Message Routing Protocols

Brian McMillin

Abstract

Message routing is a fundamental requirement of digital communication between networked computers. Protocols have been developed and improved over time to address new and varying requirements involving number of endpoints, distances involved, expected data rates, overall bandwidth, allowed error rates and maximum latency.

Comments are included concerning the history and features of different message protocols, and the deficiencies that have arisen due to the effects of modularization and the accretion of stopgap solutions. Special attention is directed toward attempts to build security on top of dated and insecure foundations.

A proposal for a secure, very high speed, stateless routing protocol to replace TCP/IP and streaming data implementations is presented. The proposal includes bit-level descriptions of the operation of router hardware ranging from small LAN-scale routers through backbone infrastructure-scale equipment. Security is of paramount concern and routing is accomplished without revealing the identities of the Originator or Destination. The existence of connections is known only to the endpoints. There is no arbitrary limitation to the size of individual messages.

Contents

Background

Modern internet connectivity is based on a set of protocols originally designed in the 1980's for the interconnection of computers into a local area network (LAN). These protocols have been adapted and extended to connect ever-increasing numbers of computers using more and more sophisticated wired and wireless media. Data rates and volumes have increased exponentially. Security requirements are now critical to ensure the availability, secrecy and integrity of all communications. Adversaries have grown from thrill-seeking hackers to nation-states and global criminal organizations.

The Internet Protocol (IP) routes datagrams based on the Source and Destination IP Address contained within each packet. This is accomplished through the use of lookup tables held by each router and Gateway Addresses that act as catch-alls for unresolved destinations. All route segments are point-to-point.

The Transmission Control Protocol (TCP) adds to the basic IP protocol by allowing connections between endpoints. Establishing a connection means that bi-directional traffic can be handled reliably between the two endpoints. This supports the common instance of a simple request followed by a large response such as loading web pages. In addition, latency along the route may be stabilized, helping to ensure acceptable performance for voice and video streams.

TCP/IP Deficiencies

Routing requires every router to contain lookup tables to be used to enable handling of any possible packet that might be received. These tables must be continuously updated and maintained in order to support new connections in a timely manner.

Connections require every router to maintain a table of active connections in order to speed the handling of bi-directional data messages for the connection.

Maintaining Connections requires periodic transmission of keep-alive messages along otherwise idle paths in order to prevent timeouts and the unwanted discarding of connection table entries.

Timeouts are used for automatic clearing of failures, resetting connections and retransmitting dropped packets. Incorrect and arbitrary values for these timeouts lead to hung connections, slow recovery, poor network performance and inexplicable failures.

Security is compromised by the presence of both the source and destination IP address of the connection in plain text in every packet. This means that an adversary monitoring any router in the path may identify and log the identities of the correspondents, as well as the nature, frequency and volume of traffic.

OSI Protocol Layers Considered Harmful

The Open System Interconnect (OSI) Protocol Layer concept is designed to isolate and encapsulate each of the features of a telecommunication process in order to simplify the design and troubleshooting of complex networks. This “object orientation” was widely viewed as the future of systems architecture in the late 20th century. Unfortunately, fundamental architectural requirements do not fit into this strict hierarchy and only the most elementary network topologies can be directly supported.

The glaring example that I address in this paper is the conflation of the Session, Transport and Network Layers necessary to the implementation of modern communications.

The Network Layer (3) requires access to routing tables and address lookups in order to correctly recognize and route packets. The routing tables may become arbitrarily large and must be delivered and updated by mechanisms outside the OSI definition. Routing of individual packets is handled on an individual basis with no provision for path diversity. Route optimization is not possible unless implemented within the tables loaded by an omniscient outside source.

The Transport Layer (4) requires knowledge of packet segmentation to allow for simple acknowledgement and other requirements designed to ensure reliable message transmission. These requirements mean that sophisticated protocols that could handle dropped packets, burst errors, broadcast transmissions, asymmetric return paths, etc. are precluded due to the arbitrary decision to install “reliable transmission” as a feature of precisely one layer. One need only consider the abysmal performance of links involving geostationary satellites to recognize that reliability should not be the province of a low-level protocol.

The Session Layer (5) is intended to allow the establishment of a persistent session for the exchange of large numbers of packets between two endpoints. This is all well and good, but (as we have seen) the knowledge of connections must leak into lower layers in order to ensure consistent routing of messages over the entire lifetime of the session.

Even simple operations such as bridging network segments require hardware devices that are capable of examining and understanding elements from all three of these layers. Furthermore, translation - on the fly - of fields in every packet is required. Tables of active sessions must be maintained in devices that ideally would be stateless.

OSI myoipa has led to an ever-increasing number of patchwork solutions designed to allow direct communication between non-adjacent protocol layers. Concepts, computations and storage that rightfully would belong in one layer find themselves within the province of other layers.

Altogether, the goals of simplification, isolation and testability have not been realized.

Limitations to Bandwidth Utilization

Most modern network operations require hundreds or thousands of independent, short-term connections. Examples include loading graphical web pages.

Frequently this means building up and tearing down multiple connections - including cryptographic security negotiations - between the same two endpoints, with each connection making a single request to transfer a single tiny file. Interestingly, caches do not reduce the number of required connections, but do reduce the amount of data transferred with the same connection overhead: Responses are simply converted into “Nothing changed here” indicators.

Limitations in the implementation mean that only a small number of connections can occur concurrently.

Combining limited concurrency, excessive overhead and inherent end-to-end delays means that the promise of high-bandwidth physical links cannot be realized.

Limitations to Connection Security

1. All Internet protocol (IP) messages reveal the addresses of the connection endpoints in plain text to all intermediary routers and links.
2. Requests and responses travel back and forth over the same set of routers.
3. The choice of Transport Layer Protocol (Layer 4 - frequently TCP) is also identified in plain text to all intermediaries.
4. There is no guarantee that routes used by one protocol are the same as the routes used by another. The ability for an adversary to detect traffic types makes it easy for them to lie about routes or timings, thus defeating even ICMP TRACEROUTE as a security check.
5. This same packet analysis is what bad actors use to defeat net neutrality. Throttling and prioritization is only possible because of access to the information unnecessarily revealed in packet headers.

None of these situations would be allowed in modern networking where design-for-security is paramount.

Human Rights

The security and privacy of Internet Communications are a human rights issue.

The existing protocols and logging capabilities are completely open to abuse.

Potential threats come from ISPs, telecom carriers, government agencies, casual and criminal hackers and adversarial nation-states.

TCI/IP headers contain private information that is only a tiny step away from being personally identifiable. Given any single outside source of identity information every TCP/IP header would

become Personally Identifiable Information and thus fall under the European General Data Protection Regulation (GDPR). These rules require opt-in for data collection, disclosure of all stored information and the ability to delete that information upon demand. This means that a strict interpretation would require that every traffic log of every router in the entire Internet would be subject to these rules.

Furthermore, every routing decision would require secure logging since the handling of user's data (the TCP/IP headers) must be treated as confidential and only available to approved parties.

This makes the "typos" in Border Gateway Protocol tables - the ones that route all traffic through China - clear violations of GDPR.

Of concern is the fact that the current Internet implementation simply cannot be made secure. It is not possible for any node or router to know what will happen to a packet once it is transmitted. Any traffic can be diverted to any destination, selectively or in bulk. Any attempt to prevent diversion (such as precisely setting the TTL value by the sender) can be easily defeated by an adversary who simply rewrites the value.

