Routing in Fixed Internet Protocol Networks

Janne Lindqvist
Researcher
janne.lindqvist@tml.hut.fi

September 28, 2006

Abstract

This work complements the routing chapters presented in Internetworking with TCP/IP Volume I by Douglas Comer [3]. In particular, we will take a closer look at distance vector routing and link state routing and routing algorithms. Some of the topics are discussed only lightly and included for reference.

1 Introduction to Routing

Routing in Internet Protocol networks consists of discovering routers and deciding to which router packets are sent. Routing can be categorized in many ways. The most general categorizations are static and dynamic routing. In this work, we will mostly discuss dynamic routing. Dynamic routing adapts to changes in the network with routing protocols, static routing is configured by hand. We will look at examples of the categories below.

A common analogy used when discussing routing is the postal service. Routing *is* in many ways similar to processes of delivering letters and postal packets. One key difference is that in postal services, the address contains very much information compared to IP addresses. When looking at an address in a letter, you can tell many things about the intended recipient, for example, where he or she lives. Consequently, a unicast IP address is an interface identifier and a locator. However, the address does not identify the interface securely, thus addresses can be forged. Additionally, the locator function does not give information about where the final recipient actually resides. On the other hand, a multicast address is a group identifier, it does not have the locator functionality at all. Addressing is an important topic in routing but not further discussed in this work. For addressing issues, we refer to the textbook by Douglas E. Comer, Internetworking with TCP/IP, Volume I [3]. Next, we describe a key part of routing by giving an example of a routing table and its relevance to routing protocols.

1.1 Routing Tables

A routing table consists of at least 3-tuples: (address, next hop or gateway, interface). The address could be a network address or a host address. Usually the routing tables contain also other data. Next, I explain details from a reduced routing table printed from Mac OS X.

```
Tietokone-Janne-Lindqvist:~ janne$ netstat -r
Routing tables

Internet:
Destination        Gateway            Flags    Refs   Use  Netif
default            192.168.1.1        UGSc        4   1    en1
127                localhost          UCS         0   0    lo0
192.168.1.101      localhost          UHS         0   2    lo0
```

The most interesting part is in the first line of the table, it gives the default route. If the destination address of the packet does not match other addresses in the routing table, the default route is used. For example, consider a packet destined to www.tkk.fi (IP: 130.233.220.31). In the above configuration, the packet is sent to 192.168.1.1 through Ethernet interface “en1”. You might notice that the address 192.168.1.1 is a so called private network address, that is, an address which is not routed in the Internet [12]. The address belongs to the router’s local area network interface. The router also does Network Address Port Translation (NAPT) [14]. The router has an another interface with a routable address or so called public IP address. Morever, the router has a default route, it sends all packets not destined to itself or the local area network to the the Internet Service Provider’s router. This configuration is depicted in Figure 1.

The above explanation is a typical example of static routing, which is quite common today in households with an Internet connection. More can de done with static routing, but when the network and routing tables grow, routing tables become error prone and not efficient. Also, when we go farther in the Internet router topology, there are the core network routers which have no default routes at all. When static routing is not enough or possible, we need dynamic routing. Dynamic routing is handled by routing protocols.

An important issue which cannot be overemphasized is that routing protocols do not forward packets. Routing protocols alter the routing table according to the information received from the network and the other routers. The forwarding is a

