is a sequence of discrete, discontinuous pulses, or *signal elements*

If all signal elements have the same (voltage) sign they are termed *unipolar*

The *modulation rate* of a LAN is the number of signal Elements (transitions) per second, or the *baud rate*

To correctly interpret a signal the receiver must know the length (time) of each signal element and the expected voltage levels of the bits 0 and 1

Error detection and correction

Data may be modified so that errors can either be:

- detected (you can only tell that the data is wrong) OR
- corrected (you can unambiguously tell what the data *should* have been and hence you can correct it)

Correction is required when communication must be simplex, but correction is expensive.

codewords are constructed consisting of both data and check bits

The **Hamming distance** between two codewords consists of the number of bit positions in which they differ.

The different is performed Modulo-2 Or XOR

To detect delta errors a distance of delta + 1 is required

e.g to detect 1 bit in error requires that there is no word with a distance of 1 from a valid word

TO correct delta errors a distance of 2 delta + 1 is required so that even with delta errors the damaged codeword is the closest to one valid codeword.

Hamming's correction of Single-Bit Errors

A single transmission consists of m bits for the message and r bits of seemingly redundant information

We thus transmit n = m + r bits when transmitting a message

The critical question is "how much additional information (the redundant bits) do we need to transmit so that the receiver can correct errors?

Each of 2ˆm possible messages has n illegal code words which are distance 1 from it. Therefore, each message word requires n+1 distinct bit patterns (1 legal one, and n illegal ones)

in 1950, mathematician and turning award winner Richard Hamming developed a method which achieves the lower bound of:

```
m+r+1 <= 2^r
```

Given a code word of 7 data bits and 4 check bits, we number the code word from 1 from the left hand side.

Each bit whose original position is a power of 2 is a check bit and forces the parity of some "collection" of bits including itself. Parity may be forced to be either even or odd.

A data bit contributes to the parity of all bits in its *decomposition into a sum of powers of 2*

For example:

```
3=2+1
7=4+2+1
11=8+2+1
```

Making a list of these we then put the data bits in their positions and calculate each check bit

Then its sent

When a codeword is received at the other end we must check (and possibly correct) it.

To correct when it arrives:

initialize a counter, c to 0

Examine each check bit in position k={1,2,4,8} to see if it has the correct parity.

if not, add k to c

When all check bits have been examined, and the counter c is 0, then codeword is correct. Otherwise, the incorrect bit is in position c.

Cyclic Redundancy Codes (CRC's)

The extra bits used to detect errors are called checksum bits.

Polynomial Codes

A polynomial is represented by a but string with 1 for each power represented in the polynomial, and 0 otherwise.

Polynomial arithmetic is performed Modulo-2 or XOR

Both the sender and receiver agree on a generator polynomial, G(x), with high and low order bits 1.

The message, M is also interpreted as a polynomial

The checksum (when calculated) is appended to the message so that (M+checksum), known as the transmission T(x) is divisible by G(x)

Can't find all kind of errors

The Data Link Layer

Provides the following services between the Physical and Network Layers:

- Bundling and unbundling groups of bits into frames for the physical layer
- Throttling the flow of frames between sender and receiver
- detecting and correcting "higher-level" transmission errors, such as the sequencing of packets from the network layer.

(due to the OSI philosophy" the data link layer in the sender believes it is talking directly to the data link layer in the receiver (a peer-to-peer relationship))

Three levels of Data Link Layer Complexity

- simplex connectionless - the sender simply sends its frames without waiting for the receiver to acknowledge them. No attempt is made to detect or re-transmit lost frames. Most modern LAN technologies (such as Ethernet) use this method and leave error resolution to higher layers.

This is also termed an "unacknowledged connectionless service".

- Half-duplex connectionless - each frame sent is individually acknowledged. Frames which are lost or garbled are retransmitted if the receiver requests them (again) or after a suitable timeout.

This is also termed an acknowledged connectionless service.

- Full duplex connection-oriented - Each frame is individually numbered and is guaranteed by the data link layer to be received once and only once and is in the right order. The result is that the data link layer presents a reliable frame *stream* to the network layer.

This is also termed an acknowledged connection service

Stop and wait protocol

Basic method where the sender transmits one data packet at a time and then waits for an acknowledgement from the receiver before sending the next packet.

Frame piggybacking

Technique in which the acknowledgment of a received data packet is included ("Piggybacked") on a data packet being send back to the original sender, reducing the number of messages needed for communication.

Sliding Window Protocols

The sender has to wait until an acknowledgement arrives from the receiver.

Over links with long propagation delays this results in a very inefficient use of available bandwidth.

To fix this we want to keep the sender and the medium "busy". We can achieve this by permitting the sender to send more than a single frame. Whilst waiting for the first acknowledgement.

In sliding window, or clock, protocols we have these properties.

- the sender has a sending window consisting of a list (array) of frames that have been

- sent but not acknowledged.
- The sender's window size grows as more frames are sent but not yet acknowledged.
- The receiver has a receiving window consisting of frames it is willing to accept. The receiver's window size remains constant.
- Each frame being sent has a sequence number from 0 to 2ˆ(n-1) (which fits in n bits). Stopand-wait and PAR have n=1.
- A window size of 1 implies that frames are only accepted in order.

Sliding window protocols remain synchronized under conditions of premature timeouts, garbled frames and lost frames.

Frame Pipe lining

if the distance (in time) between sender and receiver is long. OR expensive. Then bandwidth should be maximized.

The solution is to permit *multiple outstanding frames*

This is made possible by having the sender transmit many frames until the medium is "full" and then wait for acknowledgements indicating that frames have been received correctly before proceeding.

The obvious question is "what do we do when either data frames or acknowledgements are lost?"

Two Solutions

1. The go-back-N-protocol

requires the receiver to simply discard all frames *after* a bad one

- The senders window size corresponds to the number of frames transmitted by not yet received - it varies, grows and shrinks over time

- The recievers window size corresponds to the number of frames that the receiver is willing to receive - it is always fixed, 1

2. The selective Repeat Protocol

The go-back-N protocol wastes bandwidth on retransmitted frames if the error rate is high

Alternatively In the selective repeat protocol the receiver can buffer all received frames (up to some limit) and simply wait for the bad frame to be retransmitted.

- The senders window size corresponds to the number of frames transmitted by not yet received - it varies, grows and shrinks over time
  - The receiver's window size corresponds to the number of frames that the receiver is willing to receive - it too caries over time, and is awalys >= 1

Problem with selective repeat

Suppose that we use 3 bits to represent the sequence numbers (0..7), even if we use a larger integer to store the sequence number in an actual implementation.

Now imagine this happens in order:

1. Sender sends frames 0..6.
2. All arrive correctly into the receiver's window as 0..6, the receiver advances its window to 7,0,1,. . . ,5 and acknowledges the frames.
3. A "small disaster" occurs and no acknowledgements are received.
4. The sender times out and resends frame 0.
5. The receiver gets frame 0, which is within its receiving window and says thanks. The receiver acknowledges for frame 6 as it is still waiting on 7.
6. Sender now sends new frames 7,0,1,. . . ,5.
7. Frame 7 is received and frames 7 and (the duplicated) 0 go off to the network layer. Oops! **The Solution** : make window sizes half the size of the maximum sequence number.

Satellite Broadcasting

As an introduction to Local Area Networking, and multi-access communication, let's take a very simplified look at satellite broadcasting:

- Many users share a single channel.
- Propagation at the speed of light, 300 000 km/sec.
- However, the distance travelled is large - 35,880km for conventional TV satellites, resulting in round trip times of ~270-700msec.
- We can contrast this with the emerging constellations of Low Earth Orbit (LEO) satellites between 200-2000km, with round trip times of ~35msec.
- Bandwidth, typically 500Mbps, is currently 5x higher than typical LAN-based networks because it is less limited by the speed of local infrastructure.
- Cost is the same whatever the distance between sender and receiver. Satellite costs have dropped dramatically in the last few years.
- Satellite acts as a repeater of incoming signals, amplifying and re-broadcasting these signals.
- If two stations broadcast simultaneously the satellite will receive and re-broadcast the sum of these two signals, resulting in garbage. Such simultaneous broadcasts are termed collisions.
- A sender can listen to the re-broadcast of their own packets and determine whether a collision has occurred. Notice that there are no acknowledgements .
  - Users are uncoordinated and can only communicate via the channel.

Advantages

- No data link layer (each sender can verify the correctdness of their own messages (because they can hear them))
- No routing problems in the network layer subnet (no subnet)
- No congestion problems or topology optimization
- Mobile users may be supported

Disadvantages

- Long propagation delay of at least 270msec
- ALl users receive all messages, introducing security problems

Conventional Channel Allocation We can employ two common schemes to share the medium:

Polling

- Either the satellite or a ground station offers the channel to an individual user for a specified amount of time
- Delays of 270msec make this impractical
- Who should be polled?