Traffic logging of not only Headers but message data can be clandestinely inserted at any point in the network. This information will include the full text of all sent and received data during a connection.

As currently implemented the fundamental structure of the Internet may become an unresolvable human rights violation.

John Gilmore's aphorism "The net interprets censorship as damage and routes around it" applies only to a limited or unskilled censor. If the censor is the owner-operator of your entire country's Internet the concept of "routing around it" becomes moot.

National Security

The telecom infrastructure that embodies the Internet is a national asset for any country. Ensuring that it remains secure and intact is a primary goal of both companies and nations. The fact that the Internet is critical to all aspects of modern life makes it a prime target for national adversaries. Major disruptions to a nations economy can be caused by comparatively trivial manipulations of vulnerable communications.

The fact that control of the power grid, surface and air transportation, emergency services, hospitals, financial systems, environmental monitoring, etc. all rely on security-through-obscurity means that casual attacks are likely to succeed. The presence of "we don't want you to know that" vulnerabilities mean that industry-best-practices such as comprehensive security audits are not followed or ignored. Attackers with advanced knowledge are therefore even more likely to be able to wreak major disruption.

The reliance of these rickety systems on an underlying infrastructure that exhibits the vulnerabilities described in this paper means that a national adversary seeking to cause maximum damage would likely target aspects of the Internet itself instead of the systems of a particular segment of the economy.

Secure communications require robust, comprehensive encryption. Any actor that proposes weak or incomplete encryption technology is advocating vulnerability. Making strong encryption fundamental to the operation of the Internet itself at all levels should be in the enlightened self-interest of any nation.

Proper architecture of a secure Internet will go a long way toward protecting the less secure legacy systems used by large segments of the economy. Better security provided by an improved communications architecture would reduce the risk of uncontrolled outside access to critical systems.

Endpoint Security

Information is only as secure as the endpoints of the connection. One cannot expect privacy if your correspondent takes screenshots of your illicit content. Absolute protection of content during transport is the province of the network design. Managing that content at the endpoints is up to the participants.

Concerns of local Authorities rightly target the behavior of their own individual citizens, and the endpoints that they control. Nefarious endpoints (individuals or corporations) are subject to local standards that vary widely depending on geographic location.

Communications architectures should ensure protection of content en-route and should absolutely prevent malefactors, or lazy law enforcement, from arm-chair access to wholesale network traffic.

Replacing HDCP

The High-bandwidth Digital Content Protection (HDCP) mechanism is designed to provide a secure tunnel between a source of protected content and a presentation device. The idea is to prevent unauthorized duplication of the content. HDCP must work in an offline mode and is based on a set of secret keys embedded within each presentation device.

It is likely that zealous content-protection hardware could be designed that encloses the entire Phase Identification mechanism and its parameters within a secure hardware enclave. Envisioning the content player as an endpoint would allow the stream elements to be securely directed to the only device capable of decoding and reassembling the content. This would allow a three-party negotiation between the “media player device”, the “media player user interface software” and the content provider.

The user interface would provide authorization credentials (i.e. subscription login) and scrubbing instructions. It would be a network endpoint and would remain in contact with the

content provider. The desired data stream would be directed to a different endpoint acting as the media player device. Negotiation between the content provider and the media player device node would establish the decoding parameters (i.e. Phase Identification parameters) required to recover the required data.

Ideally, this negotiation would be a zero-knowledge proof. The intent is to prevent the (nefarious) content provider from linking the (personally identifiable) identity of the media player device (i.e. the endpoint unique ID) with the login account and requested content.

The content provider may be a distant networked node (YouTube, Netflix) or a local "offline" copy of (e.g., purchased) content. The three-party negotiation implemented over the network would be at least as (in)secure as the stored-secret implementation used by HDCP, but considerably more flexible.

The workload on the servers of a large content provider can be drastically reduced by eliminating the redundant compression and encryption processes used for each user stream. Bulk content can be staged at multiple geographic locations and streamed to users with less overhead by using the network's Fragment Reassembly (Phase Identification) mechanism. The transport system will be secure; the security of the content will be as good as that of the endpoints.

Concerning Firewalls and Virtual Private Networks

Firewalls attempt to provide network security by examining IP packets and allowing only communication using approved protocols between approved endpoints.

Virtual Private Networks (VPNs) attempt to provide connections between anonymous endpoints using protocols whose identity remains hidden from outside observers.

Thus, Firewalls are defeated by VPNs and VPNs are only as secure as the VPN servers that act as intermediate endpoints. In addition, the number of routers involved in handling VPN traffic is frequently more than three times the number required for a "direct" route. Each of these introduces additional delays and provides additional opportunities for adversarial action.

Even though the link from the user into the VPN will be encrypted, simple traffic analysis can reveal the nature of the protocols being used, the number and nature of the requests being made and the size of the responses received.

VPN servers form choke-points that not only limit effective bandwidth but make easy targets for monitoring by malefactors. Anonymous VPN implementations such as The Onion Router (TOR) are even more vulnerable in some cases since they allow an adversary to interpose their own server into a connection path. The bidirectional nature of a TCP/IP connection ensures that all traffic, in both directions, will travel through the same TOR node. Successive connections via TOR may take different paths, but there is no provision for path diversity within a single connection.

Firewalls and VPNs are not a real solution to secure communication, and in fact merely add additional vulnerabilities and opportunities for exploitation. The expanded threat surface exposes traffic to additional casual and sophisticated adversaries. Every connection and piece of firmware must be considered suspect, and carefully evaluated, if security is to be maintained.

Naming Conventions and Local Area Networks

In the early days of the Internet it seemed like a good idea to assign each computer a unique IP address and create a lookup table to allow that address to be determined from a conveniently memorable name. This meant that any computer could directly create a connection to any other. Soon, however, it was discovered that many computers in a local network needed to be separately administered and that a global naming / numbering scheme was unwieldy. Thus the idea of unique IP addresses was expediently discarded, along with any hope of allowing any computer to create a connection to any other on an arbitrary basis.

Instead of recognizing this fundamental flaw in the Internet architecture and solving it conceptually the various committees built a structure that (1) required Server Names to be unique, (2) required connections to be initiated by a Client to a Named Server, and (3) required all intermediate routers to know and understand the concept of connections, maintain IP address translation tables for every connection that bridged network segments and rewrite the header of every message that passed the router.

Managing the Domain Name directory was recognized to be problematic so the concept of Top-Level Domains was created to allow multiple directories. An administrator would ensure that each of the resulting domain names was unique. The requirement for uniqueness is arbitrary and unnecessary. Everyone knows that you just need only type something close to a name into a browser address bar, Google will give you a list of candidates, and you can choose one. The lookup of a named computer and the required network route should inherently allow multiple possibilities and be fully automated at all protocol levels.

Secure registration, lookup and identification of particular servers can be made automatic. Thus, there is no reason that there should be charges for Domain Registration. Domain Squatting should not be a problem and detecting Domain forgery should be a routine part of the operation of the network.

The fallacy of this increasing, distributed overhead should have been obvious from the early days, but was hidden by the negligible cost of adding memory and processing power to individual routers. With the advent of the Internet-of-Things the idea that one might wish for a million devices to establish connections into a single server is not outlandish. These connections must be persistent, secure and allow the Server to initiate a message to any of the million Clients at any time. It is unreasonable to expect keep-alive traffic from every device and it is unreasonable to expect every intermediate router to need to maintain translation and routing information for each of these connections.