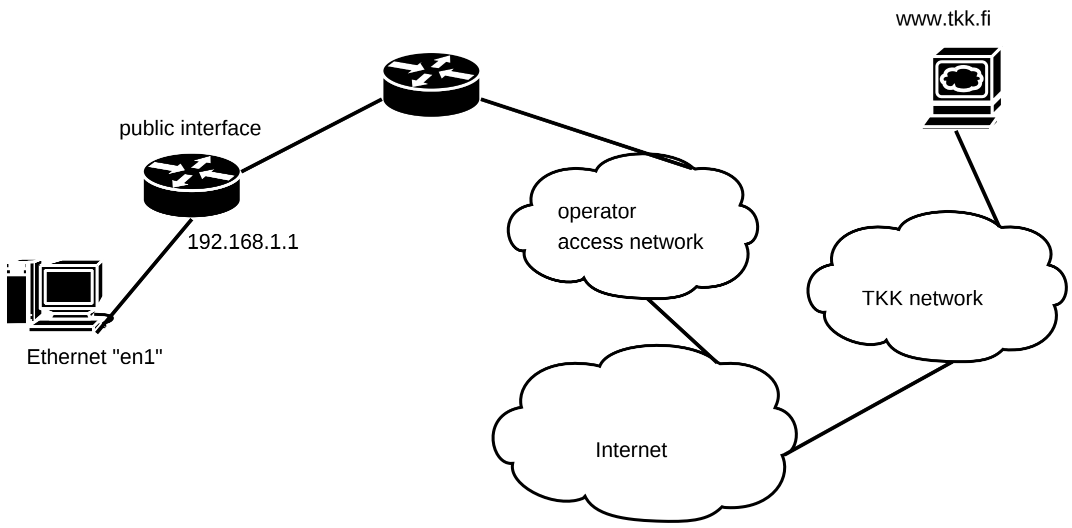

Figure 1: Household Internet Connection

separate process, which involves checking the routing table and deciding to which interface the packet is sent.

1.2 Routing Protocols

IP routing protocols are usually categorized to distance vector protocols and link state protocols. Also, in the literature an additional term, path vector protocol, is used [6]. Distance vector protocols are simple whereas link state protocols are usually complex. Path vector protocols are something between distance vector and link state protocols.

Another category for routing protocols depends on what part of the network the protocols are used in: exterior gateway and interior gateway protocols. Exterior gateway protocols work between autonomous systems (AS) and interior gateway protocols work inside an autonomous system. An autonomous system is defined as a system that has autonomy and authority for all the decisions within the system. Every autonomous system has a unique identification number: an AS number. The AS numbers are assigned by designated authorities, for example, by RIPE in Europe.

The emphasis in this work is on routing protocols in fixed IP networks. It should be noted that ad hoc networking has many interesting research issues regarding routing. For a tutorial in ad hoc networking and routing protocols, we refer to the Ad hoc networking book by Charles E. Perkins [10]. One important difference between fixed network and ad hoc network routing is that in ad hoc networks you are not able to use perimeter defence as we will see later on the routing security section. Ad hoc network routing protocols have a different variety of approaches and can be classified differently than above.

Also, the protocols covered in this work are designed for unicast routing. Multicast routing is not covered.

1.3 Routing Metrics

Distance vector and link state routing protocols use shortest path algorithms, which are described later, to decide on the preferable route. The metric is formulated as a (link) cost or distance for the algorithm.

The metric is chosen to portray a given property of the network. Thus, the metric could be hop count, that is, how many routers have to be traversed before the destination is reached. Other used metrics are 1/capacity, packet delay, and congestion. To understand, why 1/capacity can be used, we refer you to first check the intuition part of Section 2.1.4. We want to prefer the links that have the highest capacity. Thus, the greater capacity a link has, the smaller the cost should be. Similarly, but just the opposite, the more delay or congestion the link has, the more higher should be the cost.

1.4 Protocols and Algorithms

It is important to identify the difference between routing algorithms and protocols. The protocols describe how information about the network is passed to routers, and the algorithms do the counting of the shortest path based on the information. In routing literature and in many scientific publications, the terms protocol and algorithm are used vaguely, thus care is needed.

On the other hand, to demonstrate what a particular algorithm does, we have to assume some properties that are actually part of a protocol. For this reason, we develope an imaginary distance vector protocol in the next section.

2 Distance Vector Routing

The name distance vector routing comes from the fact that distance vector protocols are used between routers to exchange tuples (D,V). The D is the distance (link cost) and the V is the vector (node or router). With (D,V), a router tells another router that “I have this cost to this router”.

Distance vector routing protocols are based on the distributed Bellman-Ford algorithm or a derivation of it. The original non-distributed algorithm was developed separately by Richard Bellman and Lester Ford, Jr. Bellman used techniques from dynamic programming to solve the problem how to decide on a route between cities. The work was published in a journal article called “On a routing problem” [1] long before computer networks as we know them existed.

The original algorithm is classified as a single-source shortest path algorithm, and the distributed version is an all-pairs shortest path algorithm [4].

