Network systems

Notes on the lectures

Week 5

Routing continued

There are two major algorithms for routing: **link state** and **distance vector**.

Distance Vector algorithm

Every node in the network announces their “name” (MAC / ID) and link costs to their neighbors. The link costs indicate what nodes they can receive, and at what cost. If a node receives such an announcement, and finds that a node has a better connection to a node than the connection it stored in its forwarding table, it updates the forwarding table. After a few iterations, and assuming the network doesn’t change or fail, this will end up with every node being aware of the cheapest path to every other reachable node in the network. It’s an implementation of Dijkstra’s algorithm.

A weakness of this algorithm is that it becomes difficult to find a solution if a node becomes unreachable, or if a link fails. Possible solutions, assuming A ← 1 → B ← 1 → C ← × → Z:

- Split horizon: if node A routes node Z via neighbor B, it doesn’t tell B.
- Split horizon with poison reverse: if node A routes node Z via B, it tells B that the cost is infinite.
- Send entire path in routing updates: works, but it’s less efficient.
- Hold down: postpone good news, until bad news has had a chance to spread.

In general, the link state system is faster and more robust than distant vector, but very inefficient.

Routing in the Internet

The Internet is divided into Autonomous Systems (AS):

- Stub AS: only one connection to the rest (university, small ISP).
- Multihomed AS: more than 1 connection to the rest, but it doesn’t carry traffic for others.
- Transit AS: carries traffic for other ASs.

An AS is identified with a 16- or 32-bit number. Every AS can use its own preferred protocol for routing. Routing among ASs is done using the Border Gateway Protocol (BGP). Possible routing protocols for an AS include:

- Static routing
- Bridges / switches with (rapid) spanning tree protocol
- Routing Information Protocol (RIP), based on the distance vector algorithm. Nodes exchange their routing tables every 30 seconds, with a maximum hop count of 15.
- Open Shortest Path First (OSPF), based on the link state algorithm. Divides the AS’s network further into routing areas.

Border Gateway Protocol

The BGP is based on the distance vector algorithm. It doesn't just advertise the distance (cost), but also the complete path information, as a list of AS numbers. This helps to detect loops. Administrators may install policies, for example to prevent paths through some ASs.

In practice, the BGP consists of ASs of certain tiers. Providers get money for the traffic, peers neither receive or pay money, and customers give money for the traffic. The latter is preferred over the former.

Multicast

In unicast, one packet from one source is destined for one recipient. However, certain applications, like software updates or livestreams, require identical packets to be sent to multiple destinations. This is multicast. The trivial, naive implementation of multicast, is to simply send separate copies of the packet to each destination, but that's very inefficient.

Efficient multicast on a LAN is trivial; all packets reach all hosts anyway, so sending one packet with a special address is sufficient. ARP is an example of such an implementation.

To enable multicast in a network, among multiple LANs, a more complex solution is required. In such a solution, the source should only have to send its packet once, after which the routers should replicate it to the intended recipients. The source may send the intended destinations in the packet header, but this is very inefficient, and in cases like IPTV, the sender might not even know all the destinations.

Another solution is to send the packet to a single, specific destination address, which identifies the group of recipients. Routers keep track of which hosts in their network are part of what groups, and copy the packets accordingly. In IPv4, class D addresses are used to identify these groups. In IPv6, the address block FF00::/8 is used.

Hosts can tell their routers that they're interested in a group using the Internet Group Management Protocol (IGMP). In the rest of the network, multiple network-layer multicast routing algorithms are possible.

Reverse path forwarding

In reverse path forwarding, paths from every node to the source of the multicast stream form a spanning tree, routed at the source. The tree is pruned by non-interested nodes telling their upstream neighbors. This is inefficient: it's an opt-out system, meaning all the data reaches all the nodes before they can indicate that they're actually not interested. That's why, for example, such a system wouldn't work on a wireless network, as a lot of bandwidth would be wasted on sending the data to mobile devices that aren't interested.

PIM-SM (Protocol Indicated Multicast – Sparse Mode)

PIM-SM uses a rendez-vous point, to which the source can send its data, and where the interested nodes can subscribe to the data from the source. The nodes may send a source-specific join request to the source, to create a source-specific tree.

Other solutions

- MSDP (Multicast Source Discovery Protocol): Extensions for sources in multiple domains.

- PIM-SSM: Source-Specific Multicast: there is no rendez-vous point, only a source-specific tree. This works well for multicast sessions with a single source, like TV shows.
- BIDIR-PIM (BI-DIRectional tree): a multicast tree is created, where the source is simply another member. This fits well to multicast sessions with many senders, like conferencing.

In the Internet, most connections don't have multicast routing. However, lots of internal networks use it, like VCK distributing IPTV on the campus, and KPN distributing IPTV to their customers using a separate VLAN.

Mobility

If devices in a network move, errors may occur. If a host moves from one network to another, it will get a new IP address, and lose its existing TCP connections. If a bunch of nodes are known to move in and out of each other's radio range, but still want to remain connected, ad-hoc networking is required, with a special routing algorithm that supports it.

