TÉCNICA MPLS PARA OPTIMIZACIÓN DE TRÁFICO EN REDES IP
ANEXO A

CAROLINA ANDREA CARRASCAL REYES
CLAUDIA XIMENA MOSQUERA LEYTON

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
GRUPO DE I+D EN NUEVAS TECNOLOGÍAS EN TELECOMUNICACIONES
POPAYÁN
2001

TÉCNICA MPLS PARA OPTIMIZACIÓN DE TRÁFICO EN REDES IP
ANEXO A

CAROLINA ANDREA CARRASCAL REYES
CLAUDIA XIMENA MOSQUERA LEYTON

Monografía para optar al título de
Ingeniero en Electrónica y Telecomunicaciones

Director
Ing. Esp. OSCAR J. CALDERÓN CORTÉS

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERIA ELECTRÓNICA Y TELECOMUNICACIONES
GRUPO DE I+D EN NUEVAS TECNOLOGÍAS EN TELECOMUNICACIONES
POPAYÁN
2001

TABLA DE CONTENIDO
ANEXO A – ENRUTAMIENTO EN REDES DE PAQUETES

PARTE 1. ALGORITMO DIJKSTRA 1

PARTE 2. MODEL FOR ROUTE DISCOVERY AND SELECTION IN PACKET SWITCHED NETWORKS .. 5

1. INTRODUCTION 5
1.1 Routing Protocols 5
1.2 The Internet as a Model 6
1.3 The Routing Problem 7
1.4 Genetic Algorithms 8
1.5 Applying Genetic Methods to the Routing Problem 9

2. PROBLEM DEFINITION 11
2.1 The Network 11
2.2 Constraints and Metrics 11
2.3 Flows 11
2.4 Route Table 12
2.5 Utilization 12
2.6 Average Delay and Loss Probability 13
2.7 Optimization Problem 14

3. RELATED RESEARCH 14
3.1 Routing 14
3.1.1 Static Metrics 15
3.1.2 Dynamic Metrics 15
3.1.3 Random Routing - No Metrics 16
3.2 Genetic and Evolutionary Algorithms 17
3.2.1 Encodings 18
3.2.2 Initial Population 18
3.2.3 Selection 18
3.2.4 Reproduction 19
3.2.5 Mutation 19
3.2.6 Applications 19
3.3 Genetic Routing 20
3.3.1 GBR 20
3.4 A New Approach 21

4. PROPOSED SOLUTION 22
4.1 Parasites 22
4.2 Packet Destination 22
4.3 Packet Forwarding 23
4.4 Feedback 23
4.5 Routing Loops 24
4.6 Population Control 24
4.7 Packet Loss 25
4.8 Overhead 25
4.9 Summary 25

PARTE 3. PROTOCOLOS DE ENRUTAMIENTO IP 27

1. INTRODUCCION 27
1.1. Open Shortest Path First (OSPF) 29
1.1.1. Background 29
1.1.2. Routing Hierarchy 30
1.1.3. SPF Algorithm 32
1.1.4. Packet Format 32
1.1.5. Additional OSPF Features 33

2. BORDER GATEWAY PROTOCOL (BGP) 34
1.2. Background 34
1.3. BGP Operation 35
1.4. BGP Routing 36
1.5. BGP Message Types 36
1.6. BGP Packet Formats 37
1.6.1. Header Format 37
1.6.1.1. BGP Packet-Header Fields 37
1.6.2. Open Message Format 37
1.6.2.1. BGP Open Message Fields 38
1.6.3. Update Message Format 38
1.6.3.1. BGP Update Message Fields 39
1.6.4. Notification Message Format 39
1.6.4.1. BGP Notification Message Fields 40

ÍNDICE DE FIGURAS
ANEXO A – ENRUTAMIENTO EN REDES DE PAQUETES

Figura 1.1. Topología de la Red 1
Figura 1.2. Cálculos de Costo de Ruta más Corta 2
Figura 1.3. Cálculos de Costo de Ruta del Camino más Corto para IS2 e IS3 3

Figure 2.1. Model of a Simple Network 5
Figure 2.2. Router R2's Initial View of the Network 6
Figure 2.3. Router R2's Least-Cost Topology 7
Figure 2.4. Model of a Transit Network 9

Figure 3.1. Autonomous Systems 27
Figure 3.2. An OSPF AS Consists of Multiple Areas Linked by Routers 31
Figure 3.3. OSPF packets consist of Nine Fields 32
Figure 3.4. Core Routers can use BGP to Route Traffic between Autonomous Systems 34
Figure 3.5. In pass-through Autonomous System Routing, BGP pairs with another intra-Autonomous System-Routing Protocol 35
Figure 3.6. A BGP Packet Header Consists of Four Fields 37
Figure 3.7. A BGP Open Message Consist of Six Fields 38
Figure 3.8. A BGP Update Message Consist of Five Fields 39

INDICE DE TABLAS

Table 2.1. Sample Link Metrics for the Network Shown in Figure 2.1. 6
Table 3.1. OSPF Metrics. 29

ANEXO A – ENRUTAMIENTO EN REDES DE PAQUETES

PARTE 1. ALGORITMO DIJKSTRA

La mejor manera de describir éste algoritmo es considerar una red específica como se muestra en la Figura 1.1. Los enlaces entre cada Sistema Intermedio (IS – Intermedium System) se muestran como circuitos punto a punto, cada uno con un costo asociado que puede variar dependiendo de la métrica. Para cada métrica, la tarea consiste en encontrar el trayecto a través de la red desde un IS de origen a cada uno de los otros IS de la red, que tenga el menor costo acumulativo. Los distintos pasos del algoritmo se representan en las partes (a) a (g) de la Figura 1.2.

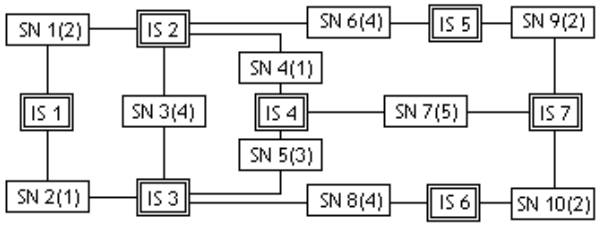


Figura 1.1. Topología de la Red.

Se supone a IS 1 como el origen. Cada uno de los demás IS tiene asociada una medida (que se indica entre paréntesis) de la distancia acumulativa desde ese IS hasta el IS de origen a través del IS indicado. Así, una entrada (2,IS 4) significa que el costo del camino de regreso a IS 1 es de dos unidades a través de IS 4. En un principio, los costos de camino a todos los IS que no están conectados directamente al origen son desconocidos y por tanto se les asignará un costo de camino infinito, es decir, el costo más alto posible. Además, mientras no se sepa que un valor de costo es el mínimo, será *tentativo* y el cuadro del IS tendrá un contorno hueco. Una vez que se conozca el costo de camino mas corto desde un IS hasta el origen, será *permanente* y el cuadro de IS aparece con borde negro.

Al comenzar, como IS 1 es el origen, aparece con borde negro. Los costos de camino a los dos IS conectados directamente (vecinos - IS 2 e IS 3) se dan como iguales a sus costos de camino de enlace respectivos. Así, IS 2 tiene la entrada (2, IS 1), que indica que el costo de volver a IS 1 directamente es 2, mientras que IS 3 tiene una entrada de (1,IS 1). Todos los demás costos siguen teniendo el valor máximo. A continuación, se escoge el IS con menor costo entre *todos* los IS que todavía son tentativos. Desde luego, el valor de costo mínimo corresponde a IS 3, cuyo

costo es 1. Éste se marca ahora como permanente y se calcula el nuevo conjunto de valores de costo acumulativo desde IS 3, mismo que se muestra en la parte (b) de la Figura 1.2.

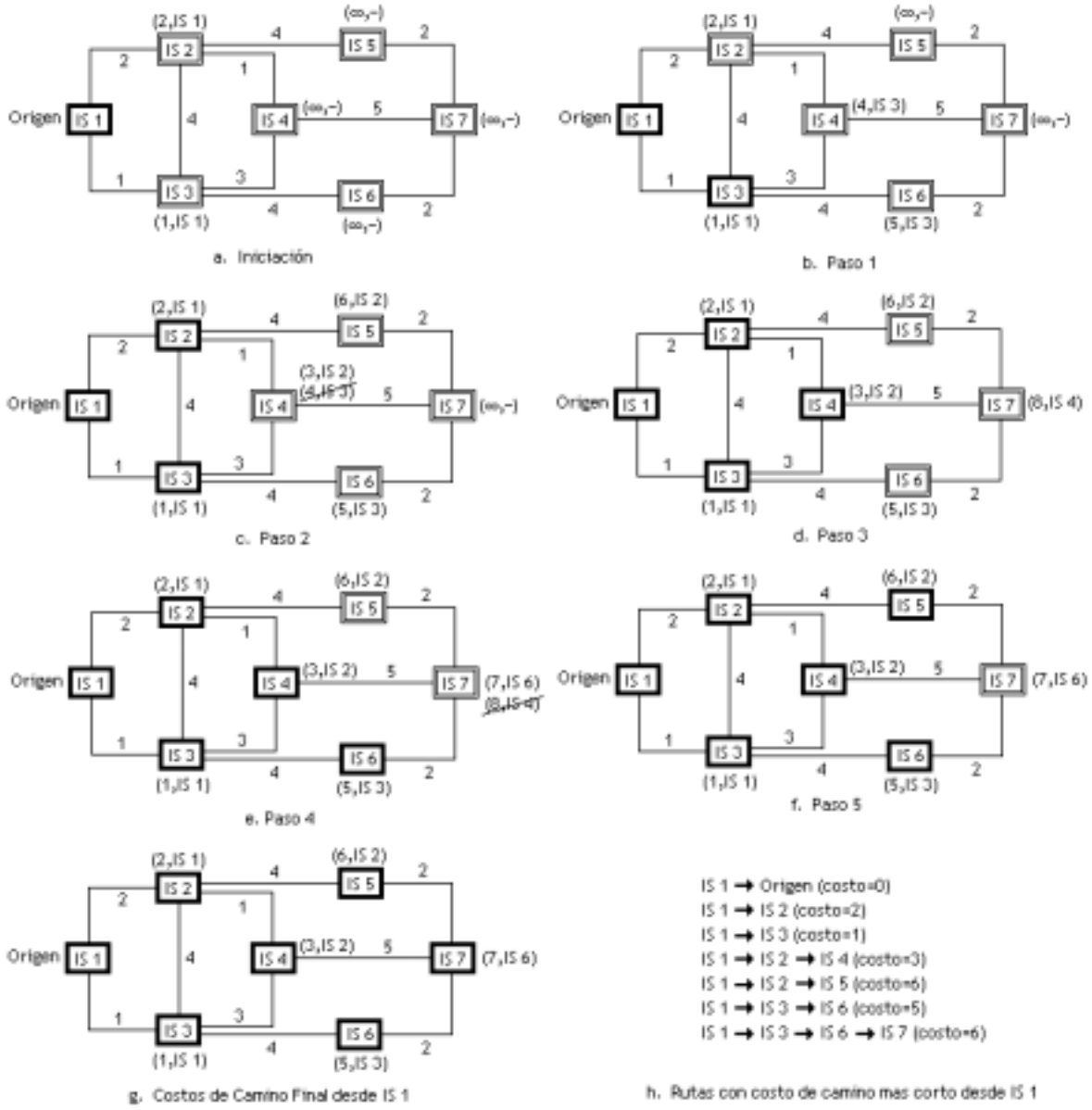
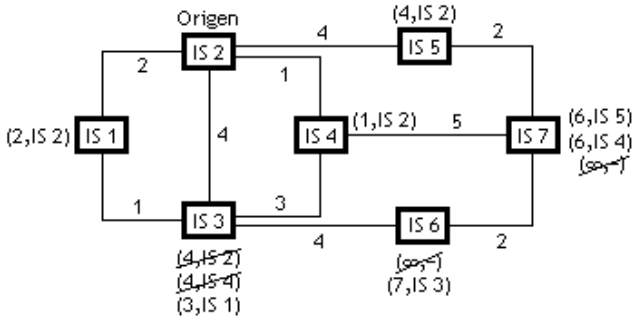


Figura 1.2. Cálculos de Costo de Ruta más Corto.