2.1 Distributed Bellman-Ford Algorithm

The routing problem can be formalized as a graph theoretic problem. We define an undirected graph as $G = (V,E)$, where V is a finite nonempty set and $E \subseteq V \times V$. The V is a set of vertices (nodes) v and the E is a set of edges (links) e. A *path* is a sequence of consecutive edges in a graph that are traversed to reach v_2 from v_1. A cost is a positive integer value assigned to each edge. The problem can now be stated as the question: from vertice v_1, which edges can be traversed to reach v_2 so that the path is shortest. The shortest path is defined as the smallest total cost of the edges. Remember, that as in Section 1.3 the cost is in routing protocols based on a metric that usually is based on a property of the network. Note, that by following the above definition, we assume symmetric bidirectional links for the computer networks. This is the usual case in fixed IP networks if the metric for deriving the link cost is for example delay. However, in wireless networks, the links are usually not symmetric and may not be birectional.

The following formal description of the algorithm is based on the textbook Communication Networks by Leon-Garcia and Widjaja [8]. A slightly different asynchronous version and a proof is presented in Data Networks by Bertsekas and Gallagher [2].

2.1.1 Definitions

First, some definitions. From now on, we will refer to the edges of the graph as links and the vertices as nodes.

D_j current estimate of the minimum cost from node j to the destination node
C_{ij} link cost from node i to node j
$C_{ii} = 0$ (the link cost to the node itself is zero)
$C_{ik} = \infty$, if nodes i and k are not directly connected

2.1.2 Initialization phase

$$D_i = \infty, \forall i \neq d$$

$D_d = 0$

2.1.3 Update Phase

$D_i = \overset{min}{j}\{C_{ij} + D_j\}, \forall j \neq i$

Repeat the update phase until no more changes occur in the iteration.

2.1.4 Intuition

To understand why the algorithm solves the problem described in the beginning of Section 2.1 the following intuition is given. The basic idea is that if a node C is on the shortest path from A to B, then the path from A to C is a shortest path. Moreover, the path from node C to node B is also a shortest path.

In the initialization phase, that is, when the actual router boots up and the routing protocol is started, we put the distance to the node itself *d* to zero and the distance to all other nodes we mark as infinity.

The update phase consists of counting the costs of paths and choosing the path that has the smallest cost.

2.2 A Distance Vector Protocol

We now describe an imaginary distance vector routing protocol. For simplicity, we assume that a router in the network is able to know what other routers it is connected to and what are the link costs to these neighbor router. Also, a router is able to detect if a link becomes unavailable. We will discuss in the link state routing section how these can be established, but for now, assume the properties as given.

Similarly, as described in Section 1.1 the router keeps a routing table, which the protocol updates. The 3-tuple consists of (N,D,SP), where N denotes the node where destination D is available, and SP is the known shortest path to the destination D. N can be thought of a combination of the next hop and interface sets of the routing table in section 1.1 and D can be thought of as the address part.

The running of the protocol consists of sending synchronized update messages to all other adjacent nodes (neighbors). The update messages consist of the information (D,SP), that is, “so far I know that I can reach destination D and the cost reach to reach it is SP”. As described earlier, this is the distance vector exchange, the SP is the distance and the D is the vector.

2.2.1 Example of Route Discovery

In the following example, we denote a link with ”-” and a node with a number. The link cost for every link is 1. We are considering a situation where the node 4 becomes online. The routing tables are formed as follows: the sources are listed in the second row and the (N,SP) pairs are in the following rows. The situation is depicted in Figure 2.