Three things should be true today:

First, every connected device should have at least one unique identifier (think of a GUID).

Second, every device should have the ability to publish its ID in one or more public or private directories along with (not-necessarily unique) convenient names and routing / accessibility information. This published information can include signed Certificates, public keys and WHOIS-style information.

Third, it should be possible for any device to be able to create a connection from one device to any other device with no information concerning the existence of the connection stored anywhere except at the two endpoints.

Enabling these capabilities is the primary goal of this paper.

Fixed-size Bit Fields

The use of arbitrary, fixed-size bit fields as elements of protocol headers has caused problems with every protocol. Arbitrary limitations introduced in this way are quite difficult to overcome and lead to layers of unnecessary overhead and increased vulnerabilities.

A very fundamental example of the flaws inherent in arbitrary field sizes is the concept of the character. The early, arbitrary decision to make the eight-bit byte the basis of the character set used by the network naming system has resulted in ever-increasing headaches. This decision seemed reasonable when the “only” character set was ASCII. Even then, the astute observer would see key flaws. The lack of even basic currency symbols for foreign transactions should have raised alarm. Now the bizarre implementations of “multi-byte character sequences” used in domain names, and CHARSET= designators in HTML headings have become necessary.

Fixed length IP addresses and fixed length port numbers are obviously problematic limitations. Less obvious are hop counters (TTL values), protocol numbers, segment numbers and packet lengths. A properly designed protocol should have no need for any of these concepts, much less be limited by arbitrary bit-conserving decisions.

All routing elements will have a finite number of possible connections between ports. This Crosspoint number represents the selection of a connection from one Input port to one Output port. During initial route discovery, this crosspoint number will be determined. A routing Coupon indicating this Crosspoint number may be specified by simply appending an appropriate sequence of bits to the route header. No disclosure of the identity of the particular ports or the “addresses” of the particular routers need be made. Furthermore, the identity of the Originator or desired Destination will never be present in a header. This Sequential Relative Routing (SRR) can completely replace the absolute address routing that plagues current implementations.

See the section on Bit-Efficient Serial Representation of Small Integers.

In addition, the values used in Sequential Relative Routing can be obfuscated and dynamic so that even messages over the same virtual connection between endpoints would have uncorrelated header values.

DNS, DHCP and Other Questionable Inventions

Configuration of a network and subsequent creation of arbitrary connections between nodes requires some conventions that have evolved throughout the history of the Internet.

In order to establish the names of particular nodes for human convenience as well as the operation of routing algorithms, the concept of the **Domain Name Server** (DNS) was introduced. The DNS is essentially an ad-hoc translation table relating arbitrary text names with numeric IP addresses. It turns out that the implementation is insecure, vulnerable, slow, limited and creates unexpected distributed overhead due to caching "recent" lookups. The top-down tree structure used to distribute "authoritative" changes to DNS entries, the slow propagation of changes, the ability of anyone sufficiently capable to change any DNS entry and the lack of entry verification or validation are all critical issues.

The **Dynamic Host Configuration Protocol** (DHCP) is an attempt to create unique identifiers for machines connected to local network segments. It reduces administrative overhead by assigning arbitrary IP addresses to each network port. Unfortunately there is no provision for human-friendly names or documentation of the arbitrary address assignments. The implementation also introduces the concepts of Leases, Renewals, Expiration, and other arbitrary, headache-inducing features. It also reinforces the concepts of Gateway Addresses for routers on the network segment, and arbitrary, administrator-assigned IP addresses for the DNS Servers.

As a matter of policy, no autonomous network element should incorporate timers or timeouts. Attempts to retry failed communications, for example, should be done at the Originator's behest. Isolated black boxes, trying to "do the right thing" will inevitably do exactly the wrong thing under certain circumstances. It is axiomatic that standard, one-size-fits-all timeouts and retry limits will be suboptimal and no one will adjust them after deployment. The interactions of multiple autonomous boxes like this form an exploitable weakness in the design.

Independent, uncoordinated timeouts and retries at different nodes can easily cause difficulties in one node to be reported as a failure by a completely different node. This effectively defeats any attempt to create error reports or logs. Most errors cannot reasonably be corrected by retransmission. Having independent nodes repeatedly trying a failed connection and expecting different results is the definition of insanity - and it is contagious.

The necessity of timeouts to release resources related to connection maintenance after a failure can be mitigated (as proposed throughout this paper) by simply eliminating any stored-knowledge of a connection in the first place.

The **Border Gateway Protocol** (BGP) is a method of exchanging routing and reachability information among systems sharing particular IP Routing Prefixes. In order to perform the

required operations, the Internet is broken into Autonomous Systems (ASs) which are assigned Autonomous System Numbers (ASNs), each of which are controlled by independent organizations that are responsible for routing within their particular Autonomous System. Configuration and interconnection between Autonomous Systems is handled manually. Routing and Reachability rules are exchanged among all ASs via a fully connected mesh network over TCP connections. It kind-of works OK - if you don't mind having your emails secretly routed through China.

Transport Layer Security (TLS) and the older **Secure Sockets Layer** (SSL) are methods of negotiating and enabling end-to-end encryption for a TCP/IP connections between a client and host. Both of these protocols use an elaborate message sequence to perform a handshake that negotiates a set of options, secrets, keys and certificates prior to beginning the actual data exchange. The subsequent data exchange uses the selected algorithm and keys which are unique to each session / connection. The number of fundamental flaws in this approach is surprising. First, the handshake sequence is unnecessarily long and significantly slows the construction of every secure connection. Typically only the Server is authenticated - the Client that initiates the connection provides no proof of identity. The unique keys for each connection must be maintained at each endpoint for the duration of the connection. There is no guarantee that the route traversed by the individual messages is the intended one. There is also no provision for path diversity for a given connection. Any dropped messages will typically trigger a full renegotiation of the encryption system. Thus, streaming connections exhibit unexpected failure modes and higher-level techniques for out-of-order message assembly are rendered ineffective.

As with all Certificate-based authentication decisions, the overall trustworthiness is based on trusting a particular authority. This typically involves a list of Trusted Authority public keys stored in the Client. All certificates are only as secure as the DNS service, which (as we have seen) is neither trustworthy or secure. Certificate expiration is problematic for long-lived connections since the certificate is only checked when the connection is established. Certificate revocation is problematic since there is no reliable location to query for specific revocations and doing such a query would add even more superfluous messages and extraneous connections to the handshake sequence.

Message Lengths, Fragmentation and Streaming

In general, the purpose of a telecommunications connection is to transfer an arbitrary block of information from an Originator to a Destination. The size of this block of information may be arbitrarily large or small, and may, in fact, be unlimited in length or duration. Additionally, this should be accomplished with high bandwidth, low latency and negligible error rates.

Real world considerations concerning increasing data transfer rates and limited computational speeds meant that a store-and-forward architecture for most computers and routers was appropriate. Once the decision was made that each router should store the entire message before sending it on its next hop, the question became "How big can a message be?" This led to the concept of the Message Transfer Unit (MTU) which is usually (arbitrarily) set to something

like 1500 bytes. As with any performance or capacity figure given in relation to computers, YMMV. With the addition of headers and error detection/correction bits the actual effective data transfer rates and overhead percentage of a particular physical link are revealed only through arcane divination.