Por ejemplo, la entrada de IS 4 (4,IS 3) indica que el costo de camino acumulativo de regreso al origen a través de IS 3 es de 4 unidades: 3 para regresar a IS 3 y 1 para regresar a IS 1. En el caso de IS 2, como el costo de camino a través de IS 3 sería de 5 unidades (4 mas 1) no se modifica el valor tentativo que existe, que es menor. Una vez más, se escoge el IS con el valor de costo mínimo de entre todos los que siguen siendo tentativos; en este caso es IS 2, con un valor de costo de 2, así que se marca como permanente y se calculan los nuevos costos de camino a partir de él. Estos costos se muestran en la parte (c).

Como podemos apreciar, el nuevo costo de camino para IS 4 a través de IS 2 es de sólo 3 unidades, por lo que la entrada existente se sustituye por (3,IS 2). Una vez calculadas todas las demás distancias, se escogerá de nuevo la mínima (IS 4) y se repetirá el procedimiento. Los pasos restantes se muestran en las partes (d) a (g).

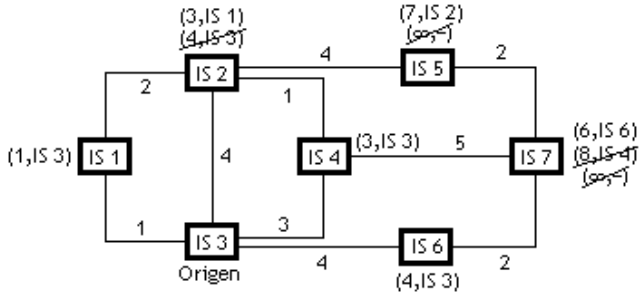
Por último, ya que se ha determinado el valor de costo de camino mínimo desde IS 1 hasta todos los demás IS, es decir, una vez que son permanentes, es posible determinar las rutas con costo de camino más corto (mínimo) de IS 1 a cada uno de los otros IS. Éstas se muestran en la parte (h). Así, para llegar de IS 5 a IS 1 la ruta es IS 1 →IS 2→ IS 5.



a. Cálculos de costo de camino desde IS 2

IS 2 → IS 1 (costo=2)
IS 2 → Origen (costo=2)
IS 2 → IS 1 → IS 3 (costo=3)
IS 2 → IS 4 (costo=1)
IS 2 → IS 5 (costo=4)
IS 2 → IS 1 → IS 3 → IS 6 (costo=7)
IS 2 → IS 4 → IS 7 (costo=6)
IS 2 → IS 5 → IS 7 (costo=6)

b. Rutas con costo de camino más corto desde IS 2



c. Cálculos de costo de camino desde IS 3

IS 3 → IS 1 (costo=1)
IS 3 → IS 1 → IS 2 (costo=3)
IS 3 → Origen (costo=0)
IS 3 → IS 4 (costo=3)
IS 3 → IS 1 → IS 2 → IS 5 (costo=7)
IS 3 → IS 6 (costo=4)
IS 3 → IS 6 → IS 7 (costo=6)

d. Rutas con costo de camino más corto desde IS 3

IS 1

Destino	Camino	Costo
IS 1	-	-
IS 2	IS 2	2
IS 3	IS 3	1
IS 4	IS 2	3
IS 5	IS 2	6
IS 6	IS 3	5
IS 7	IS 3	7

IS 2

Destino	Camino	Costo
IS 1	IS 1	2
IS 2	-	-
IS 3	IS 1	3
IS 4	IS 4	1
IS 5	IS 5	4
IS 6	IS 1	7
IS 7	IS 4/IS 5	6

IS 3

Destino	Camino	Costo
IS 1	IS 1	1
IS 2	IS 1	3
IS 3	-	-
IS 4	IS 4	3
IS 5	IS 1	7
IS 6	IS 6	4
IS 7	IS 6	6

e. Ejemplos de tablas de enrutamiento

Figura 1.3. Cálculos de Costo de Ruta del Camino mas Corto para IS 2 e IS 3.

Consideremos ahora el mismo procedimiento pero primero con IS 2 y luego con IS 3 como origen. Los valores de costo de camino finales asociados a cada IS, junto con sus rutas de costo de camino mas corto, se muestran en la Figura 1.3a a 1.3d. Se pueden hacer varias observaciones acerca del algoritmo:

- Si dos IS tiene el mismo costo de camino calculado, podrá escogerse arbitrariamente cuál se hará permanente.
- Si un nuevo costo de camino calculado para un IS tentativo a través de una ruta (IS) distinta es igual al que ya se tiene, es posible conservar ambos, pues en tal caso cabrá la posibilidad de compartir la carga.
- Si un IS está en la ruta de costo de camino más corto desde IS 1 hasta otro IS de destino, también será el camino más corto desde ese IS al mismo destino. Por ejemplo, la ruta más corta de IS 1 a IS 7 es IS 1→ IS 3 → IS 6 → IS 7 y la ruta desde IS 3 es IS 3 → IS 6 → IS 7.
- Las rutas de costo de camino mas corto calculadas son reversibles; por ejemplo, IS 2 → IS 1 → IS 3 y IS 3 →IS 1 → IS 2.

El efecto combinado de todo esto es que si cada IS calcula su propio conjunto de caminos más cortos desde él hasta todos los demás IS, los caminos calculados para cada IS coincidirán. Esto significa que un IS sólo necesita retener información de enrutamiento que le permita enrutar una unidad de datos de protocolo al primer IS (vecino) en la ruta; es decir, el enrutamiento puede hacerse salto por salto. Las tablas de enrutamiento para IS 1, IS 2 e IS 3 son las que se presentan en la Figura 1.3e.

PARTE 2. MODEL FOR ROUTE DISCOVERY AND SELECTION IN PACKET SWITCHED NETWORKS

1. INTRODUCTION

Route discovery and selection are fundamental to the operation of packet-switched networks. As a packet transits a network, routers must coordinate a path of circuits through which the packet will travel. Each router along the path should have a consistent view of how the packet is to proceed; otherwise the packet may be caught in a never-ending cycle as it passes back and forth between routers. In addition to being consistent, the selected route should be eficient. Network designers must answer two questions: (1) what information should be shared between routers (route discovery) and (2) how routers will use this information to make route decisions (route selection).

1.1 Routing Protocols

The basic problem faced by a router is selecting an exit interface for a packet. Consider the network in Figure 2.1. While it might be easy to look at this simple graph and determine a path from *R*2 to *R*6, the router *R*2, absent external information, views the network as in Figure 2.2. Routing protocols are used to assist the routers in building a more complete view of the network such that intelligent routing decisions can be made.

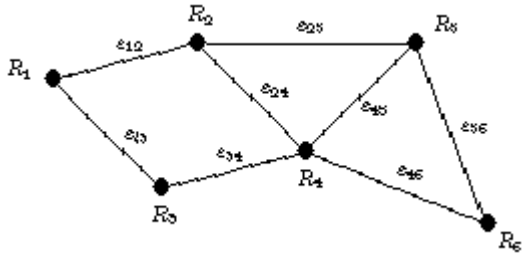


Figure 2.1. Model of a Simple Network**.**

A routing protocol is the language and grammar routers use to exchange information needed for routing decisions. The protocol determines (1) what information needs to be shared, (2) how this information is used to determine packet routes, and (3) how this information is delivered. The amount of time taken to gather this information and modify the routing rules (route tables) is called convergence time.

Information shared by a routing protocol might include delay, jitter, cost, loss probability, bandwidth, and utilization. The routing protocol defines how these values will be obtained from

the network. Often, the values for a link are manually combined to form a single composite called a \metric. " Routing protocols use these metrics and one of the many available shortest path discovery algorithms to determine the least-cost topology of the network. Figure 2.3 shows the least-cost topology of Figure 2.1 given the link metrics of Table 1.1.

Edge	Metric
*e*12	3
*e*13	5
*e*24	4
*e*25	8

Edge	*Metric*
*e*34	2
*e*45	4
*e*46	1
*e*56	2

Table 2.1. Sample Link Metrics for the Network Shown in Figure 2.1.

Routing control can fit into two broad classes: centralized and distributed. Centralized routing protocols focus the decision making to a single location. The routing rules are dispersed from this location to the routers in the network. Distributed protocols, on the other hand, establish peer relationships between routers. The routers exchange information to establish a consistent view of the network. The distributed model is generally superior in terms of robustness and fault tolerance; consequently, it was selected for use on the Internet nearly three decades ago and is still in use today.

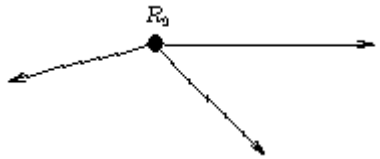


Figure 2.2. Router *R*2's Initial View of the Network.

1.2 The Internet as a Model

Based on its success, the Internet serves as a good model upon which to explore the routing problem. Initially a research project limited to the U.S. military and select educational institutions, the Internet has evolved into an integral part of today's economy. As part of this evolution, the economic value of ensuring proper routing decisions has also grown. Over the past thirty years many techniques have been tried and refined in an effort to provide increasingly stable and robust service. It is difficult to predict if the current routing protocols will scale with the rapid growth of the Internet. Some researchers assert that we are \eating the intellectual seed corn of the 70s". Continued research is necessary.

A wide variety of applications use the Internet, many of which have different requirements for their end-to-end packet transmission characteristics. Real-time interactive voice transmission requires low delay and low jitter. Transferring large files might be less concerned with jitter and

delay and more concerned with bandwidth. Supporting a variety of classes is the subject of much ongoing research. Most applications benefit from the optimization of two key parameters: delay and loss probability. Within the current model of the Internet, network service providers (NSPs) usually try to optimize at least these two parameters.

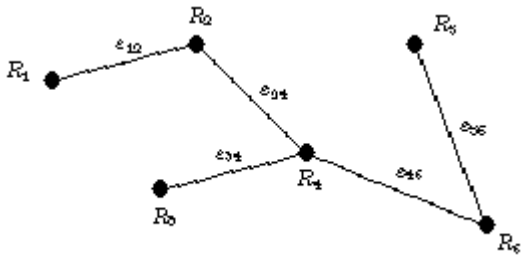


Figure 2.3. Router *R*2's Least-Cost Topology.

NSPs often use a link-state routing protocol, such as OSPF or ISIS, for optimizing these parameters. A link-state routing protocol creates in each router a graph which represents the network. The state of each edge, or link, in the graph is known by all routers, hence the name. The state information is distributed through link-state packets through some form of reliable flooding. Knowing the complete topology and the metrics associated with the links, each router uses a shortest-path algorithm to build a shortest-path tree for the network. The router is the root of the tree while the branches define the minimum-cost paths to every destination in the network. The cost or distance of the path is the sum of the metrics associated with each link in the path.

1.3 The Routing Problem

Determining how to set the metrics for the links is a difficult problem for large networks. Originally the metrics were automatically determined and dynamically updated, and were a function of the available capacity on the link. Largely this practice has been abandoned in favor of statically set metrics for each link. The process of setting the link values is typically a manual effort. When assigning the metrics, trained network operators attempt to take into consideration multiple factors including current link utilization, physical proximity, traffic patterns, and link capacity. Inevitably adjustments must be made, and the metric values are iterated upon in search of an optimum.