Frequency and time division multiplexing

- There are two significant forms - frequecy multiplexing allocation and time divison multiplexing allocation
- Using FDMA the channel is divided into N frequency bands (slots) for a maximum of N users. Guard bands are placed between these to limit interface
- Using TDMA the channel is divided into slots based on time intervals, typically 125usec. Each potential User may then use the whole channe; for their time quantum
- Both FDMA and TDMA are very inefficent since the actual number of users, M, is generally <>N and traffic is often “bursty”

Pure ALOHA

Contention based protocol in which

- Users transmit whenever they wish
- Users detect their own collisions
- After a collision, a user “backs off” for a random time period and then retransmits

Expectations from this approach

- Infinite number of users each thinking and sending
- Genertation of packets is a Poisson distribution.

Slotted ALOHA

The slotted Aloha mechanism makes an obvious improvement to improve expected throughput:

- The Satelite generates a timing pulse
- users then only transmit when they get this pulse

This quantization of time reduces the vulnerability period by half (now to a single packet transmission time)

Local Area Networks

- Machines connect to a (logically) single cable within a 1km radius (often the same building or campus).
- Total data rate $>$ 10Mbps (short round trip time, simple data link layer with, say, a one bit sliding window)
- Single organization ownership (no political domains to traverse).
- Usually use broadcasting (no routing problems, but everyone sees all frames).
- One general definition, often cited, is:

  "A LAN is a routerless network, using the same protocol stack for each device, and using only a uniform, local, networking media."

Carrier Sense Networks

- All stations can sense the electircal carrier before sending
- This is possible because of the use of high speed cables over short distances.

  We will be concerned with carrier-sense multiple-access (CSMA) protocols

Factors Affecting CSMA LAN's

1. All frames are of constant (or small, bounded) length.
2. There are no transmission errors, other than those of collisions.
3. There is no capture effect.
4. The random delay after a collision is uniformly distributed and large compared to the frames' transmission time.
5. Frame generation attempts (OLD and NEW) form a Poisson process with mean G frames per time.
6. A station may not transmit and receive simultaneously.
7. Each station can sense the transmission of other stations.
8. Sensing of the channel state can be performed at the same time as transmitting.
9. The propagation delay is small compared to the frames' transmission time, and identical for all stations.

Persistent CSMA Protocols

Using 1-persistent CSMA protocols, each station first sense the activity on the channel.

If two stations A and B are waiting for C to finish, they just pause for a period, and when they sense that the channel is free, transmit with a probability of 1

This results in a good chance of collision.

```
The longer a propagation delay on a LAN the more collisions there will be and, hence, the wo
```

Non- and p-persistent CSMA protocols

In non-persistent and p-persistent protocols, each station may again sense a busy channel.

If the channel is found too busy, contending stations 'back-off' for various intervals.

- Non-persistent stations back-off for random intervals
- p-persistent stations 'jump in' with a probability p (or back-off for a constant amount of time with a probability q = 1-p)

The lower the value of p the fewer collisions there will be.

The Ethernet System

The Ethernet system is a member of the protocols for LAN networks. It uses a 1-persistent carrier-sense multiple-access with collision detection.

Physical properties of Ethernet

Each packet must be at least 64 bytes long to provide some reasonable chance of detecting collisions over long-ish propagation times

Due to power losses within the Ethernet cables each segment cannot exceed 500m, so *repeaters* are used to connect up to 5 segments in a single LAN

Ethernets Contention Algorithm

Each station wanting to transmit, listens to the ether and on finding it silent begins transmission

On detecting a collision a station:

- "backs-off" for a random period which is a multiple of the slot time
- After the first collision each station backs-off for 0 or 1 slot times before trying again, if there is a second collision, a station backs-off for 0,1,2 or 3 slot times

- in general, a station will back off from 0 -2ˆi-1 times after the ith collision. This continues for a maximum of 10 collisions after which the station stayss at 1023 backoffs for 6 more collisions
- After 16 collisions the station considers the 'ether' severed and reports back to the Networking Layer

This method is termed **exponential back-off** and ensures a short delay for each station when a small number of stations collide and a reasonable delay when many stations collide

Ethernet addressing schemes

Each packet contains the source and destination address each of 6 bytes (48 bits)

- If all 48 destination bits are set to 1, the packet is a broadcast address destined for all stations on a LAN
- The high-order bit (bit 47) is used to indicate addressing domains, 0 being for individual addresses and 1 being for multicase addresses (to deliver packets to a group of stations)
- Bit 46 (high order less one) indicates whether the address is for the current LAn or for a more global station on another LAN

Packet Transport Mechanisms

Each station connects to the ether with a transceiver. The must have the following features.

"In particular, failures of the transceiver must not pollute the ether, power failure must not 'cloud' the ether and disconnection must not be noticed by other stations."

802.3 uses five significant mechanisms to reduce the probability and cost of losing a packet. - Carrier detection: Ethernet uses the carrier sense mecahnism of *phase encoding* which guarantees that there is at least one phase transition on the ether during each bit time. - Packet error detection - interference detection: each transceiver has an interference detector: a station can detect interference because what it is receiving is not what it is transmitting. Interference detection has three advantages: 1. A station can detect collisions and re-schedule transmission (no need to wait for a lack of acknolwedgement) 2. Interference is deteced within the propagation time. 3. The frequency of collisions is immediately used to dynamically change the back-off times. - truncated packet filtering: interference detection and deference cause most collisions to result in truncated packets of only a few bits. To reduce the overhead of obviously damaged packets, the hardware is able to filter them out. - Collision consensus enforcement: Whenever a station detects a collision of its own transmission it deliberately jams the ether to ensure that other colliding stations hear the collision as quickly as possible and then to stop transmitting.

Hubs, Switches and Collision domains

hub transmits frame to all other nodes, switch intelligently transmits to the node its meant to go to

A collision domain is the set of devices (potentially) receiving a frame collision.

Wireless LAN protocol

dBm = power ratio in decibel (dB) referenced to one milliwatt (mW). It is an abbreviation for db with respect to 1 mW. eg: 3 dBm means 2 mW, 3 dB means a gain of 2

Hidden Node, Exposed node

Ethernet protocols don't work for wireless because not all nodes are within range of each other

Further problem is that most wireless cards are unable to both transmit and receive at the same time, on the same frequency - they employ half-duplex transmissions.

This means that while collisions do occur, they generally cannot be detected (whilst transmitting)

Collision Avoidance

Unlike their wired counterparts wireless LAN do not apply the Carrier Sense Multiple Access with Collision Detection protocol.

Instead wireless Lan employs collision avoidance to reduce (but not eliminate) the likelihood of collisions occurring. The algo is called Multiple Access with Collision Avoidance (MACA) or CSMA/CA, in which both physical channel; sensing and virtual channel sensing are employed.

The basic idea is that before transmitting data frames, the sender and receiver must first exchange additional control frames before the 'true' data frames. The success or failure of this initial exchange either reserves the medium for communication between A and B, or direct how A, B, and all other listening nodes should act.

Access-Point Association

Although two mobile nodes can communicate directly, the more usual approach is for all communication to be through fixed access-points

We say that a mobile node *associates* with a single access-point if all of its communications are via that point, and all such related nodes form a cell

Communication between nodes in different cells requires two access-points connected via a distribution system.

The mechanism employed by a mobile node to select an access-point is termed *active scanning*

1. The mobile client node sends Probe frames,
2. all access-points within range reply (if they have capacity) with a Probe Response frame,
3. The mobile node selects an access-point and sends an Association Request frame and,
4. The acces-point responds with an association Response frame.

An alternative is for an access-point to use passive scanning:

1. the access-point periodocially sends beacon frames advertising its existence and abilities (e.g supported bandwidths)
2. the mobile node may choose to switch to this new access-point using Dissociation and Reassociation frames.

When a node selects a new access-point, the new access-point is expected to inform the old access-point using the distribution system.

The network layer

The network Layer's responsibility is to get packets from an actual source machine to a destination machine. There may be many hops along the way.

The network layer is thus the lowest OSI layer that has to deal with end-to-end transmission

The network layer must be aware of the immediate topology of its subnet (its neighbours) to make a routing choice that makes progress and avoids introducing congestion on links and avoids already congested links

If the source and destination machine are on different networks (physically || politically) then we must support *internetworking.*

Network layer design objectives

The design objectives of an effective Network Layer are: - to be independent of processor/communication technology - to be independent of the number, type and topology of the subnets, and - to provide a uniform addressing scheme for all hosts in the network

hosts, subnets

Except in the trivial 2 node case some nodes will have more than one physical link to mange

When the network layer presents a *packet* (data or control message) to the data link layer, it must now indicate which physical link is involved.

Each link must have its own Data Link protocol its own buffers for the senders and receiver's windows and state variables such as ackexpected, nextframetosend, etc.

Responsibilities of the Network Layer

The network layer includes nearly all of the functions we consider important in multi-node networks.

Each node can be responsible for everything from handling its own links to managing the network topology

The requirements are of three types

1. Source and destination functions

   end to end protocols, the network layer software interface, fragmentation and reassembly of messages, management of network layer sequence numbers, and creation, manipulation and deletion of headers

2. Store and forward functions:

   choice of the best route for packets, local flow control, and local error control

3. Network-wide management functions

   network flow control, topological awareness and modification, and network performance measurement and monitoring

An example NL responsibility-packet fragmentation and reassembly

Messages may be divided into smaller data units (packets) before transmission. These packets may traverse the network independent;y until they reach the destination node.