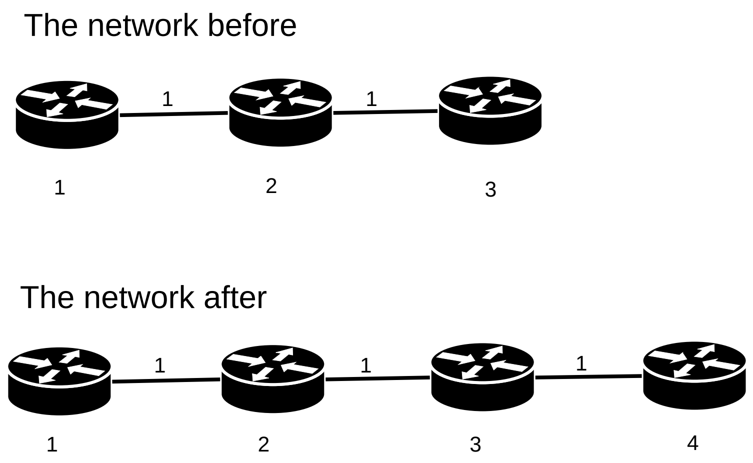

Figure 2: Network in Route Discovery

Partial routing tables for the nodes

event	**source**		
	1	2	3
bootup	$(-1,\infty)$	$(-1,\infty)$	$(-1,\infty)$
1. update	$(-1,\infty)$	$(-1,\infty)$	$(4,1)$
2. update	$(-1,\infty)$	$(3,2)$	$(4,1)$
3. update	$(2,3)$	$(3,2)$	$(4,1)$

First, the routers are booted up and for demonstration purposes they mark the routes in the table to all destinations as $(-1,\infty)$ to regard that no destination can be reached. Of course, a real router does not know during boot up what other routers exist in the network, so it would not make a routing table at all during boot.

During the first update, all routers gather knowledge of their neighbors. Router 3 has router 4 as its neighbor, thus it can update the routing table. During the second update, router 3 informs its neighbors of the routes that it has and router 2 receives a route to router 4. Similarly, router 1 receives the information during the third update.

2.3 Counting to Infinity

In this subsection, we present an example how the count-to-infinity problem can occur when using distance vector routing protocols. Note that all routing protocols have to have some mechanism to attack the problem. For example, in link-state protocols and new protocols proposed for ad hoc network routing, sequence numbers are used to battle the count-to-infinity problem.

Now, consider the network described in the previous section. We have reached the situation where the counting has stopped and we have the routes for node 4. Then, suppose that the link between nodes 3 and 4 disconnects (Figure 3). What happens now is that node 3 does not receive any traffic from the link 3-4, but node 2 on the link 2-3 advertises (4,2), that is, “I have a route to node 4 which has the cost 2”. Thus, node 3 updates its routing table accordingly. During the second update, node two receives the information from all its neighbors (1 and 3) that they have routes to the node 4 with cost 3. It randomly chooses one of the routers and updates its routing table. Thus, the routers would in theory be updating the routing tables and counting to infinity.

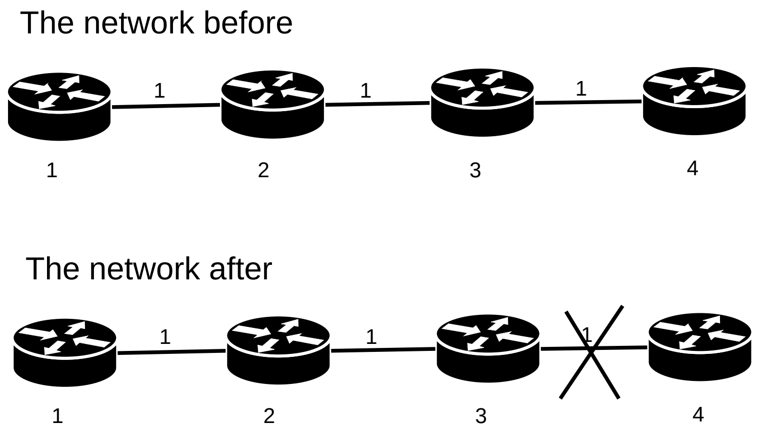

Figure 3: Count to Infinity Example

Partial routing tables for the nodes

event	**source**		
	1	2	3
before disconnection	$(2,3)$	$(3,2)$	$(3,1)$
1. update	$(2,3)$	$(3,2)$	$(2,3)$
2. update	$(2,3)$	$(3,4)$	$(2,3)$
3. update	$(2,5)$	$(3,4)$	$(2,5)$
4. update	$(2,5)$	$(1,6)$	$(2,5)$

3 Routing Information Protocol

The Routing Information Protocol (RIP) [5] [9] is a distance vector routing protocol for interior gateway routing. RIP uses hop count as its metric. “Infinity” is set to 16, which means that RIP can support networks where every destination is maximally 15 hops away.

To battle the count-to-infinity problem, RIP uses two techniques in addition to the preset maximum count. These techniques are split horizon with poisoned reverse and triggered updates.