In order to support a reasonable conceptual architecture at higher protocol levels the details of packing large amounts of data into small packets was pushed to lower layers in the OSI stack. The techniques of packetizing, transmitting, acknowledging, retransmitting when necessary, sequencing and reassembling message segments are all handled invisibly. This black-box, one-size-fits-all approach is perfectly adequate conceptually but becomes evermore complex and unwieldy as additional modern requirements are added. Most notably, making reliability a low-level feature precludes the possibility of higher-level, bulk error correction being used to handle communication errors and drop-outs. This becomes especially onerous with examples such as high-speed links with large latencies where a single corrupt packet causes large volumes of data to be redundantly transmitted after the long (round-trip) delay required by the NAK. Even a minimal error rate under such circumstances can make an OSI protocol stack unusable.

Modern data encoding, synchronization, correlation and forward error correction techniques make the effective bit-error rate negligible in most physical links. Of much greater concern is full-scale drop-outs and burst errors which frequently cause connection failures.

Advanced techniques such as path diversity cannot be implemented at low levels and require significant storage and computational power. Implementing bulk error correction and sequencing of large shards of data received from multiple sources in randomized order is reasonable to expect at the Destination endpoint. Once these high-level features are given, it becomes unnecessary to devote any resources at all to the inadequate, low-level implementations.

Traditionally, packetization has been used to address the following issues:

1. Data-rate adaptation between different links
2. Error detection
3. Error correction via coding or retransmission
4. Multiplexing connections or data streams
5. Routing data in virtual connections over physical links
6. Equitable sharing of bandwidth among multiple users.

Examples abound of protocols piled willy-nilly on top of other protocols as a New and Better Idea™ tries to patch a fundamental design flaw. A case in point is the use of H.264 for multimedia streaming. The goal is to encode a multimedia presentation containing multiple video resolutions, color depth and aspect ratios, audio including stereo and surround sound, descriptive captioning in various languages, and other features such as thumbnails, alternate camera views, commentary, and picture-in-picture. The advanced MPEG-like data compression and encoding of motion video is just one tiny, interchangeable part of the standard. Once the compute-intensive encoding process is completed, the resulting H.264 file can be used for things like Blu-Ray discs.

Now consider the goal of streaming a multimedia presentation over a telecommunications link. The H.264 format is one giant file with a bunch of interspersed slices: each slice encodes portions of content data, algorithm parameters or control information. Much effort goes into making it possible to scrub forward and backward within a presentation and get the different media properly synchronized for presentation. Data-rate adaptation and display resolution adjustments make it possible to handle such situations as arise when using mobile devices over poor-quality connections.

The problem with all of this is that the H.264 file has all of these slices randomly strung together in a wonderfully bit-efficient manner. This makes it easy for a Blu-Ray player with almost no memory or processing power to "kind-of" play the video. Conversely, it makes it quite difficult for a streaming server to respond intelligently to user's screen sizes, scrubbing requests and dynamic variations in network performance. All of the different media types are separated into different slices and each have CODEC operating parameters and state information that must be obtained before any presentation can occur. This means that any selected point in the presentation requires obtaining an arbitrary number of slices from an arbitrary set of prior points in the stream. And there is no ability to request or index these slices: the only possibility is to "back up a 'ways and take a run at it". Sometimes this works; sometimes you wind up with audio out of sync, corrupt video, frozen images, etc.

The entire concept of multiple, independent data streams, interspersed over a network, each with independent rates, error correction and indexing is what telecommunications protocols should be good at. And yet, here we have another half-baked implementation trying to function on a flawed foundation.

The Fallacy of the Address

There are four fundamental concepts that are required to create an Internet. Casually speaking, we can describe them as follows:

Identity: Basic, immutable, unique identifier for a computer, node or server
Name: Friendly name that may be associated with a server
Address: Designator that allows sending messages between computers, node and servers
Route: The path between two endpoints that traverses multiple nodes

Misunderstanding the importance of each of these, and conflating their nature in the early days has let to most of the current Badness Of The Internet™. The casual descriptions above hint at the problem: there is no rigor in the definitions. In fact, the early internet had no actual concept of Identity. Any computers that answered at a particular IP address were deemed to be equivalent. Even modern implementations can mistakenly assign the same IP address to multiple computers, leading to faults, connection hijacking and leakage of information. Due to the geographical and administrative structure of IP addresses there is frequently an intentional reassignment of previously used addresses. Any reuse at all affects logs, caches, firewalls, routing and all aspects of security.

Confusion abounds when the very structure of an Address gets associated with or implies particular regions or directories. Limitations within such conflated identifiers make it impossible for a particular destination to retain its address as network configurations change.

The presence of multiple Name Servers for redundancy, expedience and optimization means that name lookup failures and conflicts will occur. The lack of tolerance of such failures and the expectation of perfect performance from the Name Server system - even in the presence of active adversaries - is a serious design flaw.

The requirement for an unambiguous, unique Identity for every node was not recognized during the formative years of the Internet and is still not a feature of any end-to-end networking system. Attempts to prevent conflicts based on MACs or ESNs are stopgap measures implemented casually within the Physical or Data Link layers of the OSI model. These surprisingly short unique identifiers have problems with the infrastructure used to assign them to specific hardware, their ease of forgery, the impossibility of validation or revocation, etc. Further, the realities of administration result in the assignment of blocks of numbers to particular manufacturers. This allows an adversary to obtain information about the physical hardware implementation of a particular port and more easily look for potential vulnerabilities in a specific company's drivers.

Ideally, the Identity should be a cryptographic public/private key pair. This would allow secure interrogation of the public identity of a particular element without the possibility of forgery or accidental duplication. These Identities should be able to be assigned to hardware devices, ports, physical and virtual links and ephemeral connections. Any of these elements may be the owner of multiple Identities as required. Identities may be generated as needed and discarded when compromised or no longer needed.

A key concept is that end-to-end encryption is insufficient. Modern security based on SSL/TLS is designed to ensure that particular messages reach their destination verifiably unaltered, unread by third parties and with authentic sender and recipient. The two parties have no ability to control (or even know) the path taken by their messages. The ability of all intermediaries to log information relating to the two endpoints, the volume of traffic, the times and rates of transmission and the other concurrent connections for each endpoint reveals far too much to a skilled adversary.

The original idea of a Domain Name Service - allowing a name to lookup an IP address - obscures a deep-rooted fallacy. The idea that I can reach any computer by just knowing its IP address ignores these facts:

1. Local Area addresses (e.g., 192.168.*.*) are not reachable from outside the LAN,
2. A web site located on a shared server requires both an IP address and full URL.
3. Security certificates are only issued to URLs, not IP addresses

Failure to Learn From the Post Office and Phone Company

The Post Office has a long history of performing message delivery between arbitrary endpoints. The only requirement for mail delivery is a valid, plain-text destination address.

1. Messages may originate a virtually any point: an ordinary endpoint address, a post office (intermediate node) or even be handed to a letter carrier en-route.
2. Messages may be addressed to a physical address or an individual at that address.
3. Messages may held for pickup at a post office, either via a specific, temporary request or a P.O. Box contract.
4. Messages may encounter forwarding instructions from one address to another.
5. Messages may be deemed undeliverable and returned to an OPTIONAL return address
6. Messages may have associated return receipts handled automatically by the Post Office
7. Messages may require identity verification before delivery to a particular address
8. Addresses are variable-length text strings and are extremely fault tolerant
9. Addresses generally represent a hierarchical approximation of geographic locations but particular elements such as street names or ZIP codes may change over time without affecting delivery of messages using older notations.
10. Different classes of mail can have different associated costs and delivery latencies
11. All deliverable addresses are codified in a master directory that maps addresses to Zip+4 numbers
12. The size of an actual message is arbitrary, essentially unlimited and, in general, affects only the (prepaid) cost of message delivery
13. Postmarks not only confirm payment for message delivery but establish the point in time and space at which the message entered the system

Shortly after Don Ameche invented the telephone, a network of electrical connections between endpoints and central offices arose. The central offices allowed direct connection between any two local endpoints. Extended connections could be established by involving a hierarchy of regional offices covering longer distances.