Despite the flexibility inherent in this system, links in the network often become overloaded and the packet loss probability (packet drop) becomes unacceptable. At any given time, it is rare that more than a few of the links in the network will be overloaded.

Changing a metric to move a small amount of traffic off the link often results in a large shift and the overload reappears elsewhere in the network. This effort is similar to trying to fatten a bubble by placing a glass dish on top of it. By carefully pressing down the bubble might fatten

slightly, but inevitably it will just move somewhere else. In the worst cases, failure to accurately predict the traffic shift results in a situation worse than what was being fixed in the first place.

Recently, protocols have been introduced to assist NSPs in traffic management. One such protocol is MPLS. MPLS seeks to address four perceived problems: (1) scalability of network layer routing, (2) greater flexibility in delivering routing services, (3) increased performance, and (4) simplified integration of routers with cell-switching-based technologies.

Many NSPs which have deployed MPLS have done so primarily for the second reason. Using MPLS, network engineers can define specific paths for traffic flows in the network, effectively allowing them to deaggregate a portion of the traffic. These paths are not required to follow the least-cost topology. In this framework, engineers can move small increments of traffic from a congested link with more control over where these flows end up. The IETF (Internet Engineering Task Force) is developing CBR (constraint based routing) to provide an automated system for path assignment. Even though this approach provides additional control, questions remain regarding how systems can optimally assign the paths. Minimizing both loss probability and delay has been shown to be in a class of problems known as nondeterministic polynomial-time complete (NP-Complete). There is no known deterministic polynomial-time algorithm capable of producing a solution for a problem in this class, and if one did exist, it could be used to solve every other nondeterministic polynomial-time problem. It is unlikely one will be found, so algorithm designers are left with two options: simplifying their models (itself heuristic in nature) or devising heuristic-based approximation algorithms. This paper proposes one such heuristic method based on evolutionary or genetic theory.

1.4 Genetic Algorithms

"How can computers be made to do what needs to be done, without being told exactly how to do it?".

Evolutionary and genetic algorithms are based on the principles of Darwinian selection.

In general terms, these algorithms mimic nature's two-step iterative process of random variation followed by selection. Genetic algorithms typically require little problem-specific knowledge; they need only a measure to determine the quality of a potential solution. This measure can be explicit or, as proposed herein, implicit. Genetic algorithms operate on the premises of creating large populations which iteratively undergo evaluation, selection, reproduction, and mutation. Members of the population best adapted to the environment or landscape (as defined by the evaluation function) thrive (through selection and reproduction) while those less well-suited are repressed. Reproduction and mutation allow the population to explore the landscape in search of better solutions. These two operators are especially important if the landscape is changing. Because the evolutionary approach does not require the calculations to be restarted from the beginning as the landscape changes, they are well suited for a dynamically changing problem, such as route selection.

1.5 Applying Genetic Methods to the Routing Problem

"Evolved solutions are not brittle; they are usually able to grapple with the perpetual novelty of real environments."

Genetic methods are well suited to address the route discovery and selection problem. The instantaneous network conditions are in constant flux and can not be efficiently communicated to all routers in the network. The complexities of the optimization problem itself disqualify it from being solved precisely even in the presence of complete information. When genetic techniques are applied, much of this complexity can be ignored. A primary task is identifying a means by which route choices can be compared to each other. In the context of the general optimization problem for routing, a method is needed to compare routes based on loss probability and delay. The genetic method proposed here assumes the model of a core transit network in the context of the Internet's architecture. Such a network accepts packets at one edge and delivers them to another. Figure 2.4 shows the model network of Figure 2.1 inside the cloud serving as transit for traffic from a source, *S*, toward a destination, *D*.

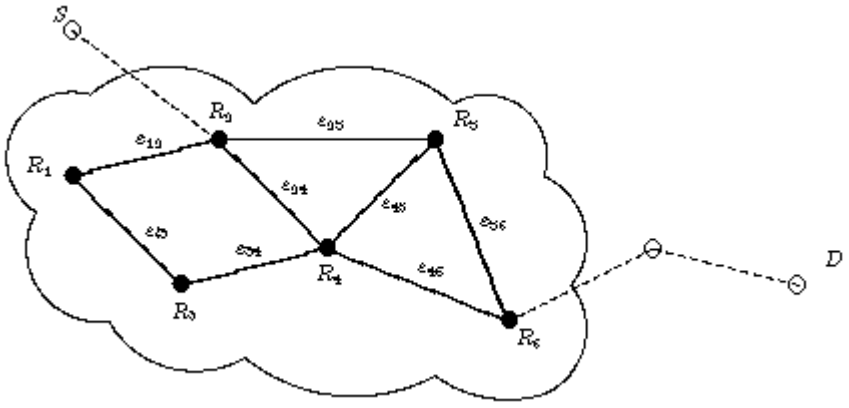


Figure 2.4. Model of a Transit Network.

For the purposes of this proposal the destination will be redefined to be the address of the egress edge-router the router from which the packet will leave the network. In the context of Figure1.4, *R*6 is the egress edge-router for a packet entering the network at *R*2 destined for *D*. Using this definition for destination, the expected number of destinations will be limited to the number of edge-routers in a network. The aggregation of all traffic destined for a particular egress edge-router represents a larger flow of packets than if each final destination was considered individually. Convergence time will be a function of the number of packets owing toward a destination. This aggregation of packets has the potential to allow the routing to converge more quickly than if the destinations were not aggregated.

Within each router an object representing a population will be sustained for each egress edge-router. We can visualize this population as a pool of parasites (the reason for this name will be

apparent shortly). Each parasite favors a specific exit interface on the router[1]. The parasite also contains enough information that it might eventually return to its original population.

Upon receiving a packet, the router takes a single parasite from the destination's population and attaches it to the packet, which is then sent on the interface favored by the selected parasite. The process repeats itself along the path to the egress edge-router with each router appending a parasite to the packet.

Prior to sending the packet off the network, the egress edge-router will remove the parasites acquired by the packet as it traveled through the network. The packet will then be forwarded off the network toward its final destination. The parasite string will be sent back on the reverse path as defined by the parasite string. This transmission will not be dependent upon genetic routing, as each hop in the network will be completely defined by the parasite string. Each router which attached a parasite to a successfully routed packet will be able to recover that parasite and place it back into the appropriate population. Parasites attached to packets which are dropped will be lost from the population. Parasites attached to packets which take suboptimal paths through the network will not be as strongly represented in their base populations as they will spend more time along both the forward and reverse paths.

Parasites attached to packets taking better paths (both in terms of loss probability and delay) will quickly return to their original populations. The representation of the parasites which favor the best exit will generally be larger and these will be favored in the selection of parasites for future packets. Small populations of parasites favoring suboptimal routes will be maintained through mutation and reproduction. In the event of major topology changes these parasites will be needed for reconvergence. This research will demonstrate that it is possible to automatically discover near optimal routes within a core network without any manual configuration of link characteristics or any explicit measurements of traffic patterns.

This proposal is organized into the following sections: Section 2 provides details about the network model and the optimization problem. Related research in the fields of routing and genetic algorithms is discussed in Section 3. Section 4 describes the proposed solution. The plan of work is outlined in Section 5. The summary is found in Section 6.

[1] Only point-to-point connections will be considered. For multipoint connection the parasite will have to represent a LAN neighbor. This will be significantly more complex.

2. PROBLEM DEFINITION

2.1 The Network

A network can be modeled as a directed graph, $G = (V; E)$, where the set V represents the routers in the network and the set E represents communications links connecting the routers.

2.2 Constraints and Metrics

Each edge, ***eij*** , in E has a capacity, c(***eij***), a delay, d(***eij***), and a loss probability, p(***eij***).

$$c(e) = \Re^+ \; \forall e \in E$$

$$d(e) = \Re^+ \; \forall e \in E$$

$$0 \leq p(e) \leq 1 \; \forall e \in E$$

The capacity is a measure of how many units can be carried from ***i*** to ***j*** on edge ***eij***. Often, the capacity is referred to as the bandwidth of the link and is defined in bits per second. The delay represents the amount of time taken for a packet to travel from ***i*** to ***j*** on edge ***eij***. d(***eij***) can be modeled as the sum of the propagation delay (physical distance divided by the speed of light in the media), the transmission delay (size of the packet divided by the bandwidth of the link), and the queuing delay (function of the instantaneous load of that link). In wide-area IP networks transmission delay is often negligible compared to propagation delay. Given the speed of light in fiber, there are 43 ms of delay for every 1000 km of fiber path distance, while the transmission delay for a 1500-byte packet on an OC3 (155 Mbs) is only 77 μs. For an OC48 the transmission delay is 12 times smaller: 4 μs. Therefore, in modern high-speed WAN circuits transmission delay is negligible. Other factors, such as the temperature of the media, may affect the delay, but these are typically small enough to be ignored. The loss probability is also multi-variate. Primary causes of packet loss include link overutilization (oversubscription) and transmission errors.

All links are bidirectional and typically have the same capacity in each direction. Two of the three components of delay (propagation and transmission delay) are the same for both directions. Queuing delay will vary based on the load of the router sending the data. The loss probability may be different in each direction.

2.3 Flows

In real-world networks it is unreasonable to model each packet individually; therefore, packets may conceptually be combined into flows. A flow, f(***i***; ***j***), is defined as the expected aggregate traffic of all packets entering the network at a common node, ***i***, destined for a single exit node, ***j***, measured in bits per second. The flow may take multiple paths through the network and transit intermediate nodes. The amount of traffic between intermediate nodes resulting from this flow is not included in their own flow values. For example, in Figure 2.1, ***f*(2; 6)** represents

the flow between ***R*2** and ***R*6**. Some of *f*(2*;* 6)'s traffic might be sent to ***R*4** enroute to ***R*6**. This traffic from ***R*2** to ***R*4** supporting ***f*(2*;* 6)** is not part of the flow ***f*(2*;* 4)**.

By definition the flow for ***f*(*i; i*)** is zero. This notation will help simplify some of the following equations.

2.4 Route Table

There are many ways of defining how the flow will travel through the network. In the contemporary Internet each router makes its own decision regarding the path of a packet. Routers do not have an impact on a routing decision to be made by a downstream router beyond their choice to pass the packet in the direction of the downstream. This implies the routing decisions must be consistent to avoid routing loops. Cisco's tag switching and MPLS overcome this limitation to some extent through the creation of overlay paths. This work will only consider the general routing problem in which each router bases routing decisions entirely upon the destination address of the packet.

The priority list for node *s* defining the probability of taking a particular exit, ***eij***, for traffic toward destination ***d***, ***r*(*s; d;* esj)**, is referred to as the route table for ***s***. From the perspective of each node the sum of the probabilities for each destination must be one.

$$\sum_{j \in A(s)} r(s, d, e_{sj}) = 1 \qquad (1)$$

where ***A*(*s*)** is the set of nodes adjacent to ***s***.

Often the values for ***r*(*s; d;* esj)** are set to either one or zero, meaning the route table uniquely identifies the exit interface for each destination. By definition,

$$r(v, v, e) = 0 \ \forall e \in E, \qquad (2)$$

meaning node *v* does not exit traffic destined for itself.