This requires fragmenting or segmenting the messages into packets at the source node and then reassembling the packets into messages at the destination

Furthermore, each packet's header must indicate that it is part of some bigger message.

An important consideration for fragmentation is determine the optimal packet size

Compared to variable-sized packets, fixed size packets offer advantages

- Throughout the network, each node's buffer sizze may be fixed.
- It is simpler to prevent congestion at a destination node, since the destination may accurately estimate the number of buffers to pre-allocate
- Fixed-size packets result in simpler memory allocation schemes - typically 'once-only' allocation can be used instead of constantly using 'random-sized' fully dynamic allocation.

The main disadvantage with fixed-size packets is under-utilization of memory when the message size slightly exceeds an integral multiple of the packet size.

Network Layer Head Management

The network Layer data unit, the packet, carries a Network Layer header which must serve many purposes.

The NL header is created in the source node, examined (possibly modified) in intermediate nodes, and removed (stripped of) in the destination node.

The Network Layer headers typically contain:

- The source and destination node addresses
- packet size, if packets may be of more than one size
- MEssage number, and possibly packet number within that message
- Several control bits indicating if the packet is a ‘user-data’ packet or a control packet, whether or not fragmented, fixed or variable length
- Flow control information, such as permission to send additional messages, or flags to change the rate of flow,
- The packet priority

Finally, the data portion (payload) of the packet includes the ‘user-data’ control commands or network-wide statistics

Whereas the data-link layer must acknowledge each frame once it successfully traverses a link, the network layer acknowledges each packet as it arrives at its destination.

The two contending network layer schemes

One community (telecommunications companies) believe that all data should be transmitted reliably by the network layer.

Their proposal is fora connection-based service, termed a virtual circuit. In which:

- Before sending data, the transport layer should establish a ‘semi permanent’ connection between the source and destination

- THe two transport layers then agree (argue) about the type and quality of service that is required and will be available

- communication then occurs between the two parties in duplex fashion along the acquired connection

- the connection is then closed to both parties

- each routing device maintains tables of pairs of source and virtual circuit numbers and desination and virtual circuit numbers the analogy here is telephone conversation. The connection-based service is often called a session-based service.

The other group is the internet community, who base their opinions on 40+ years experience with a practical, working implementation.

They believe that the subnet's job is to transmit bits, and *it is known to be unreliable*

they state that with this connectionless, datagram scheme

- That the hosts must perform any required error processing. Thus error processing is 'relinquished' to the Transport Layer, above.
- that each host operates in a connectionless fashion in which each 'lump' of information is moved between source and destination without a permanent connection between them.
- that each packet moves independently of all previous packets between the same source and destination machines - with each packet possibly taking a different route.

```
the anaology here is with the postal service
```

The result is that the only services provided by the Network Layer to the Transport Layer are to perform the operations of *send packet* and *receive packet* and no negotiation is possible.

Network Layer Routing Algorithms

The routing algorithms are the part of the network software layer that decides on which outgoing line an incoming packet should go.

- For virtual circuits the route is chosen once for each session
- For datagrams the route is chosen once for each packet

Desirable properties for the routing algorithms are similar to those for the whole Network layer:

- Correctness, simplicity, robustness, stability, fairness, optimality.

In particular, robustness often distinguishes between the two main classes of routing algorithms -adaptive and non-adaptive

Robustness is significant for two reasons

1. Once a large network is off and running it is difficult to change the routing algorithm software, and

2. Network topologies often change - hosts, IMP's and lines frequently fail

the better a routing algo can cope with changes in topology, the more robust it is.

The two classes of routing algorithm

Non-adaptive Algorithms

are characterized by the face that the choices of routes between each two hosts are computed in advance and loaded into each host in the whole network before the network is “brought-up’

Non-adaptive choices are often termed ‘static routing’

Adaptive Algorithms

are characterized by their attempts to adjust their route choices based on their current knowledge of the network topology. There are three types of adaptive algorithms.

1. global algorithms use information periodically collected from the whole network (central routing),
2. local algorithms use only the information that each router knows about itself, such as queue lengths and waiting times, and
3. combined algorithms use some of both global and local information (distributed routing).

Adaptive choices are often termed dynamic routing

We know when a host, router or line has failed because:

- either the host tells us that it is about to shutdown, or
- We receive too many consecutive Network Layer timeouts for a destination host or router, or too many Data Link Layer timeouts for a link.

Flooding

a naive non-adaptive routing algo

Initially, every ‘incoming’ packet is retransmitted on every link:

Advantage: flooding gives the shortest packet delay (the algo choose the shortest path because it chooses all paths) and is robust to machine failures.

Improved Flooding Algorithms

Disadvantage: flooding really does *flood* the network with packets! When do they stop?

Partial Solutions:

- Do not retransmit packets on the link on which they arrived - they have already travelled from that direction
- Each packet can include a *hop count* and after any hop count exceed the diameter of the network the packet can be discarded

- As data packets arrive, each router can remember the *link of minimum hop count* on which it arrived - all subsequent packets for that destination go on that link
- router can keep a list of sequence numbers seen and if any packets are re-seen by any router they area discarded

Adaptive Routing - Distance Vector Routing

Distance vector routing algorithms maintain a table in each router of the best known distance to each destination, and the preferred link to that destination

The Count-To-Infinity Problem

Problem with Vector Routing Algorithms

In distance vector routing, routers share how far they are from each destination with their neighbors. If a router goes down, the neighbors might not realize it's down right away. They might hear from another neighbor that there's still a way to get to the downed router, just a bit longer. So, they update their information to show this new, longer path. This incorrect information can keep spreading and getting worse, with distances to the downed router getting longer and longer ("counting to infinity") because the routers keep adding distances based on wrong information.

Adaptive Routing - Link State Routing

When the internet was created everting was the same speed but then when different links became different speeds you can't just count hops anymore. Because one hop on a super slow link may be slower than two hops on two faster links.

Link state routing - each part of the network (each router) starts to keep a detailed map of the network: how each connection (link) is performing, how fast it is, and how it connects to other parts of the network.

e.g each player in the game has a detailed map that doesn't just show the steps but also which paths are faster. Even if they have more steps.

Routers using Link state routing periodically undertake 5 (easy to understand) steps:

1. Discover their neighbours network addresses (send request packets, receive reply packets)
2. Measure the delay or cost to each of its neighbours (immediate delay, or queued delay)
3. Construct link-state packets containing all just learnt
4. Broadcast this packet to all neighbours (use a flooding variant to ensure quick propagation)

5. Compute the shortest path to each other router

Congestion and flow control in the network layer

Additional problem in the network layer occurs if **too many packets are dumped into some part of the subnet** this cases network performance to degrade sharply.

Worse, when performance degrades, timeouts and re-transmission of packets will increase

Also, if all available buffer space in each router is exhausted, then incoming packets will be discarded (what?!?) causing further re-transmissions

Congestion is both a problem that a node's Network layer must avoid, and must address

-**Congestion Control**: is concerened with ensuring that the subnet can carry the offered traffic - a global issue concering all hosts and routers working together

-**Flow control** is concerend with end-to-end control (possibly multiple hops apart)

the sender must not swamp the receiver, and typically involves direct feedback from the receiver to the sender to 'slow down'

Both congestion and flow-control have the same aim - to reduce the offered traffic entering the network when the load is already high

Congestion will be detected through a number of local and global metrics -

- percentage of packets discarded for lack of buffer space
- average router queue lengths
- number of packets timing out requiring retransmission, and
- average (or standard deviation of) packet delay

Disagreement as to whether congestion and flow control are responsibilities of the Network Layer or the Transport Layer

Open Loop Control attempt to present congestion in the first place (good design), rather than correcting it. Virtual circuit systems will perform open loop control by preallocating buffer space for each virtual circuit (VC)

- New VC's will *not* be created via an intermediate router is low on memory
- A new VC may have to be created via another path

Closed loop control maintains a feedback loop of three stages -

- monitoring of local subnet to detect where congestion occurs,
- passing information to where corrective action may be taken, and
- adjustment of local operation to correct problem

End-to-end Flow Control

It is reasonable to expect conditions, or the quality of service, in the network layer to change frequently.

For this reason, the sender typically requests the allocation of buffer space and 'time' in the intermediate nodes and the receiver

Packets are then sent in groups while the known conditions prevail.

Load shedding

If not all demand can be met, some section is deliberately disadvantaged

The approach is to **discard packets** when they cannot all be managed. If there is inadequate router memory (on queues and/or in buffers) then discard some incoming packets.

The network Layer now introduces an unreliable service

Load shedding must be performed under many constraints

- must balance 'reasonable delay' and 'user freedom'
- Maintain fair access to the network for all users.
- respect rights (priviliges) of priority users

A number of reasonable strategies exist for discarding packets:

- consider the priority of each packet (dont drop high priorirty packets). Also need a strategy to encourage sender to send low-prioirt packets
- dont discard ACKs, throw out DATA instead.
- make packets carry a hop counter, don't discard a DATA packet if it has travelled a long way. Instead discard one that has only travelled a short distance.
- examine the *type* of traffic being carried. File transfers must eventually see all frames - all discarded frames will be retransmitted. Discard frames with high sequence numbers, not low numbers. Multimedia data will likely not retransmit old frames, only new frames are of intrest.