1. Early operator-mediated calls could be placed by verbally naming the person or business desired
2. Phone numbers were created to simplify the operator's selection of holes in a plugboard.
3. Minimum-length phone numbers with no check digits or other error detection mechanisms led to frequent wrong number situations
4. Multi-drop implementations reduced wiring costs (party lines) and led to signaling conventions to resolve different desired endpoints along a single cable
5. The introduction of mechanical switching offices (exchanges) using relays allowed customers to directly enter a desired phone number using a pulse-coded representation
6. All calls were point-to-point, bidirectional and real-time. Connections that could not be established resulted in a variety of busy signals or intercept tones.
7. A small number of switching registers were shared among a large number of physical phone lines. This led to the possibility of Denial of Service when too many callers attempted to dial at the same time
8. Phone Directories containing name, address and number were published to allow customers to look up the required number themselves. Duplicate names could be resolved

using the published addresses. Operators used Soundex directories to help overcome the problem of variant name spellings and sorting conventions.

9. Phone numbers were fixed-length and adding more capabilities was very disruptive to both users and infrastructure. Interoperability between offices was problematic.
10. Provision for dialing between exchanges introduced variable-length phone numbers but this required the ability to designate and recognize the difference between the 'exchange' and 'number' parts. This was done through numbering conventions that varied by region.
11. The invention of the transistor enabled direct distance dialing. Direct distance dialing was implemented by the creation of Area Codes.
12. International calling was implemented through country codes. The format and validity of the digit string following the country code is completely up to the individual country.
13. Connection costs are billed based on distance and duration in a combination of prepaid and postpaid options.
14. Pay phones allowed totally anonymous pay-as-you-go connections.
15. Private Branch Exchanges (PBXs) allowed for the addition of hundreds or thousands of 'extensions' to each phone number, and calling between extensions within a company.
16. The Hold option to allow a user to perform multiplexed connections
17. Voice Mail as an option to replace the Busy Signal, and convert the real-time connection to store-and-forward messaging
18. Caller ID to support call screening, voice mail and call return, although it does not work from within a PBX.
19. Arbitrary decisions regarding dialing formats within PBXs, locally, internationally and regarding '0' for Operator, '911' for emergencies, etc. all stem from the original failure to include delimiters or end symbols in the actual dialed number.
20. The introduction of wireless technologies and cell phones make it obvious that a phone number is more likely related to an individual instead of a physical location
21. Local Number Portability eliminated any lingering traces of a link between geographic location, service provider or endpoint connection technology and a particular number. This implied that all call origination must go through a master directory of all possible numbers in real time.
22. An additional coded option allowed a Carrier Identification Code to precede the number in order to force selection of a specific long distance carrier. Further optional digits could specify billing account numbers or prepaid calling card numbers (with PIN)
23. Additional signaling options for landline service allowed for hold, call waiting and conference features.
24. Cellular phone operators have essentially eliminated the concept of a phone directory to look up a number given a name, and have ignored the possibility of delivering a standardized text name with the Caller ID to the recipient.
25. Contact Lists maintained by individual users have replaced phone directories, eliminating name standardization, increasing user workload and masking unknown callers.
26. Efforts are made to confirm identity using SIM cards and an IMEI, but these are hidden from the customer and rife with fraud.
27. The lingering obsession of the Phone Number as a (small) integer has led to the rise of Robocalls and spam based on a simple "what if we just dial all the numbers?" philosophy
28. Call Forwarding is a common feature of the network. It is also possible to have a call forwarded only if there is not a prompt answer at the original number

29. Call Return can be used to connect to a caller after a missed call
30. Redial can be used to re-establish the last connection, or to retry a failed connection
31. No ability to redial or reconnect to an extension within a PBX has ever been implemented
32. Calls from a particular source can be selectively blocked or forwarded
33. The PIN Register has been shifted to the cell phone which allows for backspace and a number-ending SEND button, but does not address the requirement for delimiters in segmented numbers for extensions or voice-response trees.
34. There is no provision for signaling “call me back at a different number” or “I am calling on behalf of…”
35. Modems allowed conversion of analog voice connections to digital for the transfer of data and images (the Fax machine)
36. The Short Message Service (SMS) adapted features of the call setup signaling system (SS7) to implement datagrams that require no actual connection between parties.

Each of these points represents a valuable insight into a feature that might be useful in the design of a modern telecommunication network. Too often expedience and low-level “techie myopia” prevent reasoned analysis and careful implementation of features that would actually be useful in the long term.

Proposed Implementation

The intent of this paper is to provide a vision of a reorganized and minimalist Internet based on experience gained and lessons learned over 40 or more years of telecommunications.

We recognize that

What Grove giveth, Gates taketh away.

We wish to reset at least portions of both hardware and software to a much simpler, more robust and sustainable state, suitable for future growth and inherent security.

It is expected that a 16-port gigabit switch could be fully implemented in a single FPGA. This switch would be capable of fully autonomous operation after setup by a Routing Processor (RP). The burden of creating and maintaining a virtual connection from an Originator to a Destination server would be shifted entirely to the Originator and ensuring a high level of security for all communication.

Consider the following individual statements in order:

1. It is possible for the Originator to determine the path of a given Server Request *a priori* using TRACEROUTE or by interrogating an omniscient service.
2. It is possible to specify the sequence of routers by name or IP address as part of the prefix to messages.
3. There are a limited number of direct connection paths to any individual router.
4. The particular router-to-router link for each step may be specified in a compact manner.
5. Communication between the Originating endpoint and any router in the path may be encrypted based on *a priori* negotiation between the two.

6. It is possible to ensure that the routing instructions for each step are individually encrypted.
7. It is possible to intersperse Route Negotiations, Encryption Handshaking and Server Requests into the sequence of consecutive routing instructions.
8. Responses from a server request need not reverse the path taken by the request - they may take an alternate path or none at all.
9. It can be made essentially impossible to determine the number of hops between the originator and the target server by encrypting variable-length step specifiers and adding padding to the message headers.
10. Route Negotiations between an Originator and particular Router may include alternative routes with different properties such as speed, latency, congestion, etc.
11. The Originator may select particular paths based on its own criteria - completely independent of the local network view of any particular router.
12. The Originator may store path headers for particular targets and reuse them as required. New negotiation will only be necessary when the network topology changes along a route.
13. Connections may persist for days or months with no keep-alive traffic to expose the presence of the connection.
14. It is unnecessary to ever identify the Originator of a particular message. Since all routes are explicitly specified it is possible to create a message that appears to be sent “on behalf of” some anonymous previous set of hops. Further, the response may reach its actual recipient before the end of the routing instructions. Thus, the encrypted, anonymous Originator may instigate round-trip communication while leaving no obvious identifiers.
15. It is possible to implement multiple consecutive Requests to the same Server using completely different message routing.
16. It is possible to ensure that message routing stays local, avoids the Internet Backbone, international cables or government monitoring points. Further, the traffic (including headers) can be encrypted, anonymous and fragmented.