We will next present how split horizon with poisoned reverse can handle the count-to-infinity problem described in the previous example (Figure 3). It should be noted that split horizon with poisoned reverse works only with links with two routers, if the same link has three or more interconnected routers, it will not work. The idea is that when sending the routing information to neighbors that are on the path to the destination node, the distance is set to infinity.

Partial routing tables for the nodes

event	**source**		
	1	2	3
before disconnection	$(2,3)$	$(3,2)$	$(3,1)$
1. update	$(2,2)$	$(2,2)$	$(-1,16)$
2. update	$(2,2)$	$(-1,16)$	$(-1,16)$
3. update	$(-1,16)$	$(-1,16)$	$(-1,16)$

During the first update, node three receives nothing from the link 3-4 and from the link 2-3 it receives that the distance is infinity (16) because the route from node 2 to 4 travels through 3. Thus, it updates the routing table accordingly. During the second update, node two receives from both its neighbors that the distance to node 4 is infinity. And finally, node 1 receives on the third update that the distance to node 4 is infinity.

RIP version 1 [5] is considered historical by the IETF. RIP version 2 [9] has added features: support for subnetting and routing messages can be authenticated.

In addition, routing information is not broadcasted, it can be delivered to all RIP routers using multicasting.

4 Link State Routing

Whereas distance vector routing protocols are usually simple, link state routing protocols are complex. Based on Tanenbaum [15] link state routing protocols can be divided to the subprotocols described below.

4.1 Neighbor Discovery Subprotocol

First, when a router is booted, it tries to discover its neighbors. The router broadcasts a HELLO message to all its network interfaces. All other routers that receive the HELLO message are required to reply immediately. The HELLO message includes a globally unique name, for example, the IP address of the router, so that routers can distinguish between different routers.

4.2 Measuring Cost to Neighbors Subprotocol

The router has received HELLO replies and is now aware of its neighbors. Now it needs to measure the link cost to them. This is done with an ECHO message. For example Internet Control Message Protocol (ICMP) [11] could be used. The router sends an ECHO to all its neighbors, and the neighbors immediately try to respond with an ECHO REPLY. Time between sending an ECHO and receiving an ECHO REPLY can be calculated and used as a link cost. More complex metrics can also be used.

4.3 Sending the Learned Data Subprotocol

Now, the router has data of its neighbors and the link cost to them. Next, it needs to send the data to all other routers in the network. This is done with an INFORM message. The INFORM message consists of 3-tuples (N, C, SEQ), where N is the neighbors unique name, C is the link cost, and SEQ is a sequence number. The sequence number is increased whenever a new INFORM message is sent.

The INFORM messages are flooded to the network. The simplest form of flooding is that when a router receives a packet from one interface, it sends the packet to all other interfaces. This can cause that the packets never cease to propagate in the network, if a Time-to-Live field in a lower layer protocol is not used. Next, we present a simple algorithm, which can prevent the flooding problem.

4.3.1 Algorithm for Handling INFORM Messages

All routers in the network have a table in which they record the received INFORM messages. Note that this is not the routing table. When an INFORM message is received, a router refers to the table and compares the sequence numbers. If the sequence number in the packets is

- **greater** than in the table, then insert the new data in to the table and forward the INFORM message
- **smaller or the same** as in the table, then discard the received data.

This works, if the routers are never rebooted. After reboot, a router cannot know what the last sequence number it used was. This problem can be solved by including an age field in the table. After a certain time has passed, if no new (sequence number greater) INFORM messages have been received, the particular record is removed from the table. The INFORM messages should also include a Time-to-Live field to ensure that the message is at some point of propagation discarded. Next, we describe how all these subprotocols are used to finding the routes and to update the routing table.

4.4 Dijkstra's Algorithm

Now, based on the data above, the routes can be calculated. We present Dijktra's single source shortest path algorithm according to Leon-Garcia and Widjaja [8].

4.4.1 Definitions

set N consists of the nodes for which shortest paths have been counted
s is the source node
D_i the *current* minimum cost from source node s to node i
C_{ij} the link cost from node i to node j

4.4.2 Initialization phase

$N = \{s\}$
$D_j = C_{sj}, \forall j \neq s$
$D_s = 0$

4.4.3 Next node to process phase

For $i \notin N$ such that
$D_i = \overset{min}{j \notin N} D_j$
Add i to N.
If all nodes are now in N, STOP.