Traffic Shaping - Leaky Bucket Algorithm

Consider a leaking bucket with a small hole in the bottom, Water can only leak from the bucket at a fixed rate. Water that cannot fit in the bucket will overflow and be discarded.

The leaky bucket algo enables a fixed amount of data to leave a host (enter a subnet) per unit time. It provides a single server queueing model with a constant service time.

If packets are of fixed sizes. One packet may be transmitted per unit time. If packets are of variable size, a fixed number of bytes may be admitted.

The leaky bucket algorithm enables an application to generate *burst traffic* (high volume, short period) without saturating the network.

Traffic shaping - Token Bucket Algorithm

The leaky bucket algo enforces a strict maximum traffic generation. It is often better to permit *short bursts*, bur thereafter to constrain the traffic to some max.

The *token bucket algo* provides a bucket of permit tokens - before any packet may be transmitted, a token must be consumed. Tokens 'drip' into the bucket at a fixed rate; if the bucket overflows with tokens they are simply discarded (the host does not have enough traffic to transmit)

packets are placed in an 'infinite' queue. Packets may enter the subnet whenever a token is available, else they must wait.

The token bucket algorithm enables a host to transmit in short bursts, effectively 'saving up' limited permission to generate a burst.

Pre-Amble

1. syncro

- the peramble synchronizes the clock of the receiver with that of the sender. This is necessary because it ensures that the receiver correctly times the incomming bits

2. The preamble signlas the start of a frame alerting the receiver that meaningful data is about to be transmitted. THis is important for the receiver to distinguish between idle periods (when no data is being sent) and the beginning of a transmission.

Structure

- usually a 7 byte sequence of alternating 1s and 0s. Followed by the start frame delimiter (SFD) which signals the end of the preamble and start of the frame.

The Requirements of Internetworking

Networks come in differing topologies and speeds no single network configuration suits everyone.

The technology known as internetworking draws the multitudes of networking technologies into a common framework that combines networks into Internets. In general an internetwork must:

- Provide a link between *networks*(at minimum a physical and link control connection)

- Provide Routing and delivery of data between processes on different networks
- Provide an accounting service, keeping track of the status of networks and gateways
- Accommodate the difference between different sub-networks. Of which there are many…

THE TCP/IP Protocol Architecture

The TCP/IP protocol suite is based on the view that communication involves three agents:

- networks (which contain hosts)
- hosts (which run processes)
- processes (which generate and consume data)

Therefore, a network need only be concerned about routing data between hosts as long as the hosts agree how to get data to individual processes.

With this in mind, the TCP/IP architecture organizes protocols into four layers

- Protocols in the network access layer route data between hosts (physically) attached in the same network

- Protocols in the internet layer route data across multiple (possible dissimilar) networks and hosts

Internet layer protocols are implemented in both hosts and gateways - where a gateway is understood to connect two networks and must relay data between both networks using an internet protocol.

The host/host layer contains protocols able to deliver data between processes on the different hosts. Depending on the quality of service and the length of connections required (if any) at this layer, four host-host layer protocols are in frequent use -

- A reliable connection-oriented data protocol providing reliable, sequenced delivery

- A low overhead, minimum functionality datagram protocol

- Reliable datagram protocols, providing low overhead communication for 'bursty' delivery between 'random' endpoints - good for speech and low-definition video streams.

- A real-time streaming protocol, characterized by the need for handling a steady stream with minimum delay variance.

The process/application layer protocols define resource sharing and remove access

Well understood process/application layer protocols include the File Transfer Proton (FTP), HTTP, SMTP, ETC.

Traditional Class-based IP Version 4 Addressing

Each computer or device accessible using Internet (IP) protocols has at least one unique address within its subnet.

If every device was accessible over the global internet, each device would require its own unique address (worldwide); each address consists of a network portion and a local portion. The network 'portions' are assigned by the central DARPA authority.

Internet addresses are described using a dotted decimal notation to describe the 32 bit address

The different (hardware) encoding of 32-bit integers between architectures demands a standard representation for Internet addresses. The internet standard for byte order specifies that integers are sent most significant byte order first (big-endian)

Classless Inter-Domain Routing (CIDR)

With the advent of CIDR, the classful restrictions no longer exist. Address space may be allocated and assigned on bit boundaries, and routers may use one aggregated route (like 194.145.96.0/20) instead of advertising 16 class C addresses

Mapping Internet addresses to Physical Addresses

An obvious question is "What physical address should the sender use to send Internet Datagrams to a specific Internet site?"

In some cases a physical address may fit into a Class A internet address, but more typically an Ethernet address will not fit in the Internet addressing schema.

The address resolution Protocol (ARP) is a special protocol designed to map Internet to physical addresses. When a gateway needs to know the physical address for an Internet address of a host known to be on its network, it broadcasts an ARP frame requesting the physical address. The required host replies; the gateway caches the address for future reference.

The Address Resolution Protocol (ARP)

ARP is a low-level protocol that hides the underlying physical addressing, permitting one or mote Internet addresses to be assigned to each machine. ARP is considered part of the physical network system, not strictly part of the Internet Protocols.

Unlike most protocols, ARP does not have a fixed format. Its design permits it to indicate how big its own fields will be, in this case that Ethernet addresses are 6 bytes (octets) long, and Internet addresses 4 bytes long. This permits ARP to be used with arbitrary network addressing schemes.

When making a request the sender (making the request) also supplies its own Ethernet/Internet address mapping. As all hosts on the Ethernet monitor the broadcast, they can update their mapping tables for future reference (ARP snooping).

Configuration of Network Devices

The standard booting sequence for most OS's involves the computers hard-disk providing a short bootstrap program of several hundred bytes, which in turn reads the operating system true code from nominated blocks on permanent media. To configure its network connection, a client host requires (at least);

- One unique IP address for each of its network interfaces
- The clients hostname
- The address of its default router (where to send packets that we don't explicitly know how to deliver)
- Each interface's subnet mask to determine how many bits of the IP address provide the network and host ids
- The IP address of an initial domain name server, to resolve host names to their IP address

Example config file

```
DEVICE=eth0
BROADCAST=130.95.1.255
NETMASK=255.255.255.0
IPADDR=130.95.1.8
BOOTPROTO=none
GATEWAY=130.95.1.41
GATEWAYDEV=eth0
HOSTNAME=budgie.csse.uwa.edu.au
```

Problems with Static Configuration

- System admins may have to oversee hundreds of machines on a network. Manual maintenance of distinct files becomes intractable
- A single (political) network domain may service many more (dial up or mobile) computers than it has relisted IP addresses. Of course, this scenario hopes that not all computers wish to be connected at once.
- Mobile computers may wish to frequently connect to different (unrelated) networks, and
- Some previously trusted computers may become untrusted, and system administrators may have lost access to modify their config files

A partial solution

The most ‘stable’ attribute in most networking config is the network interface card’s MAC address, such as a card’s 48-bit Ethernet address (many NEW Ethernet cards can change their MAC addresses programmatically)

As with the ARP protocol described earlier a newly bootstrapped client computer can broadcast its Ethernet address via the RARP protocol (reverse address-resolution protocol)

The client broadcasts its RARP request, and any host acting as a RARP server may reply with the client’s allocated IP address.

The bootstrap Protocol (BOOTP)

The Bootstrap Protocol (BOOTP) is a UDP/IP-based protocol that allows a booting host to config itself dynamically, and more significantly, without user supervision. BOOTP is fully defined in, and more significantly, without user supervision. A servers BOOTP response includes several config items, and fits in a single (Ethernet) packet.

It provides a means to assign a host its IP address, a file from which to download a boot program from some server, that server’s address, and (if present) the address of an Internet gateway.

PROBLEM - How can BOOTP use a ‘higher’ protocol from the TCP/IP suite (in this case UDP) if it is trying to determine IP-Level information? The answer lies in the use of special IP addresses within the BOOT request packet.

- To Send the BootP datagram, the client machine sets the destination IP address to the global broadcast (225.225.225.225) and its own IP address to 0.0.0.0
- The responding BOOTP server either replies with an IP broadcast (heard by the requesting client) or responds directly to the client’s MAC address.

Booting over a Network

An additional feature of BOOTP is its support of providing a computer’s (or any “dumber” devices’s) operating system’s image)

- The client may provide a generic string in the TFTP-boot-filename field of its request.
- The BOOTP server may accept this part of the request, and respond with the pathname of an OS’s kernel image available from the server.
- The client may then use this pathname in a subsequent request using the TFTP to fetch its boot image (loading it directly into memory)
- This feature is excellent for maintaining identical operating system kernels, or booting diskless machines.

Dynamic Host Configuration Protocol (DHCP)

TCP's purpose is to enable individual computers on an IP network to extract their configs from a server (the DHCP server)

In general, the servers will have no static information about the individual client computers until information is requested.

The overall purpose of this is to reduce the work necessary to admin a large IP-based network.

The most significant piece of information distributed in this manner is the IP address.

DHCP is based on BOOTP and maintains some backward compatibility. The main difference is that BOOTP was designed for manual pre-config of the host information in a server db. While DHCP allows for dynamic allocation of network addresses and configs to newly attached hosts. Additionally, DHCP allows for recovery and reallocation of network addresses through a leasing mechanism.