2.5 Utilization

To calculate the utilization of a link the definition of flow needs to be extended. A flow ***f*(*i; j*)** includes only the traffic originating on node ***i*** destined for node ***j***. A new variable, *h*, representing the hop count, is used to define a flux of traffic from ***i*** to ***j***, $\lambda(i,j,h)$. $\lambda(i,j,h)$includes all traffic which arrives at ***i*** destined for ***j*** having passed through ***h*** nodes. When ***h* = 0** the flux is equal to the flow.

$$\lambda(i,j,0) = f(i,j) \qquad (3)$$

The flux of traffic at node ***i*** destined for node ***d*** with hop count ***h*** can be found by summing the traffic destined for ***d*** which is sent to node ***i*** at ***h* - 1**. Given node ***m*** adjacent to ***i***, the amount of traffic destined for ***d*** sent to node ***i*** with ***h* - 1** is

$$r(m, d, e_{mi})\lambda(m, d, h-1). \quad (4)$$

The sum of this traffic from all nodes adjacent to ***i*** is

$$\lambda(i, d, h) = \sum_{m \in A(i)} r(m, d, e_{mi})\lambda(m, d, h-1) \quad (5)$$

where ***A*(*i*)** represents the set of nodes adjacent to ***i***. Since ***r*(*d*; *d*; *e*) = 0**, there is no need to explicitly remove ***d*** from the set created by ***A*(*i*)**. The utilization of the edge is the sum of all fluxes using the edge.

$$u(e_{ij}) = \sum_{d \in V} \sum_{h=0}^{\infty} r(i, d, e_{ij})\lambda(i, d, h) \quad (6)$$

It can be observed in Equation 6 that inconsistent route assignments might lead to infinite utilization. A trivial example of this is if ***r*(*i*; *d*; eij) = 1** and ***r*(*j*; *d*; *eji*) = 1**. Node ***i*** would send all of the traffic destined for *d* toward ***j*** and ***j*** would send all of it back to ***i***. If there is any steady-state traffic toward *d* through ***i*** or ***j***, it will be caught in a loop.

2.6 Average Delay and Loss Probability

The average delay can be obtained from the utilizations without specific path information. The total traffic, *t*, in the network is given by:

$$t = \sum_{i \in V} \sum_{j \in V} f(i, j). \quad (7)$$

The average delay of all traffic in the network is the weighted sum of the delays. The weighting factor is the fraction of the total traffic passing over each edge.

$$u_{ave} = \frac{1}{t} \sum_{e \in E} u(e) d(e) \quad (8)$$

The loss probability is significantly more dificult to express in closed form and is not presented in this proposal. Because a flow represents an infinite stream of data, the probability that at least one packet within a flow will be lost approaches one. Therefore, in the context of packet loss probability, the percentage of packet loss is used.

2.7 Optimization Problem

The optimization problem for network routing dealt with by this paper can be summarized as follows. Given a network and a set of flows, determine the values for ***r(s; d;* esj)** such that the following criteria are met:

1. No edge utilization exceeds edge capacity.
2. The average delay is minimized.
3. Packet loss is minimized.

The first and last criteria are somewhat related, as link over utilization causes packet loss. The last criteria is meant also to capture the loss probability due to link characteristics.

A solution to the above optimization problem should also have the ability to adjust to changing network dynamics. The flows are estimates and change over time. The network topology also changes due to link and router failures and the addition of new circuits and routers to the network.

The following section will discuss some of the current solutions to the above problem Researchers have been able to find polynomial time algorithms to solve the optimization problems individually, but a literature survey discovered no method for joint optimization.

3. RELATED RESEARCH

Much research has been done in the areas of routing and genetic research. Recently, researchers have begun to combine the two fields and are exploring genetic or evolutionary routing. This section begins with an overview of some of the work done relating to the basic routing problem. This will be followed by an overview of genetic and evolutionary algorithms. Lastly, applications of genetics to the routing problem will be examined.

3.1 Routing

The most common approach to solving the routing problem is to assign weights (metrics) to the edges in the graph, distribute this information to each router, and route traffic along the shortest paths. Commonly used routing protocols on large NSPs include IS-IS (intermediate-system to intermediate-system and OSPF (open shortest path first). The shortest-path problem can be solved in ***O(V*lg *V* + *E*)** time using Dijkstra's algorithm and Fionacci heaps. The fundamental problem is determining how to assign the metrics to ensure link capacity is not exceeded. Further complicating this problem is inherent lack of flexibility in distributing the traffic across multiple paths for a given source-destination pair. Some protocols can split traffic equally between equal-cost multipaths (ECMPs), but the ratio can not be changed. Because of this limitation, identifying the optimal set of metrics has been shown to be NP-Hard.

3.1.1 Static Metrics

Cisco recommends setting the metrics inversely proportional to the link capacities. Missing from this recommendation is consideration for the flows. This approach has been shown to be non-optimal, by but is among the better traffic-oblivious approaches. It is reasonable to use this metric as a starting point and then iteratively adjust the metric in response to traffic characteristics. More sophisticated approaches include flow information in their calculations. This has potential benefits, but the flows are only predictions of anticipated traffic. There may be limited value in precisely solving a problem with imprecise data. Further complicating the problem is the fact that the flows may be affected by the metrics, as BGP (border gateway protocol) may select an exit based on the values of the metrics. One heuristic approach put forth in produced near-optimal solutions for the capacity management problem. However, in a dynamic network environment, the computational complexity of this approach may be prohibitive.

Neither of the above approaches seeks the joint minimization with delay or other parameters. Joint optimization problems are often NP-Complete, and it has been shown that the problem of finding a path subject to any two of the following metrics is NP-Complete: delay, loss probability, cost, and jitter.

MPLS (Multiprotocol Label Switching) is an overlay protocol which assists the underlying routing protocols by building explicit paths. One of the initial applications of MPLS is traffic engineering. Its importance is discussed in and. MPLS allows the flows to be disaggregated and routed on paths not following the least-cost topology. This allows traffic engineers to manually shift traffic from overutilized to underutilized links in a controlled manner. Using this capability, demonstrated a linear programming technique to optimally assign paths to the flows. It is not clear if the technique can be extended to operate efficiently in an environment where both the flow values and network topology are dynamically changing.

3.1.2 Dynamic Metrics

Static metrics, while inherently simple, are, as their name implies, static. Their main limitation involves their inability to adjust to the changing network. This weakness is easily recognized and many researchers have attempted to overcome it though techniques based on dynamically adjusting the metrics in response to traffic changes. In a sense, these algorithms seek to automate the manual task of iteratively adjusting the metrics to account for current traffic conditions.

The ARPANET (the original name given to the Internet) served as a testbed for approaches that automatically adjust link metrics to avoid congestion. The original routing algorithm altered the link metrics according to the measured queue length. (The queue length is an indicator of the utilization of a given link at the instant it is sampled). This technique did not work well. Basing the link metric on the instantaneous sample of queue length created instability as the network load increased. This approach also suffered because it did not account for bandwidth or latency.

At the time, the bandwidth of the wide-area network links did not vary greatly, so it was not as pronounced a problem It would be a more serious problem now.

Learning from the mistakes of the original approach, researchers introduced a new routing mechanism. This approach considered both link bandwidth and latency, as well as delay. Delay was measured by tracking packet departure times and comparing them with the arrival time of the acknowledgements. Using these measures, the reliability of the link was also considered, as packet loss could be identified by the protocol. Perhaps most importantly, the value was averaged over many packets to smooth out the large swings in the original approach.

Although this new approach represented a significant improvement, it too tended to break down under heavy load. One of the causes for this was the large dynamic range of the metrics: one link could be made to cost up to 127 times as much as another.

This would virtually assure it would not be used. In addition, satellite links were unduly punished, which at the time created problems.

The ARPANET researchers tried yet another dynamic metric approach, termed the revised ARPANET routing metric. This approach compressed the dynamic range of the metrics and dampened the rate at which the advertised metric could change. The new approach had the following characteristics:

- A highly loaded link could cost no more than 3 times its lightly loaded cost.
- The most expensive link (highly loaded 96 kbs satellite link) could cost no more than 7 times as much as the least expensive (lightly loaded 56 kbs terrestrial link).
- Terrestrial links were favored over satellite links having the same bandwidth and utilization.
- High-speed satellite links were favored over low-speed terrestrial links.
- The cost is a function of utilization only at high loads.

The slopes, offsets, and breakpoints for the above parameters were determined by trial and error.

In the long run, this approach did not scale well with the rapidly expanding Internet. As newer, higher capacity circuits became available the graphs were not modified. Furthermore, the Internet's transition into commercial control overshadowed the need for dynamic metrics. Statically set metrics then became accepted as the defacto standard.

3.1.3 Random Routing - No Metrics

One of the more radical approaches to routing is randomly selecting exit interfaces for packets. Surprisingly, this approach has some attractive qualities:

- It is simple, both in conception and implementation.
- It is relatively insensitive to changes in the structure of the network. The routers do not have to know about topology changes.

- No routing messages are required.
- It is one of the few routing procedures where it is possible to get meaningful analytic results.

However, it has some significant drawbacks as well:

- It does not take advantage of available information.
- The number of nodes visited by each packet is large. The remaining drawbacks are direct implications of this.
- The mean total traffic that the network can accept is small.
- It is inefficient in terms of message delay.
- It increases the internal traffic.

Due to these disadvantages, this approach receives very little attention outside of theoretical work. However, the approach proposed herein is similar in some ways and seeks to keep many of the positive attributes and eliminate the negative ones. In particular, the proposed approach aims to be simple, adaptive, and free of routing messages.

3.2 Genetic and Evolutionary Algorithms

The underlying metaphor of genetic algorithms is that of natural evolution. "In evolution, the problem each species faces is one of searching for beneficial adaptations to a complicated and changing environment. The 'knowledge' that each species has gained is embodied in the makeup of the chromosomes of its members". Research in genetic and evolutionary algorithms has been going on for over forty years. Three works are generally recognized as foundational for the field:

- John Holland's *Adaptation in Natural and Artificial Systems*.
- Ingo Rechenberg's *Evolution strategies: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution*.
- Lawrence Fogel's *Artificial Intelligence through Simulated Evolution*

For the first three decades the field languished. Researchers assumed the techniques could not be used to solve even the simplest of problems. However, the techniques may not have been as much at fault as the limited computing power of the time. As computing power has increased so has the ability to simulate the processes of evolution. Genetic algorithms have the following basic components:

- a means of encoding solutions to the problem as a chromosome,
- a means of obtaining an initial population of solutions,
- a function that evaluates the "fitness" of a solution,
- reproduction operators for the encoded solutions, and
- appropriate settings for the genetic algorithm control parameters.

These components are used iteratively to allow the population to evolve and adapt over time.

3.2.1 Encodings

Encoding maps the problem being solved into a representation usable by a computer program. Optimal encodings have caused much debate. Prior to the No Free Lunch Theorem, a sense of optimism existed that an ideal encoding would be discovered. During a conference on evolutionary algorithms, a speaker asserted, “We are confident we'll soon find the ideal encoding, because Nature was able to do so. " Emanuel Falkenauer accurately replied, “I don't buy that argument, because we have of course never seen the beautiful creatures Nature could have evolved had it used another encoding."

The most typical encoding uses a binary string of ones and zeros. Early researchers claimed that this method maximized what they termed the \implicit parallelism" of the algorithms, and therefore encouraged their use. Eventually this theory was proved wrong. Researchers are now left to determine which encoding best suits the specific problem they are trying to solve. The majority of genetic algorithms, while inspired by natural evolutionary systems, can seldomly be viewed as biologically plausible. The most significant gap is in how the individuals to be evolved are represented. However, it is argued that staging a genetic algorithm in a more biologically realistic framework serves to improve its performance. Therefore, care must be taken when choosing an encoding to ensure it is as biologically realistic as possible.

3.2.2 Initial Population

If nothing is known about potentially good solutions the initial population may be randomly generated. Otherwise it may be carefully created with potential solutions in mind. If the population is overly specialized from the start, the population may converge and other good solutions may not be explored. It is observed in that “One might employ a heuristic to choose the initial population in an attempt to introduce the `right' genetic building blocks into the population. However, this can lead to problems since genetic algorithms are *notoriously opportunistic*". For this reason as well as the inherent simplicity, a semi-random process is often used to create the initial population.