In the naive solution, where a moved host gets a new IP address, existing connections are lost. Another solution would be to install a specific route for this host in the network forwarding tables, but that's not scalable.

In ad-hoc networks, nodes are mobile, come and go, and the connectivity changes all the time. Usual routing algorithms as discussed before will not work, as too much data will be exchanged if the changes are frequent. The solution is to only exchange routing information when needed, on demand.

In such a system, the source broadcasts a Route Request (RREQ). A node replies to the request if it is the destination, or if it has a fresh enough route to the destination. Otherwise, it rebroadcasts the RREQ, and updates its route table with information about the source. Loops are prevented using a unique identifier in every request. The response is forwarded to the source using routes learned from the RREQ, and the nodes along the path record the route to the destination in their route table.

If the source moves, it can simply restart network discovery. If the destination, or a node on the path moves, it sends a RREP with infinite cost, to remove outdated routes. If a node moves that is not on the path, nothing happens. RREQs and RREPs have sequence numbers to ensure that the latest is used.

Week 6

End-to-end protocols

End-to-end protocols provide logical communication between processes running on different hosts.

- They guarantee message delivery;
- They deliver messages in the same order that they were sent;
- They deliver at most one copy of each message;
- They support arbitrarily large messages;
- They support synchronization between the sender and the receiver;
- They allow the receiver to apply flow control to the sender;
- They support multiple processes of themselves on each host;

The network on which the transport protocol will operate will occasionally drop messages, reorder messages, deliver duplicates, limit the message size or deliver messages after a long delay. End-to-end protocols try to fix this.

There are multiple types of end-to-end services and protocols.

Simple (de)multiplexer / datagram service

A service that tries to send a stream of bytes, while retaining integrity, but not guaranteeing that the bytestream arrives completely. The “multiplexer” part indicates that the service provides a way to send multiple streams at a time. An example is UDP.

UDP (User Datagram Protocol) is a very simple protocol, which provides a “best effort” service. UDP segments may be lost, or delivered out-of-order. However, it is connectionless, meaning it is stateless, and requires no connection establishment, making it faster. Furthermore, the UDP header size is small, and it does not have congestion control, enforcing no limit at the speed of UDP.

A UDP header contains the source and destination port, the length of the datagram and a checksum to verify integrity.

Reliable bytestream service

A service that provides a platform to reliably send a stream of bytes. An example is TCP.

TCP (Transmission Control Protocol) is a full-duplex service that offers reliability, multiplexing, flow control and congestion control. The protocol is connection oriented.

A TCP header contains the source and destination port, a sequence number, an acknowledgement number, the length of the header, some flags, the advertised window, a checksum for integrity, and some other fields. Available flags are SYNchronize, FINish, ReSeT, PUSH, URGent and ACKnowledge.

The sequence number is a counter that indicates how far in the transfer the host is. The acknowledgement number indicates what sequence number is expected next from the other host. For example, if a host sends 80 bytes after 120 bytes already have been transferred, the sequence number would be 81, and the acknowledgement number is 201.

To establish a connection in TCP, a three-way handshake algorithm is used. The client sends a SYN datagram to the server, with a certain sequence number. Then, the server sends back a SYN+ACK datagram, with a certain sequence number, and the previous sequence number + 1 as the acknowledgement number. Finally, the client sends an ACK datagram, with the previous sequence number + 1 as the acknowledgement number.

A TCP host can be in many states, indicating if it’s ready to receive data, sending data, closing its connection, et cetera.

TCP has a hard limit by design for its speed. A TCP/IP packet may “live” for 120 seconds, and a sequence number has 32 bits. Because of this, it’s only possible to send 2^{32} bytes per 120 seconds, which is about 286 Mbit/s. Another problem with TCP is that the window may be too small for longer links, which prevents having more than a certain amount of unacked data.

TCP has some extensions, which are indicates as options in the header.

- Timestamp: Every sent packet contains a timestamp, which is echoed by the ack, and may be used to accurately measure the RTT.
- PAWS (Protection Against Wrapped Sequence number): Used the Timestamp option to distinguish old from new segments with the same sequence number, eliminating the hard limit of 286 Mbit/s.
- Window scaling: Hosts agree that the window size is no longer in bytes, but in multiples of 2^k bytes, where k is negotiated.
- SACK (Selective ACKnowledgement): Hosts acknowledge data that has been received out of sequence.

Both TCP and UDP use port numbers. Some of these are “well known”, meaning that you can expect a certain service to run on a certain port. An example is HTTP, which runs on TCP port 80 by default, or DNA, which runs on UDP port 53 by default.

Request/reply service

A service where there isn’t a stream, but only a request and a reply. An example is RPC (Remote Procedure Call).

Real-time service

A service which ensures real-time communication, like RTP. Used in applications like media broadcasting and conferencing, where timing is an important factor.

Internetworking continued

Didn’t summarize this, never will, as I actually got a 6 for the third test :^]