Recognizing that each of these statements are true in an abstract sense, we can work to design a networking architecture that implements them. This new architecture will recognize the knowledge and experience gained over forty years or more of trial and error. History provides guidance and advice but does not dictate features or design processes.

All modern messages should be presumed to be (1) compressed and (2) encrypted by the Originator. This means that the actual data transferred over the network is an arbitrary sequence of bits that should be indistinguishable from random noise. In fact, deviations from randomness most likely indicate flaws that can be exploited by an adversary. Secret knowledge, prearranged with the recipient should allow recognition, synchronization and decryption of the received bitstream. Intermediaries and adversaries should not have this capability.

Any universal protocol features such as byte (octet) orientation, fixed length fields and explicit or implied data lengths represent vulnerabilities and should be excluded. Since the recipient should be able to autonomously recognize its own intended bit streams, there should be no boundary or length information of any kind visible to the transport or delivery mechanisms.

This implies that actual messages should be able to be bracketed by random noise, or other messages. Message corruption, duplication, or missing sections, should be detectable by the recipient and dealt with in a manner of its choosing.

Any retransmission requests would likely be made only after the recipient determined that its highest level error recovery mechanisms had proven ineffective. It is well known that known-plain-text attacks represent critical vulnerabilities to the security of cryptographic systems. Therefore, requests for missing segments of a message should cause the sender to select overlapping regions of the plain-text message and perform new compression and encryption before sending the response bit stream. I.e., the retransmission should be indistinguishable from a continuation of the ongoing data transfer. Only the intended recipient should be able to recognize a retransmission for what it is - after reaching the highest protocol level.

Routing, transfer rate, bit error rate and latency are all things that should be available for monitoring and control by the Originator. Current implementations assume that independent "best effort" decisions made by autonomous modules will yield an acceptable result. The provided level of connectivity is what you get. Take it or leave it.

In its most idealized form a router should be a device with multiple ports, each capable of sending or receiving a serial sequence of bits. A particular sequence of bits sent into one port should emerge from another. We make no assertions about the latency involved, or the rate of bits as they enter or leave the router. We simply wish to assure a high correlation of the input sequence with the output sequence without regard for timing. Stating this as a "high correlation" allows for a certain error rate, and for noise, dropouts and fragmentation. Dealing with such real-world issues is not the province of the idealized router.

Interoperability with Legacy Implementations

It is recognized that (short of a legal mandate) wholesale replacement of equipment using legacy protocols will not occur. Therefore the focus should be on implementing (1) new networks (such as spacecraft data links, or design for space colonies), or (2) private networks and backbone infrastructure that tunnel legacy data structures.

For cases of new LAN deployments the selection of protocols and the completeness of the implementation will be under the control of a single organization and should be straightforward.

For the case of backbone additions, the requirements would include bridging Routers with the ability to support a combination of legacy ports and full protocol links or fabric links. These could be installed in any ISP or backbone data centers. The advantages would include faster and more stable transport over new or existing links. Bridging Routers would be installed in these data centers and would act as the Originator and Destination elements for transport. Legacy packets would be bundled and encapsulates within larger Messages bound for a pre-configured Destination. This eliminates intermediate legacy routing and provides high-speed, direct connection to a distant Destination. The operation would be conceptually similar to the drivers currently used to aggregate traffic for transport data over long-latency geostationary satellite links.

In order to provide full security it is necessary to provide access to the new protocol at the endpoints. It is intended that the existing low-level physical hardware, such as Ethernet cards should be adequate. Replacing the driver software and protocol stack should provide access to the full, secure capabilities expected of an Originator or Destination. The addition of proper Routers would extend the security as far as possible from the endpoints. The use of Bridging Routers to allow secure traffic to enter the legacy network will be a primary point of vulnerability. Conceptually, this is exactly the same as the VPN situation where the attacker would focus on the entry and exit points of the tunnel.

Ground Rules

Several overriding principles guide this new design.

Everything is a Bit Stream: No "Bytes" or artificial Field Lengths; unlimited Message length.

All Routing is Relative to the current node: no absolute targets.

The Originator has control of the Path - forward and return.

Connections are known ONLY to the endpoints: no intermediate storage.

Name Service is Distributed, peer-to-peer.

No Timeouts at intermediate nodes.

No Exploitable Data or Traffic Patterns visible along the path.

Fragment Reassembly ONLY possible at the Destination.

A quick overview of each of these principles is provided here. The remainder of this paper is devoted to a more in-depth presentation, details and examples.

Everything is a Bit Stream. By eliminating fixed field lengths and artificial length limitations we can make an extensible design that is more flexible and free of the need to add kludges as unforeseen requirements arise. Data compression such as that found in streaming video is already independent of byte or word alignment requirements; we simply extend this to all data.

All Routing is Relative to the current node: no absolute targets. We eliminate IP addresses from protocol headers. The entire header is replaced with a list of Sequential Relative Routing coupons. The resulting header will be approximately the same size as a legacy header, but considerably more flexible.

The Originator has control of the Path - forward and return. DNS Lookups already place the bulk of connection setup at the beginning to be handled by the Originator. We simply extend this requirement to the complete path and eliminate all intermediate route lookups from the bulk of message traffic.

Connections are known ONLY to the endpoints: no intermediate storage. Eliminating the ability of intermediate nodes to alter the path of a connection also eliminated the requirement that those routers store connection information or rewrite headers based on the existence of a connection.

Name Service is Distributed, peer-to-peer. Name Servers store and publish DNS, WHOIS and Route information as requested by particular hosts. These published records are individually verified, unlike the current, insecure DNS implementation.

No Timeouts at intermediate nodes. The protocol needs no timers except at endpoints so no arbitrary timeouts can affect performance in unpredictable ways.

No Exploitable Data or Traffic Patterns visible along the path. All Message Headers and Data are encrypted and obfuscated to prevent disclosure to malicious intermediaries.

Fragment Reassembly ONLY possible at the Destination. Fragmentation of long Messages may occur at any point along the path. The ability to reconstruct the original Message is possible through the use of inserted Phase Identification bit sequences.

Dirty Tricks

Virtually any network feature has the ability to be abused by a skilled and knowledgeable adversary. Automatic handling of such attacks so as to minimize disruption or data leakage is a primary goal of any new design.

Denial of Service: As a simple attack, it might be possible to try creating a message requesting a large traffic volume such as streaming video be sent through a particular router port representing the connection to the target. This would cause a burst of unexpected data at the target but no handshake would occur so the video source would quickly recognize the path failure and would not transmit further.

A more sophisticated version could involve using path diversity to direct only a portion of a video stream against the target while accepting just enough to be able to acknowledge the

connection and keep the stream going. This represents only a minor amplification of message traffic and would not typically be sufficient to overwhelm a server port.

Mitigation: It is anticipated that each router maintains a number of encryption keys that are issued in a randomized fashion to each requestor during Route Negotiations. This means that it is unnecessary for the routers to know anything about the Originator or Destination of any message in order to handle a particular encrypted routing instruction. Upon detection of a route overload the router will be able to identify the particular key that is being abused and immediately void it. All traffic received at any router with invalid routing instructions is discarded. Voiding a key may pose a slight inconvenience to other connections that had been assigned that key and will require them to request a new key if and when they discover the connection has been compromised. This is exactly the same situation that would occur if a router was physically replaced or a link became unusable, so it should look like a normal part of network operation.