3.2.3 Selection

Selection is the mechanism through which superior solutions are identified in the population. The selection process can be used either to select the parents for the next generation or to select solutions which need to be eliminated from the current generation. Many forms of selection can be considered. One method is ”survival of the fittest," where only a certain number of the most well-adapted solutions are allowed to survive to the next generation. Another method involves a tournament approach in which solutions compete against each other, the weaker solutions being eliminated. The tournament is not always played to completion, allowing some of weaker solutions to survive (luck of the draw). This can be advantageous, as it allows a broader solution space to be explored. Many other approaches are available for the selection process. The general principle is to favor better solutions over weaker ones.

3.2.4 Reproduction

Reproduction is used to generate offspring solutions from surviving parent solutions. In nature, there are basically two options for reproduction: sexual and asexual. In sexual reproduction, genetic material is taken from two parents and recombined to form one or more offspring. Asexual reproduction, or cloning, takes the genetic material from one

parent and creates offspring. Asexual reproduction requires mutation to create offspring which are capable of adapting. The field of genetic and evolutionary methods does not need to restrict itself to the reproduction schemes found in nature. Indeed, it is possible to conceive of more extreme forms where genetic material is taken from three or more parents to create offspring. As with encoding, no single best variation operator for all problems exists. Potential benefits of reproduction include the following:

- greater efficiency for adjusting to a changing landscape,
- bringing together independently-created beneficial mutations, and
- removing unwanted mutations.

3.2.5 Mutation

Mutation is used to explore parts of the landscape otherwise not reachable through normal reproduction. Without it, the population tends to converge to a homogeneous state where individuals vary only slightly[44]. Mutated individuals often are created with fatal aws and are quickly eliminated from the population. However, occasionally a mutation results in an individual uniquely suited for the landscape. If this individual is strong enough, future generations will tend to evolve toward these characteristics.

A process called hypermutation seeks to more radically change a specified percentage of the population. This process has been demonstrated to provide for a selective advantage just after a major change in the fitness landscape. Less aggressive mutation will require the population to migrate slowly toward the new landscape and possibly become trapped in a local minimum. For this reason, mutation rates can be made to vary over time. If much of the landscape needs to be explored (for example if a network link has gone down) hypermutation may be used. For the steady-state, however, less disruptive mutation may be favorable.

3.2.6 Applications

Due to their flexibility, especially when the actual problem is hard to state, genetic and evolutionary algorithms have been employed in a broad range of applications. For example, one study demonstrated the application of genetic methods to supply-chain optimization from the perspective of the delivery organization to minimize the number of delivery trucks required.

This particular problem had not found satisfactory results using methods outside the genetic paradigm. Genetic methods have also been applied successfully to problems such as simulating

petroleum pricing behaviors, evolving nonlinear controllers for backing up a truck and trailer, and military target recognition.

Genetic programming, or evolving computer programs, is one of the more extreme applications of genetic and evolutionary methods. Interest in the field increased significantly early in the 1990s. 600 papers on genetic programming were published between 1992 and 1996. Three questions must be asked about genetic programming:

- Can computer programs be automatically generated?
- Can automatically-created programs be competitive with human-produced programs?
- Can the automatic process exhibit creativity and inventiveness?

Research has shown the answer to all three questions to be in the affirmative. For example, low-pass filters were evolved. After 49 generations a classical ladder filter with seven rungs had been discovered. George Campbell had been awarded a patent for this same filter design eighty years earlier[2]. Three other evolved filters were creative enough that each would have violated a separate patent.

A simulation was created in which programs competed against each other based on their code length. Shorter programs were given more CPU. The programs consisted of three parts: a self exam routine to determine length, a reproduction loop, and a copy procedure which was called from the reproduction loop. Programs that generated errors were removed from the population. Over successive generations the evolutionary approach yielded several innovative solutions. Early in the simulation one class of programs emerged which behaved like parasites, using the copy routine of other programs, thus saving space. This corresponds to the current use of shared libraries. Even more innovative was the eventual evolution called hyperparasites. This code would allow a parasite to use its copy procedure, but prior to completion it would usurp control of the reproduction loop and claim the offspring as its own. As was the case with low-pass filter design, this study shows that genetic programming solutions can in fact discover creative solutions.

3.3 Genetic Routing

Genetic routing is an extension of genetic algorithms, where the populations consist of routing tables or flow paths. Members of these populations thrive or are repressed based on their ability to move packets to their destinations quickly and with low probability of packet loss. Various approaches for solving routing related problems have been explored.

3.3.1 GBR

A genetic-based routing algorithm (GBR) is a protocol that minimizes hop count and delay. In its initial stages, GBR operates much like existing link-state algorithms, gathering topology

[2] Patent number 1,227,113 awarded in 1917

information and identifying least-cost paths. The algorithm uses unit cost for link-weights, resulting in a hop-countoptimized shortest-path topology.

GBR is a source-routed algorithm, completely predetermining the path a packet will take through the network. The route table is comprised of an explicit path for each destination address. This implies that the routing decisions are primarily made at the edges of the network[3].

One of the key attributes of GBR is that it supports the disaggregation of traffic flows. Over time, multiple paths through the network are identified for a given destination and the traffic toward that destination is split across these paths. The proportion of the traffic each path receives is a function of the path fitness as determined by packets measuring path delay. The weights are determined by

$$w_i = \frac{1/\mu_i}{\sum_{j \in S} 1/\mu_j}, \quad \text{where} \quad \mu_i = \frac{d_i}{\sum_{j \in S} d_j}. \tag{9}$$

The measured delay for route ***i*** is ***di*** and ***S*** represents the set of all routes to the same destination. Routes with higher weights are favored. Routes with very small weights are removed.

Mutation in GBR can be formulated as follows. Given a path to be mutated, $\{n_1, n_2, n_3, \ldots, n_x\}$, where ***n*1** is the starting node and ***nx*** is the destination, randomly select a node, ***n***, from the set $\{n_2, n_3, \ldots, n_{x-1}\}$ Select a neighbor to that node, ***n0***, and find the shortest path from ***n*1** to ***n0*** and also the shortest path from ***n0*** to ***nx*** Join these two paths at ***n0*** and verify that the resulting path from ***n*1** to ***nx*** is loop free. If any loops are present, drop the route. Otherwise, add it to the set of possible routes to the destination ***nx***.

To support crossover or reproduction, select two paths to the same destination with a non-empty intersection of interior nodes. Select a node from the intersection as the crossover site. Exchange all nodes following the crossover site in both paths. As before, remove all paths containing loops.

A significant limitation of GBR is the need for source routing. However, the approach might be usable in conjunction with MPLS.

3.4 A New Approach

While much research has been done to address the routing problem, most algorithms require accurate flow information. In general, this data is difficult to obtain, does not take into account traffic bursts, and is likely to be wrong after a short while due to traffic growth. Other approaches seek to dynamically adjust to measured traffic characteristics, but these are typically

[3] IP currently has limited support for source routing. No more than 9 hops can be specified using the standard IP header. Source demand routing (SDR) defined in extends this. However, source routing is rarely used by hosts and is often disallowed by routers for security reasons.

heuristic in nature, as the precise solutions to the various optimization problems are either not defined or computationally infeasible to obtain. This paper proposes investigating a unique heuristic which diverges from previous research. No explicit routing messages are used. Rather, information is attached to the working packets and retrieved via explicit feedback. Packets following good paths (as defined by what is to be optimized) are favored, while packets following sub-optimal paths are repressed.

4. PROPOSED SOLUTION

The approach proposed herein varies significantly from the previous work. First, there will be no explicit routing messages[4]; instead, routing information will be attached to working packets those carrying user data through the network. Secondly, routers will not attempt to maintain a model of the network topology. The approach will behave much like the random routing algorithm. Unlike the random routing algorithm, however, this approach will seek to learn from the success or failure of packets transiting the network. Many of the same benefits of random routing will remain:

- It will be simple, both in conception and implementation.
- It will not need to be told explicitly about topology changes.
- There will be no routing messages.

This section will describe the basic operation of the proposed approach.

4.1 Parasites

The proposed approach will maintain multiple populations of entities termed “parasites". A parasite is a simple data structure which uniquely identifies its host address and the exit it favors. The host address is needed to ensure the parasite can return after traveling on the network attached to a packet. Initially in each population all exit interfaces will have equal representation[5]. Parasites will be removed from their populations and attached to packets. Depending upon the success of the path followed by the packet, the parasite might eventually return to its original population. Parasites favoring bad exits will become extinct as they never return. The population of parasites will evolve to be filled with those which favor good exits.

4.2 Packet Destination

Maintaining a population for every one of the 4 billion possible destinations on the contemporary Internet is unrealistically complex. First, the data structure representing a population will be non-trivial, and the aggregate memory requirements for such a scheme would be cumbersome. Secondly, the population will converge and learn as a function of the traffic volume for the

[4] The feedback packets might be considered routing messages.

particular destination. It is desirable to aggregate multiple destination addresses together that they might share the same population of parasites. This should lead to more efficient use of memory and better convergence characteristics.

The first, and most obvious, approach is to combine addresses in the same blocks (CIDR blocks) being announced on the Internet. This would reduce the current number of populations to be approximately 100,000. However, this may still be two orders of magnitude larger than needed. A second approach recognizes that even large networks are typically comprised of only a few hundred routers. The exit point for each of the 100,000 routes is one of these routers. Considering the destination to be the router upon which the packet will exit the network (the egress edge-router) will require only a few hundred populations in each router. These populations will be very active and capable of rapidly adapting to changing network conditions.

4.3 Packet Forwarding

A router must select an exit interface for every packet it forwards. To make an exit decision, the router randomly selects a single parasite from the population associated with the packet destination. This parasite is removed from its population and attached to the packet. The packet is then forwarded out the exit favored by the particular parasite. The parasite will remain attached to the packet until it reaches the egress edge-router.

The packet will acquire a parasite from each hop along its path to the network edge. The ordered set of parasites at the egress edge-router represents the path the packet took through the network. The overhead associated with these parasites will be comparable to the overhead of attaching a source-route. If this overhead is determined to be excessive, a mitigation technique might be to attach parasites to a small sample of the total packets. This is discussed in Section 4.8.

4.4 Feedback

Fundamental to the algorithm's operation is the router's ability to obtain feedback regarding the success of packet forwarding decisions. This feedback will be supported by returning the parasites once the packet has made it to the egress edge-router. Upon receipt of a packet at the egress edge of the network, the router will detach the ordered list of parasites and forward the packet off the network toward its final destination. The list of parasites will be used to source-route a packet back along the reverse path. Each router along this path will be allowed to recover its parasite from the set.

Parasites attached to packets which are dropped will be lost. Parasites attached to packets which take suboptimal paths through the network will spend relatively more time in the network than those following more optimal paths. This will reduce their chances of being selected for

[5] Potentially, the initial distribution might be tuned to favor exits of higher capacity.

future packets. Parasites attached to packets taking better paths (both in terms of loss probability and delay) will sooner return to their original populations and be strongly represented. The representation of parasites that favor better exits will generally be larger and therefore will be favored during route selection. Small populations of parasites favoring suboptimal routes will be maintained through mutation and reproduction. In the event of major topology changes, such parasites will be used for exploring the new landscape.

4.5 Routing Loops

Routing loops can be identified as the packet progresses toward the egress edge-router. To do this a router walks the list of parasites in search of one of its own. If found, the path is truncated and all of the parasites within the loop are discarded. The router selects a new parasite from the destination population, attaches it to the newly truncated path and exits the packet. A tight routing loop will tend to quickly empty the bogus parasites from the representative host populations.

The biggest drawback to this approach may be in relation to large routing loops caused by a single router making a bad decision. All routers in the loop will have parasites destroyed, even though their exit selection may be appropriate for the destination.