4.4.4 Update phase

$D_i = min\{D_j, D_i + C_{ij}\}, \forall j \notin N$

GOTO "Next node to process phase"

4.4.5 Intuition

We now describe the intuition behind Dijkstra's algorithm. First, in the initialization phase, we mark the distance to the node itself to zero. We include it to the set N. For all other nodes, we mark the current minimum cost as the link cost if the node is a neighbor.

In the **next node to process phase**, we choose a node that has the smallest current minimum cost assigned and include the node to N. If there are multiple nodes with the same smallest cost, a node is chosen randomly. If the set N now contains all the nodes, we stop because we have determined all of the shortest paths.

Updating the values to the routing table consists of choosing the smallest cost between the value (D_j) in the routing table and the value counted by adding the cost to the node chosen in the previous phase and the link cost ($D_i + C_{ij}$).

4.5 Example of Route Discovery

We consider a situation where the link state routing protocol has discovered the network topology. In our example, the source node is node 1 and it computes the shortest paths to all other nodes in the network. The link cost is 4. The network is depicted in Figure 4.

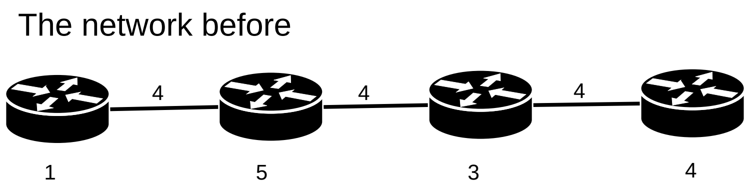

Figure 4: Network in Dijkstra Example

Full routing table for node 1

event		**destinations**		
	N	5	3	2
discovered	$\{1\}$	4	∞	∞
1. Next node phase	$\{1,5\}$	4	8	∞
2. Next node phase	$\{1,5,3\}$	4	8	12
3. Next node phase	$\{1,5,3,2\}$	4	8	12

The first and second next node phases are followed by the update phase which is reflected in the routing table. However, the third next node phase is not followed by the update phase, since the algorithm stops when the node 2 is added to N.

5 Routing Security

In security, we must always identify what we are trying to protect and from whom and the fundamental assumptions of the system. The routing protocols described earlier could be attacked easily. *Any* malicious host in the network could inform the routers that it has the best routes to all other routers. The routers try to choose the best routes, thus they would update their routing tables to route all packets through the malicious node. The malicious node could then continue with a number of attacks including eavesdropping or just dropping all the packets.

To protect the network from above described malicious behavior, the routing protocols have authentication mechanisms. The authentication mechanisms in the simplest form use a shared secret to verify that the message came from a trusted node. Messages only from trusted nodes are considered when updating the routing tables. The assumption is that the trusted nodes will not behave maliciously. Of course, if a trusted node is compromised, then authentication does not help.

Despite the security features in routing protocols, interior gateway routing can be disrupted fairly easily by a malicious user inside the network. Additionally, usually cryptographic authentication cannot be used due to performance issues [7]. On the other hand, exterior gateway routing is protected by traffic filtering on the border of the network (ingress filtering), and when properly configured exterior gateway routing is mostly vulnerable to denial of service attacks by flooding.

An example attack against routing in a local area network can be established with Address Resolution Protocol (ARP) spoofing. The malicious host in the LAN uses a modified implementation of the ARP protocol. The implementation responds to ARP queries as if it were the router. ARP has no mechanism to verify the validity of the messages. Thus, the ARP tables of hosts in the network point to the malicious host. Then, all traffic is routed to the malicious host. It can then reroute the traffic to the actual router and eavesdrop the traffic, for example.

Routing security also affects the Internet infrastructure security. BGP is used to block Denial of Service attacks against hosts, not just routers. For example, RFC 3882 suggests using "black holes" – designated parts of a network for directing traffic when under denial of service attack [16].

6 Alternatives to Routing

In this section, we will discuss the alternatives to routing in fixed IP networks. We do not always have to use routers to establish internetworks, that is, networks connected to other networks.