Large Message Sizes: An attacker might create a very large message and send it along a path in an attempt to saturate multiple routers. Even in the worst case routers will proportionally distribute messages from inbound ports to the outbound port. This will mean that no message can use more than 50% of an outbound port's bandwidth if there are any other sources requiring that port.

Loopback Abuse: One could envision an attempt to create loops or repeated passes of a single message through a target Router. Conceivably a single set of repeated Coupons could impact a particular Router. This would reduce the available bandwidth at a particular port as the malicious message uses increasing amounts of elastic buffer memory and port bandwidth. If a buffer overflow occurs internally the message will be discarded and operation will return to normal.

Unbalanced Data: A malicious Originator could compose a data stream consisting of all zeroes or all ones with the intention of biasing the data discriminator and causing intermediate routers to fail to recognize the header of the following message. The protocol expects bit streams to be generally random and run-length limited. When each router initializes a physical link to another router, part of the initialization includes sharing a whitening polynomial. This polynomial is used in a linear feedback shift register to generate a pseudo-random bit stream which is XORed with all link data. The whitening register is reset after every **START** or **STOP** marker. The polynomial, and hence the whitening sequence, is different for each physical link and will be unknown by the Originator.

Lying Routers: When requesting Route and Reachability information from a Router acting as a Name Server, the Originator expects honest answers. The path and Coupon list should honestly reflect the desires of the Originator. Well-behaved Routers will have verified the Name announcements that it is publishing. Misbehaved Routers may publish malicious or misleading answers to enquiries from specific Originators. The intent of this design is for all Originators to be anonymous to every intermediate Router along a path. The ability to support private networks over shared Routers means that requests from some Originators may include subscription information that would enable access to the private links. Examples would include

the ability to route privileged traffic via a direct link and other traffic more circuitously (or not at all). In any case, the Originator will discover that a route is incorrect when the lookup request from the next Router fails, or yields incompatible information. As a rule, any time an Originator reveals private information (such as the subscription credentials) to an intermediate Router he is explicitly violating the security design of the network. As we repeatedly emphasize, the security of all communication is in the hands of the Endpoints. The design provides that there will always be a secure alternative.

Bit-Efficient Serial Representation of Small Integers

Analysis

When considering bit-oriented operations one inevitably returns to considerations of Turing Machines. When one discusses Turing's representation of integers one envisions something like this:

0	0
1	10
2	110
3	1110
4	11110

and so on. For the conceptual purposes of a Turing Machine this is perfectly adequate since no consideration of efficiency or optimization was needed.

The problem that we encounter in this paper is that we would like to have integers of arbitrary size be represented in a compact manner without any a priori limitations involved in the definition.

We want to avoid a "length field", because the size of the field would have to be pre-defined.
for example: length n-bit binary number

We want to avoid a fixed "symbol size" such as 4-bit or 8-bit characters, even though the use of a unique terminal symbol might be tempting, as in the Unix null-terminated string. Avoiding a terminal symbol from within the defined character set also avoids the choice of something weird (like base 127 encoding) or escape characters, which require their own syntax, as well as escaping-the-escape characters.

Bear in mind that for these serial representations we are not interested in obtaining any particular numerical value from the representation. It will suffice to be able to compare two bit strings for equality, and possibly make unambiguous comparisons for greater-than and less-than. This implies that the bit-serial representation of a particular value must be unique. We are not interested in performing other arithmetic operations on the values. We will need to be able to perform bitwise operations (primarily XOR) to overlay pseudo-random codes and recover the data afterwards.

Considering the Turing case above we can describe the representation as 0-terminated. This is a limiting case of a more general class of pattern-terminated strings. For example, the terminal pattern could be 00, as follows:

0	00
1	100
2	0100
3	1100
4	01100
5	10100
6	11100

This case essentially allows any binary numbers that do not include the pre-selected terminal pattern.

Another possibility is the Nth-zero representation. The Turing example above could be called a first zero termination representation. Consider a second zero form:

0	00
1	100
2	010
3	1100
4	1010
5	0110
6	11100

Note the interesting tradeoffs in bit efficiency in the representations for different values.

Run-length codes are a family of different encoding patterns that can be used for serial data streams. Traditionally they are used to create an encoding that ensures clock recovery. Examples include Manchester encoding, and various Frequency Modulation schemes used for hard drive data separators. Synchronous High-Level Data Link Control (HDLC) also uses a run-length limited representation with bit stuffing to provide clock recovery, as well as the flag for frame separation. Examination of run-length limited representations allow us to use a common methodology for the integer-coding problem, as well as the Message encoding with the required Markers described here.

Resolution

Run-length limited binary representations present the best encoding for our purposes. In particular, inserting a single bit after a sequence of four consecutive ones or zeroes retains the advantages of true binary representation while allowing a variable length representation. This allows for terminal symbols of either five zeroes or five ones.

The rules are as follows:

1. Zero is a special case: Represent it as only the terminal symbol **11111** and we are done.
2. Represent the integer value as little-endian binary with high-order zero suppression
3. The rightmost bit of the value will always be a one, so it can be implied. Discard it.
4. Scan the representation from left to right and insert bits for run-length limiting:
    1. Any **0000** becomes **00001**

    2. Any **1111** becomes **11110**
5. Append the terminal symbol, chosen to be different from the final bit in the value:
    1. Ending **0**, append **11111**
    2. Ending **1**, append **00000**

Here are a few selected examples of encoded integer values. Spacing has been added for clarity to indicate bits inserted for run-length limiting and before the terminal symbol.

```
0:    11111
1:    00000
2: 0 11111
3: 1 00000
4: 00 11111
5: 10 11111
6: 01 00000
7: 11 00000
8: 000 11111
15: 111 00000
16: 0000 1 00000
17: 1000 11111
31: 1111 0 11111
32: 0000 1 0 11111
33: 10000 1 00000
63: 1111 0 1 00000
64: 0000 1 00 11111
582543:  1111 0 000 1 111 0 000 1 111 0 000 1 00000
```

The last value is a 20-bit binary number that uses the maximum possible number of inserted bits, resulting in an encoded value of 30 bits in length.

This is the worst case - values between 2^0 and 2^{20} require an average of about 24 bits.

Routing Coupons

The applicability of these observations comes from the fact that we would like our routing header to have these properties:

1. Compact sequence of route selection instructions.
2. Each instruction encrypted/obfuscated by unique relationship between the message Originator and the particular Router.
3. The size of the route selection instruction used by any particular router should be variable and change in an unpredictable manner with the encrypted values of the route selection instructions.

This all means that the routing header should be bit-efficient but variable in length and it should not be possible to arbitrarily break the header into individual routing instructions (or even determine the number of instructions/hops in the route) without actually following the route to each individual router. We must ensure that Messages arrive on the correct physical switch port. The Originator and individual Switch have previously exchanged a randomization value when the path was first established.

The route that a particular Message should take is defined explicitly by the Originator and encoded into the Routing Header.

Each Routing Instruction must be variable-length and encoded in a manner that can be rapidly and unambiguously evaluated by the individual Switch.

Invalid Routing Instructions cause the Message to be immediately discarded with no further action taken.

It is expected that initial negotiation between the Originator and each Router will establish a particular Forward Coupon (and possibly a Return Coupon) for the desired hop along a particular path. The Router may provide a set of Synonym Coupons, each with a different value, that will resolve to the same Switch connection (crosspoint number) when presented to the Switch. The Switch will maintain an internal list of current, valid Coupons which are essentially random selectors into a large, sparsely populated, address space of potential Coupons.