Consider again the network in Figure 2.1. Assume a packet is traveling from ***R*1** to ***R*6**. If the path followed thus far is ***f R*1(*e*12)***; R*2(*e*25)***; R*5(*e*45)***; R*4 *g*** and ***R*4** selects ***e*24** toward ***R*2** to exit the packet on, ***R*2** would be able to identify a loop when it received the packet. ***R*2** would truncate the path to be ***f R*1(*e*12)***; R*2 *g*** , killing a parasite in ***R*2's** population for ***R*6** which favors ***e*25**, a parasite in ***R*5's** population for ***R*6** which favors ***e*45**, and a parasite in ***R*4's** population for ***R*6** which favors ***e*24**. Potentially, the only parasite which was wrong was the parasite on ***R*4** favoring ***e*24**. Despite this, all parasites within the loop are destroyed. To some extent one could argue that they deserve to be punished for sending the packet toward a router which would make a bad choice. Nevertheless, this issue will need to be explored.

4.6 Population Control

Packet loss in the network will occur and populations will shrink over time. Additionally, the network will be changing, requiring the populations to adapt. To account for these changes, the populations for each destination will reproduce and mutate. No clear mapping exists for a crossover function in this environment. Therefore, the population will reproduce asexually and mutation will be used to ensure variation. Each exit choice will have at least minimal representation in the population. Populations growing too large will have starvation operators applied to downsize the population.

4.7 Packet Loss

Packets typically will not be dropped. If a router selects a parasite favoring a full link, this parasite will be discarded and a new one will be selected. Parasites favoring overutilized egress links will tend to die off rapidly until the link is relieved of a portion of its traffic. Because packets will not be sent on full interfaces they will generally not be dropped and will eventually make it to their destinations. Unfortunately, they may take a long path and jitter will increase. A study on random algorithms found that random paths can be an order of magnitude longer than an optimal path.

Packet loss will occur when all exit interfaces for a router are full. In this case, there is little that can be done other than dropping packets. This research will seek to identify the conditions leading to this. Packets will also be lost if their paths grow too long and their time-to-live counters become zero. Finally, packets will be lost when the transmission and physical layers fail. One goal of this research will be for the algorithm to automatically adjust to such events without explicit notification.

4.8 Overhead

This approach may introduce significant overhead to the network. Parasites might be encoded into as few as 4 bytes each, but over a path of 10 hops this will become larger than the overhead of the IPv4 packet header. The explicit feedback will also add overhead. Unlike traditional approaches where the routing protocol overhead is independent of the amount of end-user traffic being transferred, the overhead associated with this proposed approach is a function of the end-user traffic. This issue will need to be carefully considered and studied.

A potential mitigation may be to attach the parasite data only to a statistically small sample of the traffic. This approach is successfully used to minimize the overhead associated with attaching path information for the detection of denial-of- service attacks. If the approximate number of parasites needed to converge on a path is known, then the sampling interval can be adjusted to minimize the required overhead.

The sampling interval may be dynamic over time. During times of instability it might be beneficial to attach parasites more frequently, while during relatively stable periods less parasites might be used. Alternatively, the sampling rate may be time based, not packet based. This potentially could reduce the interdependence between overhead and end-user traffic.

4.9 Summary

The proposed solution will follow the basic rules of evolution to nurture populations of parasites for each destination. Parasites will live and die based on their ability to route packets along favorable paths. Mutation and reproduction will ensure a dynamically active population capable

of adaptation and discovery. The route selection function will be trivial: select a parasite out of the population for the given destination. Topology changes will not be explicitly communicated to routers. Rather, parasites favoring routes toward downed links will be dropped. Parasites favoring routes towards under- utilized and newly created links will survive and reproduce. Flow information will not be needed for path analysis. Instead, as flows change and links become congested, parasites favoring these links will be repressed and alternate paths will be explored. This will all be accomplished with no explicit routing messages beyond the feedback packets. Routers will not seek to discover who their neighbors are or what the topology of the network is. In effect, they will behave like smart random routers.

PARTE 3. PROTOCOLOS DE ENRUTAMIENTO IP

1. INTRODUCCION

For routing purposes the Internet is partitioned into a disjoint set of Autonomous Systems (AS) as depicted in Figure 3.1. Each Autonomous Systems is administered by a single authority and employs a single routing protocol.

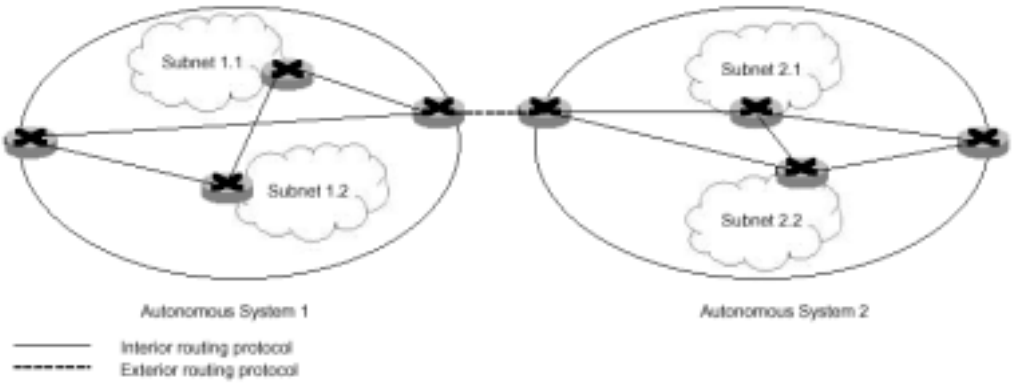


Figure 3.1. Autonomous Systems.

The Internet provides a connectionless data service by which datagrams are generally routed hop-by-hop. Routing is destination-based or more precisely on the network prefix contained in the destination address. The network prefix of incoming datagrams, is compared with entries in the routing table to find the next-hop router. This routing table lookup generally applies a longest-match search criteria. Although IPv4 does support source routing, many router implementations suffer from a performance penalty when source routing is used. This is an important reason why source routing is not widely deployed.

The current Internet routing behavior has been extensively analysed by means of 40,000 end-to-end route measurements between 37 Internet sites using 'traceroute'. These measurements have shown that about 2/3 of the Internet paths are stable for either days or weeks. Furthermore, loops have been detected when the network is in a transient state i.e. while routing information is disseminated, triggered by a network connectivity change. Over three days 60 instances of looping have been detected some lasting several hour and a few several days. Routing loops add to the router congestion. The measurements also revealed that around half of the routing paths where asymmetrical i.e. the path from A to B was at least one node different from the path from B to A.

Intra-domain routing employs an interior routing protocol, such as Open Shortest Path First (OSPF) and Routing Information Protocol (RIP), and inter-domain routing requires a exterior routing protocol such as Border Gateway Protocol (BGP) and Inter-Domain Routing Protocol (IDRP).

The inter-domain routing protocols BGP and IDRP both are path-vector protocols. The routing information exchanged between ASs includes vectors identifying the intermediate ASs through which a certain destination can be reached. No link metrics are exchanged, as each AS may have a different interpretation of a metric. Only reachability information is disseminated. Inter-domain routing protocols need to scale to very large networks and usually employ hierarchical routing to satisfy this requirement.

RIP is a distance-vector routing protocol which is simple and suitable for small Internets. Each router communicates with its neighbours to learn about the network connectivity. Each router periodically updates its neighbours about the cost of routes its knows to certain destinations i.e. exchanges distances. In this way the network connectivity information is propagated through the network. Limitations of the RIP protocol are that a single metric is used and the hop distance is limited (in practice to15) making it not suitable for large Internets. Furthermore, the protocol converges slowly after link failure.

Due to the limitations of RIP, OSPF is now the recommended routing protocol for intra-domain routing. OSPF is a link-state routing protocol that can be applied efficiently in large Internets. OSPF provides hop-by-hop routing, fast dissemination of routing information, multiple path calculation and hierarchical routing. OSPF applies the distributed map concept: each node has a copy of the network topology database which is updated regularly. OSPF has good convergence properties: when the network changes new routes are quickly found with a minimum of routing protocol overhead. For this purpose network connectivity information, contained in so-called Link State Advertisements (LSA), is periodically flooded in the network i.e. each router duplicates the incoming information on each of its outgoing interfaces (except the one it arrived on). In this way each router quickly builds a Link State Database (LSD).

OSPF uses the well-know Dijkstra algorithm to calculate the shortest path. When there are multiple equal cost paths between source and destination, the Dijkstra algorithm determines these paths i.e. Equal-Cost MultiPath (ECMP) [RFC2328]. This allows alternate routing of traffic also referred to as load balancing. Up to five different metrics can be selected, corresponding to the TOS field of IPv4, as listed in Table 1.

The TOS capabilities of a router are advertised in the OSPF Hello messages, exchanged between neighbouring routers. The T-bit in the OSPF options field indicates whether the router supports TOS zero only or the TOS routing options listed in Table 1. When a non-zero TOS shortest path is computed, routers with zero TOS only are skipped, i.e. these routers will not be part of the path.

Metric	Meaning	TOS field
Normal (default)	administrative weight or hop-count (pre-configured)	0
Minimise monetary cost	link cost (pre-configured)	2
Maximise reliability	reliability (pre-configured, dynamic)	4
Maximise throughput	bit duration (pre-configured)	8
Minimise delay	propagation + queueing delay (dynamic)	16

Table 1. OSPF Metrics.

OSPF routing provides a simple form of QoS-based routing by routing on the Type of Service (TOS) field of the IP packet. A different shortest-path tree is computed for each of the five TOS values. OSPF allows classes of traffic to follow different routes. However, overbooking may occur and thus QoS guarantees are not provided. This in contrast to the QoS-based routing solutions, where routing takes into account the QoS requirements of the flow. It is noted that TOS routing has never been widely deployed in the Internet. Therefore TOS routing has been removed from the latest OSPF specification (version 2) [RFC2328] and is not included in the IP version 6 specification [RFC1883]. Furthermore, the Differentiated Services architecture has redefined the TOS field in IPv4 as the Differentiated Service (DS) field. The Differentiated Services architecture assumes that the TOS marking, as defined in RFC1349, is deprecated.

Multi-Protocol Label Switching (MPLS), will be deployed in the future IP networks. The interworking between PSTN/ATM routing and MPLS routing is more natural, than interworking with OSPF, because both are connection-oriented in nature.

1.1. Open Shortest Path First (OSPF)

1.1.1. Background

Open Shortest Path First (OSPF) is a routing protocol developed for *Internet Protocol* (IP) networks by the *interior gateway protocol* (IGP) working group of the Internet Engineering Task Force (IETF). The working group was formed in 1988 to design an IGP based on the *shortest path first* (SPF) algorithm for use in the *Interne*t. Similar to the *Interior Gateway Routing Protocol* (IGRP), OSPF was created because in the mid-1980s, the *Routing Information Protocol* (RIP) was increasingly unable to serve large, heterogeneous internetworks. This chapter examines the OSPF routing environment, underlying routing algorithm and general protocol components.

OSPF was derived from several research efforts, including Bolt, Beranek, Newman's (BBN's) SPF algorithm developed in 1978 for the *ARPANET* (a landmark packet-switching network developed in the early 1970s by BBN), Dr. Radia Perlman's research on fault-tolerant broadcasting of routing information (1988), BBN's work on area routing (1986), and an early version of OSI's Intermediate System-to-Intermediate System (IS-IS) routing protocol.

OSPF has two primary characteristics. The first is that the protocol is open, which means that its specification is in the public domain. The OSPF specification is published as *Request For Comments* (RFC) 1247. The second principal characteristic is that OSPF is based on the SPF algorithm, which sometimes is referred to as the *Dijkstra algorith*m, named for the person credited with its creation.