DHCP Config

The DHCP daemon (process) is typically provided with the name of a network interface so that it knows from where to accept broadcast requests.

The DHCP daemon reads information from a config file storing the 'public' information for clients. To provide a truly dynamic configuration, we also need to provide a range of IP addresses given to clients.

We can also use DHCP to provide fixed configuration information, based on the Ethernet (MAC) address of the arriving request.

The TCP/IP protocol dependencies

The many Internet protocols naturally depend on each other, that is they demand the services provided by other protocols. For example, the file-transfer protocol (FTP) demands that it operates over a reliable stream protocol (TCP), delivered to a host on a network (IP) which provides flow control (ICMP)

Internet protocol (IP) Datagrams

The internet protocol (IP) provides an unreliable, best-effort, connectionless, packet delivery system.

In this unit we will initially be discussing Internet Protocol version 4

Internet datagrams resemble 'standard' physical-layer frames, but are designed to be encapsulated with the normal network framing schema. Hence, internet datagrams rare said to run on top of traditional networks.

Internet Control Message Protocol (ICMP)

ICMP allows gateways and hosts to exchange bootstrap and error information. Gateways send ICMP datagrams when they cannot deliver a datagram, or to direct hosts to use another gateway. Hosts send ICMP datagrams to test the 'liveness' of their network

As an example, the unix program ping sends ICMP echo messages to a specified machine. Upon receipt of the echo request, the destination returns an ICMP echo reply 'ping' hence both checks that a host is up and that the path to a host is viable.

If a gateway must discard a datagram due to lack of resources it sends a source quench to the datagram's sender. If a datagram cannot be delivered because a host is down or no route exists, a ICMP destination unreachable datagram is generated.

The TCP/IP Protocol suite defines over 25 (in-use) ICMP error message types including:

destination unreachable, time exceeded, parameter problems, source quench, redirection, echo etc. etc.

Port numbers

IP addresses, alone, are not enough as they only address hosts, and not individual operating system processes on those hosts.

From the perspective of any transport protocol, such as TCP (next) each arriving frame is further identified by a 16-bit positive port number that identifies the 'software end-point' to receive the payload.

One role of TCP is to demultiplexed each arriving segment to its corresponding communication end-point, using a port as an index .

Port numbers below 1024 are described as reserved ports, and on OS's with distinct users and privilege levels, elevated privilege (root or admin access) is required to create a software end-point bound to such ports

The file /etc/services on Linux and macOS or `C:\Windows\System32\drivers\etc` On Windows, lists ports commonly used (worldwide), and ports in use for dedicated/local applications.

The Transmission Control Protocol (TCP)

The transmission control protocol (TCP) transforms the 'raw' IP into a full-duplex reliable character stream.

TCP uses a 'well-understood' sliding window with selective-repeat protocol and conveys a number of important fields in its TCP frame header.

What TCP/IP provides to Applications

TCP/IP provides 6 major features:

1. Connection orientation - for two programs to employ TCP/IP, one program must first requests a connection to the destination before communication may proceed.
2. Reliable connection startup - when two applications create a connection, both must agree to the new connection. No packets from previous, or ongoing connections will interfere.
3. Point-to-point communication - each communication session has exactly 2 endpoints
4. full-duplex communication - once established, a single connection may be used for messages in both directions This requires buffering at both each input and output 'end' and enables each application to continue with computation while data is being communicated.
5. Stream Interface - from the application's viewpoint, data is sent and received as a continuous sequence of bytes. Applications need not (but may) communicate using fixed-size records. Data may be written and read in blocks of arbitrary size.
6. Graceful connection shutdown - TCP/IP guarantees to reliably deliver all 'pending' data once a connection is closed by one of the endpoints.

TCP/IP 3-way connection establishment and sequence numbers

A three-way handshake is employed in TCP's internal open sequence.

If machine A wishes to establish a connection with machine B, A transmits the following message:

A->B: SYN, ISN(a)

This inital packet request has the synchronize sequence number bit (SSN) set in its header. and an inital 32-bit unsigned sequence number ISN

B replies with:

B->A: SYN, ISN(b) ,ACK(ISNa)

To provide its own instal sequence number, ISN(b) and to acknowledge ISN(a)

A will finally acknowledge ISN(b) with

A->B: ACK(ISN(B))

TCP/IP Retransmissions

TCP/IP is employed between processes physically tens of centimetres (nano-seconds) apart, as well as processes tens of thousands of kilometers (near a second) apart.

As TCP/IP uses a sliding window protocol, timeouts are employed to force re-transmissions. As there are so many destination hosts, what should be the timeout value?

To cope with the widely varying network delays, TCP maintains a dynamic estimate of the current RTT for each connection. Because RTT's vary tremendously, TCP averages RTT's into a smoothed SRTT that minimizes the effects of unusually short or long RTT's.

SRTT = (a x SRTT) + ((1-a) x RTT)

Where a is a smoothing factor that determines how much weight the new new values are given. When a=1, the new value of RTT is ignored when a=0 all previous values are ignored. Typically a is between 0.8 and 0.9.

The SRTT estimates the average round trip time. TO also allow for queuing and transmission delays, TCP also calculates the mean deviation (MDEV) of the RTT from the measured value.

This is also smoothed.

TCP/IP Congestion Control

Perhaps the most important, and certainly the most studied and 'tinkered with' aspect of TCP/IP is its congestion control.

TCP attempts to avoid congestion collapse by using end-to-end packet loss as the metric of congestion.

- The TCP receiver normally fills the Window field of an acknowledgement header to report how much additional buffer space (the receiver's window size) is available for further data.
- When a message is lost, the TCP sender could naively retransmit enough data to fills the receiver's buffers. Instead, TCP commences by sending a single packet. If an acknowledgement for this single packet returns, the sender next transmits two packets; if all of their acknowledgements return, up to four, and so on. In effect, the protocol grows (doubles) the sender's sliding window until packets are lost; it then restarts at 1.
- TCP/IP responds to congestion by backing-off quickly, and avoids further congestion by slowly increasing offered traffic.

In combination with its closely related slow-start algo for new connections, TCP is is capable of both avoiding and recovering from most congestion.

Network Application Program Interfaces (API's)

Dating back to early operating system implementations, applications attempted to provide a common framework to access both files and devices.

Calls to Unix open() return a file descriptor which is then used in calls to read() and write().

It is preferable if the API to network I/O exhibit the same semantics as file, or steam, I/O but this is difficult for these reasons;

- The typical client-server relationship is not symmetrical - each program must know what role it has to play.
- Network connections may be connection-oriented or connectionless. With a connectionless protocol there is nothing akin to open() since every network I/O operation could be with a different process on a different host.
- Identification is more important to networking than for file operations. Networking applications need to verify peer processes when accepting new connections
- There are more associations to be made for network I/O than for file I/0
- Many file I/O models presume all data is in a continuous data stream; this precludes networking applications working with variable length datagrams.

An Example Network API - Berkeley Sockets

Sockets are a generalization of the original Unix file system model. The most important different is that the OS binds file descriptors, once to files and devices when they are opened. With sockets, applications can specific the destination each time they use the socket.

When sockets were first proposed (1982) it was unclear how significant TCP/IP would become. As a (beneficial) consequence, sockets have been designed to use many different protocols.

The current (kernel) socket implementation consists of three parts:

1. The Socket layer provides the interface between user programs and the networking
2. The protocol layer supports different protocols in use, such as TCP/IP
3. The decide driver supports the physical devices such as Ethernet controllers.

Domain Addressing

Legal combinations of protocols and drivers are specified when the kernel if configured.

For example, sockets that share common communication properties, such as naming conventions and protocol address formats, are grouped into address families.

The Linux file lists all supported address families.

Processes communicate using the client-server paradigm.

A server process listens to a socket, one end of a bidirectional communication path and the client processes communicate with the sever over another socket, the other end of the communication path.

The kernel maintains internal connections and routes data from client to server.

Naming Sockets

When initially created a socket is unbound (it has no addresses associated with it).

Communication cannot occur on an unbound socket - without a name for the process owning the socket, the kernel cannot demultiplex packets to the correct socket. The bind() routine provides an address (a name) to the local end of the socket.

The related call connect() takes the same arguments but binds an address to theremote end of the socket.

For connectionless protocols, such as UDP/IP, the kernel caches the destination address associated with the socket.

Server processes bind address to sockets and 'advertise' their names to identify themselves to clients.

Connection Establishment

Servers accept connections from remote clients and cannot use connect() because they do not (usually) know the address of the remote client until the client has initiated a connection.

Applications use listen() and accept() to perform passive opens.

When a server arranges to accept data over a virtual circuit, the kernel must arrange to queue requests until they can be serviced.

What are Client/Server Software Architectures

Client/server computing is the logical extension of modular programming

It recognizes that not all modules need to be executed within the same memory space

With this architecture, the calling module becomes the *client* (that which requests a service), and the called module becomes the *server* (that which provides the service)

What does a client process do?

The client is a process that sends a message to the server process, requesting that the server perform a service.

What does a server Process do?

Server programs generally receive requests from client programs, executre db retrival and updates, etc.