Bridging can be a interesting alternative to routing in some situations. For example, consider a corporate network which has old 10 Mb/s Ethernet in use in one part of the offices and the somewhat newer 100 Mb/s Ethernet in use in the other part. Instead of routing between the networks, you could establish a transparent bridge.

Bridging can be an alternative for home networks, too. Linux and other free operating systems can be set up as bridges. Microsoft Windows XP can be used as a transparent bridge.

The downside of bridging is that it is not usable for large and many heterogenous network interconnections. There are no clear rules of thumb, the decision when to bridge or route is the art of network engineering. However, it can be said that network bridges have had a recent rebirth. Wireless Local Area Networks (WLAN) based access networks use bridging to enable mobility. The WLAN access points are configured as bridges to act as a single network for upper layer protocols.

7 Other Routing Paradigms

Distance vector and link state routing can today be considered as classical Internet routing paradigms. A modern alternative approach is Multiprotocol Label

Switching (MPLS) [13]. Multiprotocol Label Switching borrows ideas from network switches and applies them to routing.

In a MPLS network, the packet forwarding is done based on *tags*. A border router assigns a tag for a packet. Thus, the routing decision is made on the border router. The core routers in the network only forward packets. In contrast, in classical routing, every packet is inspected by every router for making the routing decision. The result of the architecture should be that the core routers work more efficiently when the routing decisions are already made on the border of the network. The border routers can make the decision efficiently because they have inherently lesser processing load compared to the core routers. The tradeoff is that the tags have to be propagated to the network.

MPLS can be used to implement a Quality of Service (QoS) architecture called Differentiated Services. It is not a QoS architecture as such. The MPLS architecture and QoS are covered in more detail on the Computer Networks II course.

8 Acknowledgments

I thank Sanna Suoranta, Anton Alstes and Timo Kiravuo for their comments.

References

[1] BELLMAN, R. On a routing problem. *Quarterly of Applied Mathematics*, 16 (1958).

[2] BERTSEKAS, D., AND GALLAGHER, R. *Data Networks, Second Edition*. Prentice Hall, 1992. ISBN: 0-13-200916-1.

[3] COMER, D. E. *Internetworking with TCP/IP, Volume I: Principles, Protocols, and Architecture, Fourth Edition*. Prentice Hall International, 2000. ISBN: 0-13-018380-6.

[4] CORMEN, T. H., LEISERSON, C. E., AND RIVEST, R. L. *Introduction to Algorithms*. The MIT Press, Cambridge, Massachusetts, 2000. ISBN: 0-262-53091-0.

[5] HENDRICK, C. RFC 1058: Routing Information Protocol, June 1988. Status: Historical.

[6] HUITEMA, C. *Routing in the Internet, 2nd Edition*. Prentice Hall, Upper Saddle River, 2000. ISBN: 0-13-022647-5.

[7] KIRAVUO, T. Personal communication.

[8] LEON-GARCIA, A., AND WIDJAJA, I. *Communication Networks, Fundamental Concepts and Key Architectures*. McGraw-Hill Higher Education, Singapore, International Editions, 2001. ISBN 0-07-022839-6.

[9] MALKIN, G. RFC 2453: RIP Version 2, Nov. 1998. Status: Standard.

[10] PERKINS, C. E. *Ad hoc networking*. Addison-Wesley, 2001. ISBN: 0-201-30976-9.

[11] POSTEL, J. RFC 792: Internet Control Message Protocol, Sept. 1981. Status: Standard.

[12] REKHTER, Y., MOSKOWITZ, B., KARRENBERG, D., DE GROOT, G., AND LEAR, E. RFC 1918: Address Allocation for Private Internets, Feb. 1996. Status: Best Current Practice.

[13] ROSEN, E. C., VISWANATHAN, A., AND CALLON, R. Multiprotocol Label Switching Architecture, Jan. 2001. Status: Proposed Standard.

[14] SRISURESH, P., AND EGEVANG, K. B. RFC 3022: Traditional IP Network Address Translator (Traditional NAT), January 2001. Status: Informational.

[15] TANENBAUM, A. S. *Computer Networks, Third Edition*. Prentice Hall International, Upper Saddle River, 1996. ISBN: 0-13-394248-1.

[16] TURK, D. RFC 3882: Configuring BGP to Block Denial-of-Service Attacks, Sept. 2004. Status: Informational.