OSPF is a *link-state* routing protocol that calls for the sending of *link-state advertisements* (LSAs) to all other routers within the same hierarchical area. Information on attached interfaces, metrics used, and other variables is included in OSPF LSAs. As OSPF routers accumulate link-state information, they use the SPF algorithm to calculate the shortest path to each node.

As a link-state routing protocol, OSPF contrasts with RIP and IGRP, which are distance-*vector* routing protocols. Routers running the distance-vector algorithm send all or a portion of their routing tables in routing-update messages to their neighbors.

1.1.2. Routing Hierarchy

Unlike RIP, OSPF can operate within a hierarchy. The largest entity within the hierarchy is the *autonomous system* (AS), which is a collection of networks under a common administration that share a common routing strategy. OSPF is an intra-AS (interior gateway) routing protocol, although it is capable of receiving routes from and sending routes to other ASs.

An AS can be divided into a number of *area*s, which are groups of contiguous networks and attached hosts. Routers with multiple interfaces can participate in multiple areas. These routers, which are called *area border router*s, maintain separate topological databases for each area.

A *topological database* is essentially an overall picture of networks in relationship to routers. The topological database contains the collection of LSAs received from all routers in the same area. Because routers within the same area share the same information, they have identical topological databases.

The term *domain* sometimes is used to describe a portion of the network in which all routers have identical topological databases. Domain is frequently used interchangeably with AS.

An area's topology is invisible to entities outside the area. By keeping area topologies separate, OSPF passes less routing traffic than it would if the AS were not partitioned.

Area partitioning creates two different types of OSPF routing, depending on whether the source and destination are in the same or different areas. Intra-area routing occurs when the source and destination are in the same area; interarea routing occurs when they are in different areas.

An OSPF *backbone* is responsible for distributing routing information between areas. It consists of all area border routers, networks not wholly contained in any area, and their attached routers.

Figure 3.2 shows an example of an internetwork with several areas.

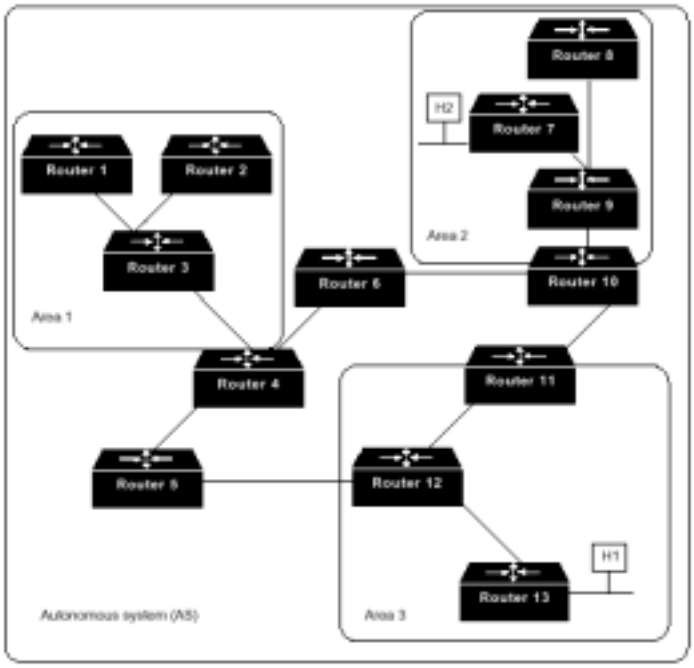


Figure 3.2. An OSPF AS Consists of Multiple Areas Linked by Routers.

In the Figure 3.2, Routers 4, 5, 6, 10, 11, and 12 make up the backbone. If Host H1 in Area 3 wants to send a packet to Host H2 in area 2, the packet is sent to Router 13, which forwards the packet to Router 12, which sends the packet to Router 11. Router 11 then forwards the packet along the backbone to area border Router 10, which sends the packet through two intra-area routers (Router 9 and Router 7) to be forwarded to Host H2.

The backbone itself is an OSPF area, so all backbone routers use the same procedures and algorithms to maintain routing information within the backbone that any area router would. The backbone topology is invisible to all intra-area routers, as are individual area topologies to the backbone.

Areas can be defined in such a way that the backbone is not contiguous. In this case, backbone connectivity must be restored through *virtual link*s. Virtual links are configured between any backbone routers that share a link to a nonbackbone area and function as if they were direct links.

AS border routers running OSPF learn about exterior routes through *exterior gateway protocols* (EGPs), such as *Exterior Gateway Protocol* (EGP) or *Border Gateway Protocol* (BGP), or through configuration information. For more information about these protocols, see Chapter 35, “Border Gateway Protocol (BGP)”.

1.1.3. SPF Algorithm

The shortest path first (SPF) routing algorithm is the basis for OSPF operations. When an SPF router is powered up, it initializes its routing-protocol data structures and then waits for indications from lower-layer protocols that its interfaces are functional.

After a router is assured that its interfaces are functioning, it uses the OSPF *Hello protocol* to acquire neighbors, which are routers with interfaces to a common network. The router sends hello packets to its neighbors and receives their hello packets. In addition to helping acquire neighbors, hello packets also act as keep-alives to let routers know that other routers are still functional.

On *multiaccess networks* (networks supporting more than two routers), the Hello protocol elects a *designated router* and a backup designated router. Among other things, the designated router is responsible for generating LSAs for the entire multiaccess network. Designated routers allow a reduction in network traffic and in the size of the topological database.

When the link-state databases of two neighboring routers are synchronized, the routers are said to be *adjacen*t. On multiaccess networks, the designated router determines which routers should become adjacent. Topological databases are synchronized between pairs of adjacent routers. Adjacencies control the distribution of routing-protocol packets, which are sent and received only on adjacencies.

Each router periodically sends an LSA to provide information on a router's adjacencies or to inform others when a router's state changes. By comparing established adjacencies to link states, failed routers can be detected quickly and the network's topology altered appropriately. From the topological database generated from LSAs, each router calculates a shortest-path tree, with itself as root. The shortest-path tree, in turn, yields a routing table.

1.1.4. Packet Format

All OSPF packets begin with a 24-byte header, as illustrated in Figure 3.3.

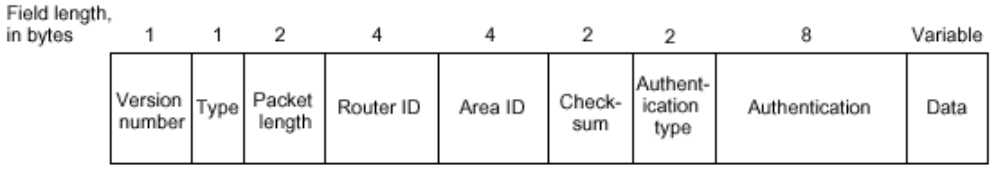


Figure 3.3. OSPF Packets Consist of Nine Fields.

The following descriptions summarize the header fields illustrated in Figure 3.3.

- *Version Number*—Identifies the OSPF version used.
- *Type*—Identifies the OSPF packet type as one of the following:

- ✓ Hello: Establishes and maintains neighbor relationships.
- ✓ Database Description: Describes the contents of the topological database. These messages are exchanged when an adjacency is initialized.
- ✓ Link-state Request: Requests pieces of the topological database from neighbor routers. These messages are exchanged after a router discovers (by examining database-description packets) that parts of its topological database are out of date.
- ✓ Link-state Update: Responds to a link-state request packet. These messages also are used for the regular dispersal of LSAs. Several LSAs can be included within a single link-state update packet.
- ✓ Link-state Acknowledgment: Acknowledges link-state update packets.

- ➢ *Packet Lengt*h—Specifies the packet length, including the OSPF header, in bytes.
- ➢ *Router I*D—Identifies the source of the packet.
- ➢ *Area I*D—Identifies the area to which the packet belongs. All OSPF packets are associated with a single area.
- ➢ *Checksu*m—Checks the entire packet contents for any damage suffered in transit.
- ➢ *Authentication Type*—Contains the authentication type. All OSPF protocol exchanges are authenticated. The Authentication Type is configurable on a per-area basis.
- ➢ *Authentication*—Contains authentication information.
- ➢ Data—Contains encapsulated upper-layer information.

1.1.5. Additional OSPF Features

Additional OSPF features include equal-cost, *multipath routing,* and routing based on upper-layer *type-of-service* (TOS) requests. TOS-based routing supports those upper-layer protocols that can specify particular types of service. An application, for example, might specify that certain data is urgent. If OSPF has high-priority links at its disposal, these can be used to transport the urgent datagram.

OSPF supports one or more metrics. If only one metric is used, it is considered to be arbitrary, and TOS is not supported. If more than one metric is used, TOS is optionally supported through the use of a separate metric (and, therefore, a separate routing table) for each of the eight combinations created by the three IP TOS bits (the *dela*y, *throughpu*t, and *reliability* bits). If, for example, the IP TOS bits specify low delay, low throughput, and high reliability, OSPF calculates routes to all destinations based on this TOS designation.

IP subnet masks are included with each advertised destination, enabling *variable-length subnet mask*s. With variable-length subnet masks, an IP network can be broken into many subnets of various sizes. This provides network administrators with extra network-configuration flexibility.

2. BORDER GATEWAY PROTOCOL (BGP)

1.2. Background

Routing involves two basic activities: determination of optimal routing paths and the transport of information groups (typically called packets) through an internetwork. The transport of packets through an internetwork is relatively straightforward. Path determination, on the other hand, can be very complex. One protocol that addresses the task of path determination in today's networks is the *Border Gateway Protocol* (BGP). This chapter summarizes the basic operations of BGP and provides a description of its protocol components.

BGP performs interdomain routing in Transmission-Control Protocol/Internet Protocol (TCP/IP) networks. BGP is an exterior gateway protocol (EGP), which means that it performs routing between multiple autonomous systems or domains and exchanges routing and reachability information with other BGP systems.

BGP was developed to replace its predecessor, the now obsolete *Exterior Gateway Protocol* (EGP), as the standard exterior gateway-routing protocol used in the global Internet. BGP solves serious problems with EGP and scales to Internet growth more efficiently.

Note EGP is a particular instance of an exterior gateway protocol (also EGP)—the two should not be confused.

Figure 3.4 illustrates core routers using BGP to route traffic between autonomous systems.

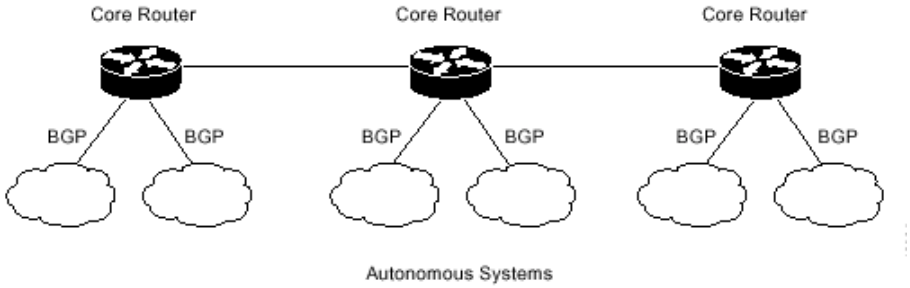


Figure 3.4. Core Routers can use BGP to Route Traffic between Autonomous Systems.

BGP is specified in several *Request For Comments* (RFCs):

- RFC 1771—Describes BGP4, the current version of BGP
- RFC 1654—Describes the first BGP4 specification
- RFC 1105, RFC 1163, and RFC 1267—Describes versions of BGP prior to BGP4

1.3. BGP Operation

BGP performs three types of routing: *interautonomous system routing*, *intra-autonomous system routing*, and *pass-through autonomous system routing*.