The server could be the host operating system or network file server, providing file system services and application services.

What is a Three-Tier Architecture

- Introduces (another) server (or an agent) between the client(s) and the traditional server.

The role of the agent is manyfold, it can provide translation services (as in adapting a legacy application on a mainframe to a client/server environment), metering services or intelligent agent services.

What is an Intranet

The explosion of the www is due to the world-wide acceptance of a common transport and server standard and markup language.

Companies figures out that these same technologies can be used for internal client/server applications with the same ease.

Characteristics of Client/Server Architectures

1. A combination of a client or front-end portion that interacts with the user, and a server or back-end portion that interacts with the shared resource.
2. The front-end task and back-end task have fundamentally different requirements for computing resources such as processor speeds, memory, disk speeds and capacities
3. the enviroment is typically heterogenous and multivendo. The hardware platform and OS of a client and server are not usually the same.
4. An important characteristic of client-server systems is scalability

- Horizontal scaling means adding or removing client workstations with only a slight performance impact.
- Vertical scaling means migrating to a larger and faster server machine, or to multiservers.

Partitioning Client/Server Responsibilities

In moving a single, monolithic application to a separated client/server config, we must address a number of issues

1. Is there a functional partition at all?

Are there separate responsibilities that can be performed by separate tasks?

2. Is there a data-driven partition?

Can different sections of the data be centralized, or split between multiple tasks? Can these multiple tasks execute on separate hardware, with separate address-spaces.

3. Is there an extensive use of global variables.
4. Are there any hidden intra-application communication mechanisms (such as variables, exceptions, or signals)?

Concurrency (and hence speed) in Servers

The primary motivation for providing concurrency is speed.

Concurrency is derived from using a ‘non-queuing’ model of execution, either by using a new (copy of the) server to support each client, or to provide faster ‘time-sliced’ response to each client.

If no concurrency is available in the server, pending requests from new and existing clients are either blocked or refused

In general, clients leave their concurrency to the OS, unless the application is sufficiently large, or time-critical, that is the only process on a CPU and it preforms its own internal scheduling.

Increase concurrency, and hence speed, is required when:

- forming responses requires significant I/O
- processing time is proportional to the type of request
- application-specific, high-performance, hardware is available

Definitions:

- Iterative servers - single request at a time
- concurrent servers - multiple ‘simultaneous’ requests

The RPC Execution Order

1. The client calls a local procedure termed the “client stub: It appears to the client that the stub is the actual server procedure that it wants to call. The purpose of the stub is to package up the arguments to the remote procedure, possibly put them in some standard form and then to build one or more network messages (marshalling)

2. The network messages are sent to the local kernel using a system call
3. The network messages are sent to the remote kernel using either a connection based or connectionless protocol
4. The server stub has been waiting on any client's request. It un-marshals the arguments from the network messages and possibly converts them to its own (architecture's) format
5. The server stub executes a local procedure call to invoke the actual server function
6. When the server procedure is finished it returns (normally) to the server stub, returning any required arguments.
7. The server stub converts the return values, if necessary, and build one or more network messages.
8. The messages traverse the network
9. The client stub reads the replies from the local kernel (it has blocked all this time)
10. After possibly converting the return values the client stub returns to the calling procedure; control flow is again in the clients code.

Naming and Interface Binding

How does the client-stub know who to call?

Most operating systems now supporting RPCs use a replicated database used to store server addresses.

When a server restarts (boots) it informs the db that it is alive and passes it:

- The program's program number
- The program's version number
- The port number (On that machine)

Thereafter, the first time a client-stub needs to locate a remote procedure it first asks the db server (the portmapper).

The server maps the procedure's name to the network address.

This process is termed *binding.*

Locating and calling the server

- When we start the server program on the remote machine it creates a UDP socket and binds any local port to that socket. The function `svc_register` in the RPC library is called to register the server with the *portmapper* process. The portmapper keeps trakc of each server's program number, version number and port number.

- We start our client program which calls clnt_create. This function contacts the portmapper on the remote system to determine the UDP port number.
- Our client calls the bin_date_1 function (the client stub). The stub sends the 'call' to the server stub using a UDP datagram. An integer is returned as the single result.
- Our client calls the `str_date_1` function (the client stub). The stub sends the single parameter to the server stub using a UDP datagram. The string result is returned in the parameter array.

Semantics of Remote Procedure Calls

Ideally, Remote Procedure Calls look like local procedure calls and the application programs may be unaware of the existence/need for the network. Like everything else, they suffer from network crashes, lost messages and delays.

Consider what happens when a server crashes. The client stub may:

- Block and await the reply (which will never come)
- Time-out and report a failure, or exception, to the client
- Time-out and retransmit the request

Should clients re-issue their requests in the event of a failure? Moreover, should the client's application (manually) or the client's stubs (automatically) re-issue a request?

It is important to understand whether a package supports at-most-once or at-least-once semantics. Remote operations, which may be repeated without consequence, are termed idempotent operations.

The External Data Representation

(XDR) is a standard for the description and encoding of data

XDR was designed specifically to provide the marshalling and un-marshalling operations for Sun's implementation of RPC

All data which is transferred in RPC is translated using XDR

XDR is also useful in situations which do not use RPC, or even the network, as it allows one to read and write arbitray C data structures in a consistent and well-defined manner.

We can use XDR to save a program's state (i.e data structures) to a file so that when the program is restarted it can resume execution where it left off.

XDR fits into the ISO presentation layer, and is roughly analogous to ISO's abstract Syntax Notation . 1.

The major difference between the two is that ASN. 1 explicity sends typing information along with the data while in XDR this information is implied.

The differences in Data Representation

Machines of different architectures represent data differently internally.

simple example: integers

On Sun-SPARC or Motorola PowerPC architectures integers are stored as byte-0 -> byte-1 -> byte 2 etc.

On Intel x86 processors, integers are stored from byte 2 -> byte 1 -> byte 0

So the Intel 1 on a SPARC or PowerPC would be interpreted as the integer 16777216 (2ˆ24) On Intel.

Other problems occur with respect to alignment and pointers:

- Different alignment schemes will cause the data in structures to be stored differently.
- Pointers have no meaning outside the program where they are used.

The XDR approach

XDR takes a canonical approach to data communication: it defines a standard XDR representation for data and makes its clients use it.

If a program wishes to use XDR to transmit data it must firstly translate its internal representation of the data to the XDR representation

A program receiving XDR information performs the opposite mapping: it converts the incoming XDR data to its own representation.

The advent of a new machine/language has no impact on existing users: the new machine is "taught" to convert between XDR and its own representation and can thereafter communicate with all other XDR users.

The XDR Data Representation

- XDR assumes that bytes (octets) are portable between architectures.
- The representation of all objects requires a multiple of 4 bytes, numbered 0 -> n-1
- The bytes are read from and written to streams such that byte m precedes byte m+1
- If the object being represented is not a multiple of 4 bytes in length then the n bytes are followed by enough 0 bytes to make the total byte count a multiple of 4.
- An unfortunate consequence of this is that sending a single char will involve a 75% waste of bandwidth!

- The standard, however, defines representations for arrays of characters to minimise this wastage in general use.

XDR defines the representation of simple types, and the representation to be used when combining these types to produce more complex types.

The simple types include:

- integers, short and long, signed and unsinged.
- floats, single and double precision
- characters singed and unsigned
- enumerated types and Booleans

These may be combined to produce the complex types.

- fixed and variable length arrays
- strings
- structures
- discriminated unions

XDR also allows some special types:

- fixed and variable length opaque data
- the “void” type

The TCP/IP Overview and Vulnerabilities

The vulnerabilities of each layer

- *application layer* protocols, such as telnet, FTP, HTTP, SMTP, run on (possibly remote) machines to which attackers may not otherwise have physical access. On a case-by-case basis, each of the application services may need to authenticate its remote client, and may use local operating system authentication to perform this, or (dangerously) employ its own mechanism.

  individual applications offering the networked services are themselves also vulnerable - they may have been poorly written (coded), exposing them to attacks which makes them perform in a manner outside of their expected domain.

- *transport layer* protocols, primary provided by the reliable, streaming *transport control protocol* (TCP) and the *user datagram protocol* (UDP) meet the data delivery requirements of most Internet applications.

However, their design introduces vulnerabilities, because applications and operating systems expect the protocols to perform in certain ways. Incorrect interpretation (coding) of protocol RFCCs, or attacks against well known sequences of actions in protocols, makes them perform not as expected, or not at all.

- The internet layer protocols consist of the Internet protocl (IP) and the *internet control message protocol* (ICMP) provide the actual routed delivery of messages between source and destination, and provide only a basic network management function by reporting any observed errors.

IPv4 particularly, is vulnerable to attack and may be exploited to not deliver messages, deliver messages to the wrong destination, or confuse a destination to the extent that it may stop providing any service.

As examples, IP datagrams may be transmitted from one (attacking) host while claiming to be from another, and forged ICMP messages may make a destination network or host appear unreachable.

As examples, IP datagrams may be transmitted from one (attacking) host while clamming to be from another, and forged ICMP messages may make a destination network or host appear unreachable.

- *Physical layer* protocls are not strictly part of the TCP/IP suite, but define how packets or frames are received via hardware, and provided to the IP (software) layer above. by its nature, interface hardware must see all packets destined for, or passing by, the interface, and most hardware may be configured by software (the operating system) to repost all activity seen.

Trivially, on a shared network, an operating system (and probably some of its programs) may capture all packets that are visible on a network.

Packet Sniffing

Most networks consist of lots of computers that are connected via a *shared* local network. Sharing means that computers can receive information that was intended for other machines.

To capture the information traversing the network is called *sniffing*

Ethernet works by transmitting addressed packets via a shared cable. The Ethernet network interface card (NIC) in the intended destination computer sees all packets, but on seeing one with the NIC's unique 48-bit address, the NIc will copy the entire packet to the OS software for analysis and eventual delivery to application programs.

There are two main problems with Ethernet's approach:

- most Ethernet NIC's can be placed in *promiscuous mode*, which results in all observed packets being sent to the OS. Many rootkits will replace the ifconfig program (an abbreviation for interface configuration) to avoid the simple detection of interfaces in promiscuous mode.
- and, most Ethernet NIC's permit their NIC address to be modified, programmatically, and so one Ethernet NIC could (accidentally or deliberately) be given the MAC address of another .

TCP/IP port scanning

Using *port scanning* an attacker tries to identify, which services are supported from a potential target host. Whenever an active port is located, an attacker may attempt to further determine the version number of any active server/service.

What information does an attacker learn from port scanning?

- If an attacker learns that a port is open, they can actually connect to the detected port.
- If an attacker learns that a port is closed, they learn that no service is listening to that port.
- additionally, some scanning software reports ports as filtered, indicating that a connected attempt was terminated with a RESET or timed out.

e.g the naive TCP connect Scan completes the TCP three-way-handshake. A SYN packet is sent to the system and if a SYN/ACK packet is received, it is assumed that the port on the system is active. If a RST/ACK packet is received, it is assumed that the port on the system is not active.

Attackers may further attempt to hide their scans by:

- scanning through the ports very slowly, and not in numerical order. unless its a very quiet system they wont be detected.
- perform hundreds of scans simultaneously from hundereds of random/spoofed IP addresses. The target host will know they are being scanned, but now know from where.

Stealth scanning

Stealth scanning involved searching for open ports, but without actually creating a connection.

Half-Open scanning only performs the first part of the TCP/IP handshake. It sends a SYN flag and awaits a reply - a reply with the SYN flag set reports an open port, with the RST flag set reports an inactive port. Half-open scanning is favoured by potential attackers because nothing is logged.

Internet Protocol (IP) spoofing

The *spoofing* of IP packets allows an intruder on the Internet to impersonate a local system's IP address.

Is possible because programs can open raw sockets, create and send malformed IP packets.

An attacker uses source address spoofing for two reasons

- to gain access to resources that only accept requests from specific source addresses, or
- to hide the source of an attack by directing the blame at others

UDP packet spoofing

UDP is a light weight protocol that achieves extra performance from IP by not implementing some of the session-based features a more heavyweight protocol (like TCP) offers and tyipically sees twice the throughput.

Specifically;

- UDP allows individual packets to be dropped (with no retries)
- packets may be received in a different order than sent, and
- applications using UDP, typically do not establish a protocl-level session with their peers. Each request and reply pair are often independent.

An attacker could see UDP's lack of connection and session establishment an exploit these properties without worrying about responses. This is useful in scenarios where the attacker does not care about receiving reply packets.

e.g flood attack or ddos they dont care about responses.

TCP/IP Sequence Number Attacks

TCP/IP establishes sessions between endpoints like so

A three-way handshake is employed in the TCP *open* sequence

If machine A wishes to establish a connection with machine B, A transmits the following message:

A->B : SYN, ISN(a)

This inital packet request has the synchronize sequence number bit (SSN) in its header, and an inital 32-bit unsigned sequence number ISN(a).

B replies with:

B->A: SYN, ISN(b), ACK(ISNa)

To provide its own initial sequence number, ISN(b) and to acknowledge ISN(a).

A will finally acknowledge ISN(b) with

A->B: ACK(ISN(b))

And the connection is established.

This session establishment is considered secure, provided that the initial sequence numbers are so random that they cannot be guessed.

Traditional BSD-derived implementations only change the 2nd byte of the sequence number every second, and each new connection changes it by 64. An attacker, having established a valid connection, is able to 'guess' the next number to be used.

A series of well known attacks exploit the non-randomness of the initial sequence numbers.

The attacker, c, establishes a valid connection with B, thus determining one of B's 'current' values for ISN(b). The attacked, c, now impersonates A by sending a packet to B, but by sending A's NIC address in the Ethernet packet:

C(as A)-> B: SYN, ISN(c)

B replies with

B->A: SYN, ISN(b), ACK(ISN(c))

to the true machien A. c will probbaly not see this message B->A, but can guess the value of ISN(b) c now sends

C(as A)->B ACK(ISN(b))

and b belives that it has a valid connection with A. But A is confused as to why it received B->A and may choose to either ignore it or inform B (with a RESET packet) that something is wrong.

If A chooses to ignore the packet then C can continue to send packets to B assuming A's identity. If c can see all replies from B->A in the session, then C can fully masquerade as A, while A ignores the transmissions of which it is not a part.

Denial of Service (DOS) attacks

Examples:

- attempts to flood a network, thereby preventing or delaying legitimate network traffic.
- attempts to disrupt connections between two machines, thereby preventing access to a service
- attempts to prevent a particular individual from accessing a service, and
- attempts to disrupt service to a specific system or person

Often, the source address of these packets is spoofed, making it difficult to locate the real source of the attack.

Smurf DDoS attacks

In the smurf DDoS attack, the attacker provides a spoofed source address, when sending an ICMP echo, or *ping*, to an IP broadcast address as the destination

- The attacker sends ICMP Echo Request packets where the source IP address has been forged to be that of the target of the attack.
- The attacker sends these ICMP datagrams to addresses of remote LAN's broadcast addresses, using so-called directed broadcast addresses, These datagrams are thus broadcast on the LAN's by the connected router.

- All hosts which are alive on the LAN each pick up a copy of the ICMP Echo Request datagram, and sends an ICMP Echo Reply datagram back to what they think is the source.
- The attacker can use larger packets to increase the effectiveness of the attack.

The use of broadcast addresses for protocol attacks is termed *amplification*

The *smurf* attack has 3 types of victims.

- the single destination victim of the attack
- a network abused (temporarily) to amplify the attack, and
- (always) the host harboring the attacker.

One way to defeat smurfing is to disable IP broadcast addressing at each internal network router, however this violates the requirements for routers.

The SYN-Flood Attack

Also known as the half-open attack

- the attacker (client) sends a SYN request to the server
- The server records the request on a queue of connections waiting to complete, replies with a SYN/ACK packet, and eagerly awaits the final ACK reply
- However, the attacker does not send the ACK reply. Instead, the attacked sends another, actually hundreds of, SYN requests with different source forged address.

To avoid SYN-flood attacks modern OS's will now not employ a large number of half-open sockets for new connections.

Distributed DDos attacks

Also known as a *packet storm attack* an attacker will flood a single system with "junk" packets to consume bandwidth - preventing legitimate packets getting through.

Pretty much attackers get hundreds of remote controlled attack servers and then they all use source spoofing to attack a single target.

Security at Network Boundaries

The greatest opportunity to an attacker is provided when they connect to the LAN from the wider internet.

Attacks from the Internet can attempt to bypass the user- or system-level security of a single machine, or possibly undertake a dos attack on the LAN itself.

In general, we wish to develop security practices at the boundary between a LAN and the wider internet, to constrain the types of network traffic that may cross the boundary.

Specifically we would like to.

- Control network traffic based on both senders' and receivers' network (IP) address
- control network traffic based on requested services (IP Ports)
- Not expose our Lan topology to the wider-Internet, hiding host-names, addresses, and available services.
- Constrain some network traffic based on its content
- Only permit internal access from remote users and services, based on their verified identities and (possibly) location
- log all internet connections, attempts and sus traffic

Packet Filtering at network boundaries

Firewall - any network device, appliance or specifically configured computer which protects the boundary of an internal network.

Specifically, we shall describe firewalls as software devices through which all network packets must pass, both incoming and outgoing.

Providing a single ingress point to the internal network clearly provides a single opportunity to apply a consistent policy to network traffic.

two practices that circumvent the purpose or effectiveness of having a firewall.

End-runs: with which a computer can access the Internet without passing its traffic through the firewall

Traffic tunneling: with which users or applications can embed certain types of unwanted network traffic within permitted protocols

Depending on the provided bandwidth a firewall may be;

- Part of a traditional, single, workstation (protecting itself)
- a computer device protecting several other workstations
- a dedicated device doing nothing else but protecting other hosts

A traditional computer acting as a firewall must inspect each packet entering and leaving the internal network via a number of different network interfaces

Possible packet filtering criteria

Network packets may be *filtered* on a number of criteria

By examining the headers of TCP/IP traffic we can detect obviously falsified traffic:

- Filter on each IP packet's source address. Packets which arrive on a network interface connected to the outside of our internal network (i.e the Internet) and announcae their source address as being from the internal network, probably have spoofed source addresses.
- Filter on each IP packets destination address. Packets destined for an internal network address should not leave the network via an external interface.
- Filter based on specific low-level routing or transport protocols, such as denying all ICMP or UDP traffic from leaving.
- Filter based on application protocols, such as permitting HTTP and FTP requests to leave, but not permitting NFS mount requests to enter, and
- Filter based on recent activity, Stateful filtering, has knowledge of recent traffic; for example; stateful FTP filtering permits incoming FTP data-connection request, only if a corresponding outgoing control-connection already exists .