Interautonomous system routing occurs between two or more BGP routers in different autonomous systems. Peer routers in these systems use BGP to maintain a consistent view of the internetwork topology. BGP neighbors communicating between autonomous systems must reside on the same physical network. The Internet serves as an example of an entity that uses this type of routing because it is comprised of autonomous systems or administrative domains. Many of these domains represent the various institutions, corporations, and entities that make up the Internet. BGP is frequently used to provide path determination to provide optimal routing within the Internet.

Intra-autonomous system routing occurs between two or more BGP routers located within the same autonomous system. Peer routers within the same autonomous system use BGP to maintain a consistent view of the system topology. BGP also is used to determine which router will serve as the connection point for specific external autonomous systems. Once again, the Internet provides an example of interautonomous system routing. An organization, such as a university, could make use of BGP to provide optimal routing within its own administrative domain or autonomous system. The BGP protocol can provide both inter- and intra-autonomous system routing services.

Pass-through autonomous system routing occurs between two or more BGP peer routers that exchange traffic across an autonomous system that does not run BGP. In a pass-through autonomous system environment, the BGP traffic did not originate within the autonomous system in question and is not destined for a node in the autonomous system. BGP must interact with whatever intra-autonomous system routing protocol is being used to successfully transport BGP traffic through that autonomous system. Figure 3.5 illustrates a pass-through autonomous system environment:

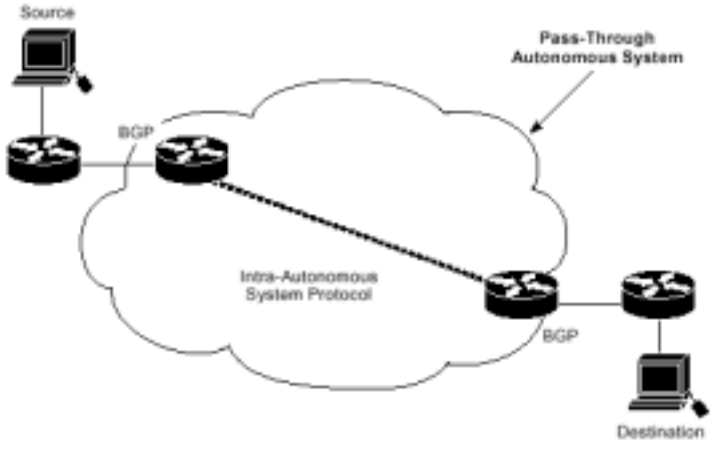


Figure 3.5. In pass-through Autonomous System Routing, BGP pairs with another Intra-Autonomous System-Routing Protocol.

1.4. BGP Routing

As with any routing protocol, BGP maintains routing tables, transmits routing updates, and bases routing decisions on routing metrics. The primary function of a BGP system is to exchange network-reachability information, including information about the list of autonomous system paths, with other BGP systems. This information can be used to construct a graph of autonomous system connectivity from which routing loops can be pruned and with which autonomous system-level policy decisions can be enforced.

Each BGP router maintains a routing table that lists all feasible paths to a particular network. The router does not refresh the routing table, however. Instead, routing information received from peer routers is retained until an incremental update is received.

BGP devices exchange routing information upon initial data exchange and after incremental updates. When a router first connects to the network, BGP routers exchange their entire BGP routing tables. Similarly, when the routing table changes, routers send the portion of their routing table that has changed. BGP routers do not send regularly scheduled routing updates, and BGP routing updates advertise only the optimal path to a network.

BGP uses a single routing metric to determine the best path to a given network. This metric consists of an arbitrary unit number that specifies the degree of preference of a particular link. The BGP metric typically is assigned to each link by the network administrator. The value assigned to a link can be based on any number of criteria, including the number of autonomous systems through which the path passes, stability, speed, delay, or cost.

1.5. BGP Message Types

Four BGP message types are specified in RFC 1771, *A Border Gateway Protocol 4 (BGP-4*): open message, update message, notification message, and keep-alive message.

The *open message* opens a BGP communications session between peers and is the first message sent by each side after a transport-protocol connection is established. Open messages are confirmed using a keep-alive message sent by the peer device and must be confirmed before updates, notifications, and keep-alives can be exchanged.

An *update message* is used to provide routing updates to other BGP systems, allowing routers to construct a consistent view of the network topology. Updates are sent using the Transmission-Control Protocol (TCP) to ensure reliable delivery. Update messages can withdraw one or more unfeasible routes from the routing table and simultaneously can advertise a route while withdrawing others.

The *notification message* is sent when an error condition is detected. Notifications are used to close an active session and to inform any connected routers of why the session is being closed.
The keep-alive message notifies BGP peers that a device is active. Keep-alives are sent often enough to keep the sessions from expiring.

1.6. BGP Packet Formats

The sections that follow summarize BGP open, updated, notification, and keep-alive message types, as well as the basic BGP header format. Each is illustrated with a format drawing, and the fields shown are defined.

1.6.1. Header Format

All BGP message types use the basic packet header. Open, update, and notification messages have additional fields, but keep-alive messages use only the basic packet header. Figure 3.6 illustrates the fields used in the BGP header. The section that follows summarizes the function of each field.

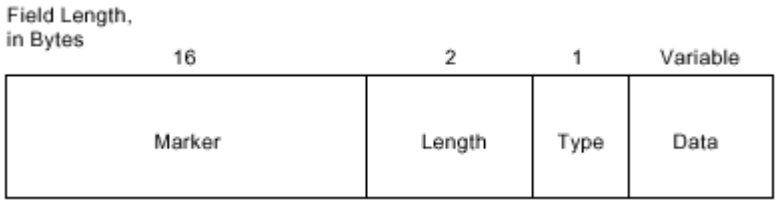


Figure 3.6. A BGP Packet Header Consists of Four Fields.

1.6.1.1. BGP Packet-Header Fields

Each BGP packet contains a header whose primary purpose is to identify the function of the packet in question. The following descriptions summarize the function of each field in the BGP header illustrated in Figure 3.6.

- *Marker*—Contains an authentication value that the message receiver can predict.
- *Length*—Indicates the total length of the message in bytes.
- *Type*—*Type* — Specifies the message type as one of the following:
  - ✓ Open
  - ✓ Update
  - ✓ Notification
  - ✓ Keep-alive
- *Data*—Contains upper-layer information in this optional field.

1.6.2. Open Message Format

BGP open messages are comprised of a BGP header and additional fields. Figure 3.7 illustrates theadditional fields used in BGP open messages.

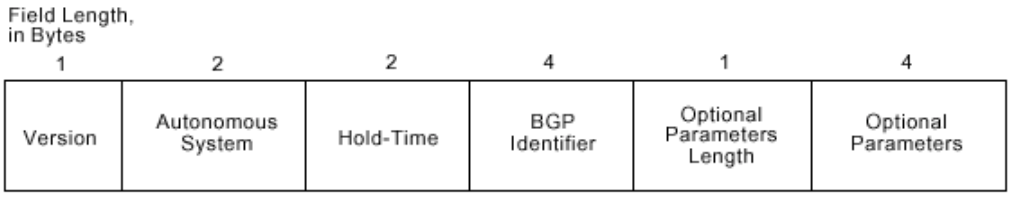


Figure 3.7. A BGP Open Message Consists of Six Fields.

1.6.2.1. BGP Open Message Fields

BGP packets in which the type field in the header identifies the packet to be a BGP open message packet include the following fields. These fields provide the exchange criteria for two BGP routers to establish a peer relationship.

- *Version*—Provides the BGP version number so that the recipient can determine whether it is running the same version as the sender.
- *Autonomous System*—Provides the autonomous system number of the sender.
- *Hold-Time*—Indicates the maximum number of seconds that can elapse without receipt of a message before the transmitter is assumed to be nonfunctional.
- *BGP Identifier*—Provides the BGP identifier of the sender (an IP address), which is determined at startup and is identical for all local interfaces and all BGP peers.
- *Optional Parameters Length*—Indicates the length of the optional parameters field (if present).
- *Optional Parameters*—Contains a list of optional parameters (if any). Only one optional parameter type is currently defined: authentication information.

Authentication information consists of the following two fields:

- ✓ Authentication code: Indicates the type of authentication being used.
- ✓ Authentication data: Contains data used by the authentication mechanism (if used).

1.6.3. Update Message Format

BGP update messages are comprised of a BGP header and additional fields. Figure 3.8 illustrates the additional fields used in BGP update messages.

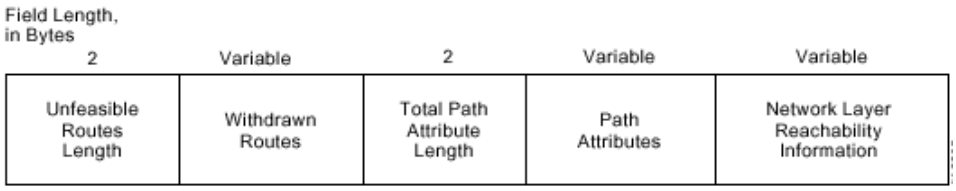


Figure 3.8. A BGP Update Message Contains Five Fields.

1.6.3.1. BGP Update Message Fields

BGP packets in which the type field in the header identifies the packet to be a BGP update message packet include the following fields. Upon receiving an update message packet, routers will be able to add or delete specific entries from their routing tables to ensure accuracy. Update messages consist of the following packets:

- *Unfeasible Routes Length*—Indicates the total length of the withdrawn routes field or that the field is not present.
- *Withdrawn Routes*—Contains a list of IP address prefixes for routes being withdrawn from service.
- *Total Path Attribute Length*—Indicates the total length of the path attributes field or that the field is not present.
- *Path Attributes*—Describes the characteristics of the advertised path. The following are possible attributes for a path:

  - ✓ Origin: Mandatory attribute that defines the origin of the path information
  - ✓ AS Path: Mandatory attribute composed of a sequence of autonomous system path segments
  - ✓ Next Hop: Mandatory attribute that defines the IP address of the border router that should be used as the next hop to destinations listed in the network layer reachability information field
  - ✓ Mult Exit Disc: Optional attribute used to discriminate between multiple exit points to a neighboring autonomous system
  - ✓ Local Pref: Discretionary attribute used to specify the degree of preference for an advertised route
  - ✓ Atomic Aggregate: Discretionary attribute used to disclose information about route selections
  - ✓ Aggregator: Optional attribute that contains information about aggregate routes

- *Network Layer Reachability Information*—Contains a list of IP address prefixes for the advertised routes

1.6.4. Notification Message Format

Figure 3.9 illustrates the additional fields used in BGP notification messages.

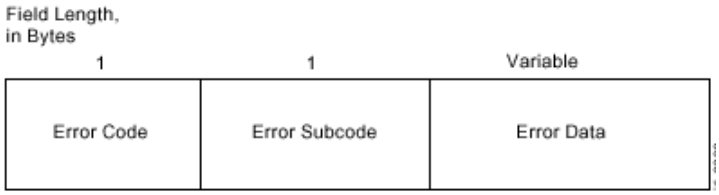


Figure 3.9. A BGP Notification Message Consists of Three Fields.

1.6.4.1. BGP Notification Message Fields

BGP packets in which the type field in the header identifies the packet to be a BGP notification message packet include the following fields. This packet is used to indicate some sort of error condition to the peers of the originating router.

- *Error Code*—Indicates the type of error that occurred. The following are the error types defined by the field:

  - ✓ Message Header Error: Indicates a problem with a message header, such as unacceptable message length, unacceptable marker field value, or unacceptable message type.
  - ✓ Open Message Error: Indicates a problem with an open message, such as unsupported version number, unacceptable autonomous system number or IP address, or unsupported authentication code.
  - ✓ Update Message Error: Indicates a problem with an update message, such as a malformed attribute list, attribute list error, or invalid next-hop attribute.