Developing a Firewall Policy

The establishment of a firewall policy simplifies the practice of deciding what traffic to permit and what to filter.

Moreover, a consistent, and consistently applied, policy is a strong argument by system administrators to deny individual requests for new small holes in the firewall by individuals.

Surprisingly the "firewall community"is divided on default behaviours. Either:

- That which is not expressly forbidden is permitted, or
- That which is not expressly permitted is forbidden

Because someone could launch an attack from your network and not just to it. You have to be sure that packets leaving and packets coming in are both filtered.

Packet filtering with iptables

Consider the 'lifetime' of a single packet as it enters and traverses a firewall

The packet could have originated on the firewall host (from a locally running program) and be destined for another host. iptables filters these packets using its OUTPUT chain of rules before they are retransmitted via an outgoing network interface.

The packet could have originated from outside the firewall host, and be destined for processes on the firewall host.

iptables filters these packets using its INPUT chain of rules as soon as they arrive via one of the firewall's incoming network interfaces, or.

the packet could have originated from outside of the firewall host, and be destined for another host. Iptables filters these packets using its FORWARD chain of rules as soon as the packet arrives via an incoming interface, and before it is retransmitted on an outgoing interface.

Of note, this generic approach permits the iptables software to act on a single workstation with a single network interface (such as an ADSL router link) protecting itself, or as a specific firewall device with several (Ethernet) network interfaces protecting a whole internal network

IP Masquerading

IP masquerading or *network address translation* (NAT) is a tekkers employed within a firewall, or border gateway, to translate, or map, one set of IP addresses (usually private) to another (usually public)

To use NAT, the firewall connecting the internal LAN to the external Internet will have (at least) two network cards, each with their own IP address:

- on the Internet side, the machine will use a fully-routable address assigned by an ISP
- On the Lan side, it will have an address from the non-routable addresses, defined in RFS

The primary motivations for using NAT are;

- your network provider may only provide you with a single IP address to use - NAT permits multiple hosts to use the same IP address,
- It simplified the later growth and re-design of a network, and
- external attackers cannot (easily) learn the topology of your internal network unless they penetrate your firewall.

An Example of IP Masquerading

Consider the following example: Machine blue (with a single Ethernet interface, and IP address 192.168.3.10) generates a packet, from its port 400, destined for server.com.

When the packet arrives at the NAT-enabled firewall (on its internal Ethernet interface, IP address 192.168.3.1), the firewall will de-encapsulate the packet, and rewrite it so that it appears to have now originated from the firewall itself (with IP address 200.33.1.1, and a currently unused port on the firewall, 1430). The packet is finally forwarded on the external Ethernet interface.

When a reply is received from server.com the destination IP address will be 200.33.1.1, port 1430. The firewall's mapping table is consulted to reverse the translation, changing the IP address to 192.168.3.10 (for blue), port 400

Network Address Translation (NAT)

NAT, as described in RFC1631, has many forms

- overloaded NAT - maps multiple unroutable IP addresses to a single registered (routable) IP address by using different ports (as just seen). This is variously known as PAT (Port Address Translation), single address NAT or port-level multiplexed NAT.
- dynamic NAT - maps an unroutable IP address to one of a managed group of registered IP addresses, and
- static NAT - maps an unroutable IP address to a registered IP address on a one-to-one basis. This is required when a device needs to be accessibly from outside the network, such as a web- or FTP-server.

Connection Tracking

Connection tracking refers to the ability for a firewall to maintain state information about connections - source and destination IP address and port number pairs.

Firewalls able to do this are termed *stateful*. Stateful firewalling is more secure.

Exchanging Encryption Keys

Diffie-Merkle-Hellman Key exchange

Allows two active participants to agree on a new temporary session key with which they will exchange a message.

Moreover, anyone eavesdropping on their agreement discussion, will not be able to further eavesdrop on the message exchange.

A simple (physical) analogy of how keys can be exchanged:

- A wants to send a key to B
- A puts the key in a secure box and locks it with A's padlock
- B does not have the key to A's padlock, so instead,
- B receives the box and adds B's own padlock to the box and returns it to A.
- A removes A's padlock with A's own key and sends the box back to B.
- B can now remove B's own padlock and remove the key which is now shared by A and B

Public key Cryptography

Using *public key encryption* we use two keys rather than just one

- The *public* key E, may be openly published
- The *private* key, D is known only by the intended recipient.

The plan is to choose eys such that even knowing the public key does not reveal the private key:

- A and B openly publish their public keys (viewed as algorithms) E(a) and E(b)
- A sends E(b) to B
- B calculates D(b) = plaintext(message)
- B can then reply with E(a) for A to read.

The MIT/RSA algorithm

We choose two very large prime numbers, p and q, each over 100 digits. We define EA to be the pair (e,n) where n = pxq (for p, q being 100 digit primes, n will typically at least 200 decimal digits).

We define DA to be the pair (d,n) where (e x d) mod ((p-1) x (q-1)) = 1

We then use: Encryption function : C := P(e) mod n Decryption function : P := C(d) mod n

Asymmetric ciphers

RSA is an example of an asymmetric cipher, employing different keys for encryption and decryption. The relationship between keys simplifies an attack.

RSA has been ubiquitous.

Keys for asymmetric ciphers need to be longer than keys for symmetric ciphers to achieve similar resistance to brute-force attacks:

Strong Encryption is not enough - the need for Digital Signatures

A digital signature cannot be a constant; it must be a function of the document that it signs.

A digital signature prevents two types of fraud -

- the forging of a signature by the receiver (or any third party), and
- the repudiation of the transmission of a message by the sender.

Two categories of digital signature are identified:

- True signatures, signed by the sender, verified by the receiver.
- Arbitrated signature may only be sent and verified through a trusted third party. The recipient is unable to verify the sender's signature directly, but is assured of its validity through the mediation of the arbitrator.

Message Digests - Basic building blocks

A message digest is 16-, 20-, 32 byte "fingerprint" of a message

Message digests are central to digital signatures. When a message is signed, its contents are first hashed to give a message digest. The digest is then encrypted with the sender's secret key giving proof of the sender's identity.

A good digest must have the properties:

- An absence of collisions. Unlike simpler file checksums, which quickly demonstrate file or data integrity, it must be *hard* to find two messages with the same digest.
- Must not be invertible. Digests are deterministic many-to-one functions
- A uniform distribution of results, A change in just one input bit should affect at least half the output bits.

Simple changes to even a single byte (even a single bit) should result in dramatic changes to the digest:

Digital signature generation

Digital certificates are different

A digital signature is a 'summary' of the original essage, but also provides an assurance that the original creator of the signature has the private key matching the public key used to generate the signature.

Digital certificates have been loosely described as the *driver's licence for the internet*

A digital certificate provides a binding between an entity's public key, and one or more attributes to it's identity.

- An *entity* may be a person, an executing piece of software, or a device such as a router or a smart-card
- A certification authority (CA) attests to the authenticity of the entity's public key by digitally signing a message with its own private key.
- The 'quality' of the certificate depends on the detail of information provided to the CA
- Either, public *and* private keys may be issues by the CA, or the CA may challenge the entity's public key.

The successful use of digital certificate appears within a large community - little is gained by issuing one's own.

Browser support for digital certificates

Digital certificates are managed by all common browsers.

The browser will display the digital certificate from the current page.

- The subject of the certificate,
- The issuer (CA) of the certificate,

- The period of validity of the certificate
- The message digest of the certificate.

Certificate Path validation

Ca's are organized in hierarchies - each parent CA signs a certificate vouching for a subordinate CA's public key.

When validating a chain of certificates, the *certificate path*, the path is followed until the top of the chain is reached (when?).

There is no automated way of verifying the top of a certificate chain other than verifying that it is one of a list of directly known (and implicitly trusted) certificates (such as in a browser)

Certificate Revocation lists

A certificate revocation list (CRL) allows clients and servers to check whether the entity they are dealing with has a valid certificate.

Trust breaks down, and CRLS's are required when

- a subjects private key is exposed.
- a CA's private key is exposed,
- the relationship between the subject and CA changes

Certificate revocation works by:

- obtain the subject's digital cert and verify its validity.
- Extract the serial number of the cert
- Fetch the current CRL from the CA
- Verify the CRL's digital structure, and record its publication time and when the next CRL is to be published.
- Examine the CRL to determine if the intended certificate has been revoked or suspended (based on the certificate serial number)
- Alert the user if the certificate is revoked.

Limits of certificate revocation

In a large public key infrastructure community, CRL's are both large and must be downloaded frequently.

Applications can be significantly slowed by the need to retrieve the latest CRL from a heavily taxed directory server (or other distribution point)

There exists a compromise between always being up-to-date, versus the risk of false certificate acceptance.