Note that the possible valid Coupons are created during Router initialization and are not specific to any Originator or connection. The same Coupon value may be shard by any Originator that creates traffic that enters the same Router port. Thus, there are no per-connection table entries and there is nothing that can uniquely identify a particular Originator stored in the Router.

The Originator can place a Coupon for a particular hop into a Message Header that is variable length, can be decoded only by the specific router, that has a large ratio of invalid-to-valid possibilities, that can have a large number of apparently random equivalent values, and that can have their validity voided by the router at any time.

Coupons, Ports and Sockets

TCP connections establish a relationship between a logical IP port number and the TCP socket that is purely internal to the protocol stack. This structure limits the total number of possible simultaneous connections at a particular IP address to 65,535. It is not possible for anyone outside the endpoint stack to access the actual socket information.

Message Routing Coupons have no such limitation. When a Router delivers a Message to a particular Endpoint (Client or Server), the next Coupon can be interpreted by the Endpoint Processor as a port number, socket number, or anything it desires. Thus, there are an unlimited number of possible connections.

This use of Routing Coupons by the Endpoints can also be used to solve the problem of multiple web sites hosted at the same IP address. The current implementation requires the explicit inclusion of the target URL within the HTML Request header. This is an operation completely outside the message handling design of IP or TCP. Another *ad hoc* solution that obscures an underlying deficiency.

Temporary Authentication and Persistent Authentication

The current Internet connection model exemplified by TCP and TLS expects security credentials to be verified at the beginning of every connection. Since connections are often extremely temporary (one small file transfer request and response) this credential verification is often repeated unnecessarily.

Ad hoc implementations of a more persistent authentication model are often used for sign-on to web sites and services, but are completely independent of the underlying TLS authentication, which knows nothing about user credentials.

True persistent connections with full security and authentication of both users and services can be implemented using the message routing provided by Coupons. Once negotiated, the logical connection between Originator and Destination (Client and Server) can be maintained for essentially unlimited intervals. Credentials need only be re-verified as a matter of policy, not because of deficiencies in the network.

Alterations in the routing path, such as network reconfigurations, will be detected and handled by the Originator. Credential and site forgery will be possible only during persistent connection setup (like a sign-on), not, as is currently the case, with every temporary TLS connection. It is reasonable to expect a more thorough vetting of credentials (such as checking for certificate revocation and verification of public keys by multiple sources) if new connections are established only rarely.

This all means that video conferences and streaming movies can be established once, paused for hours or days, and resumed instantly. Low data rate devices as part of the Internet of Things can be always online and instantly accessible without incurring any overhead on the network during idle times.

Originator Sequence of Operations

It is expected that the originator of a plain text message (PT) will perform the following steps.

1. Compress
2. Encrypt
3. Add Error Detection and Correction
4. Add Phase Identification bits

This will create a version of PT that is suitable for inclusion in one or more Message Data (MD) fields. This will not necessarily be the entire message - perhaps it is a large file, a set of real-time data points, or a live or stored streaming video. It is only necessary to have the portion that should be immediately transported. In particular, the Phase Identification Bits form a pattern that continues from the initiation of the connection. The Phase Identification mechanism allows for correct reassembly of fragmented messages. Dynamic Fragmentation may occur at any router and at any point in the message.

Since there is no pre-defined packetization or sequence numbering, the Phase Identification mechanism is used for retransmission requests in the case of dropped or corrupted fragments. A “Say Again” request specifies a general area within the sender’s buffer, based on the Phase Identification values. The retransmitted bit stream will have the same Phase Identification bits, and are used to provide bit alignment within the received stream. There will undoubtedly be overlapping data from adjacent fragments. This overlap can be used to help ensure placement accuracy.

5. Create the Message Header (MH) based on the desired route and final Destination using selected Route Coupons
6. Apply pseudo-random polynomial to the MH and MD elements
7. Perform Run-Length Limiting of the MH and MD elements
8. Insert the required **START** and **STOP** markers.
9. Send the resulting bit stream to the physical port when it becomes available

Multi-path and Stored-path Capability

The Originator has full control over the path that both Requests and Responses follow.

Unlike present systems in which no node has any control whatsoever over what happens to a packet once it is transmitted, the Originator-controlled model provides significant flexibility. Instead of relying on highly localized, best-effort, routing decisions, an Originator can ensure that Messages flow through specific Routers and avoid others. Instead of arbitrary Time to Live (TTL) values as a best-guess for the number of hops in a route, the Originator will know with absolute certainty which Routers will be involved.

During Route Discovery, an Originator will create a list of Routers to traverse to reach the intended Destination. This will often form an interconnected web, not some idealized tree structure. Thus, the Originator has the option of choosing multiple paths.

When creating its logical connection to a Destination the Originator may actually create a multitude of Request and Response paths and select from these randomly as required. In addition, the setup configuration with the Destination may allow a list of pre-selected return paths to be stored in the Destination. These would be used as replacements for the Header Suffix when sending Responses.

Route diversity implemented in this way will eliminate choke-points, obscure the existence of connections, and defeat traffic analysis by an adversary.

Router Operation

Routers have a number of physical ports which are bidirectional serial data links to either endpoints or other routers. The bidirectional nature of each physical port and link is used during initialization and setup to communicate operating parameters for that port. Subsequent operations transfer routed messages in each direction completely independently of traffic in the

other direction. In the event of a link failure in either direction, both directions enter the reset sequence.

Physical ports on a router need not operate at the same speed or even use the same media.

It is envisioned that the physical links provide for serial transmission at possibly dynamic bit rates. To that end, the encoding mechanisms described here are inherently self-clocking. Each receiver can track the actual signal edge timing to adapt in real-time to clock drift, jitter, etc. During link initialization bit rates can be dynamically selected based on actual media performance, including such things as measured bit-error-rate.

A router will have a Route Processor (RP) that handles messages that are requests for identity and connectivity information. The RP generates appropriate responses, including necessary routing Coupons, to allow an Originator to build Message Headers for future messages.

A router will include a Switch which enables virtual connections between any input port and any output port. Every possible input/output combination (crosspoint) is assigned a unique number which is used as the basis of Routing Coupons.

Coupon Types		
Null	Message used during Link Setup	Goes to Route Processor to aid in configuring physical port
Connect from Port to Route Processor	Originator wants to communicate with Route Processor	Used during Originator's request for identity and connectivity verification
Connect from Route Processor to Port	Response to a request	Message to be sent to Originator as response to identity or connectivity request
Connect from Port to Port	Ordinary message routing	Simplest Coupon format. Usually used only for Fabric Coupons
Connect from Port to Alias	Ordinary message routing	Multiple parallel physical links between routers. Alias means to select first available from group.
Connect from Port to Synonym	Ordinary message routing	Multiple synonyms for a particular Port to Port connection allow Originator to obfuscate each Coupon
Invalid Coupon	Discard message	Either (1) coupon not in switch table, or (2) Port does not match the actual From Port number

During initialization the Route Processor identifies the configuration of each port on the switch and fills in values for the hardware lookup tables used for establishing switch connection. Appropriate values will be sent to Originators as requests for connections come in from the

Originators. Once configured by the Routing Processor, the operation of the Switch is effectively autonomous. All traffic through the Switch is handled without involving the RP. The RP handles identity and connection information requests, and Switch configuration changes (perhaps caused by link failures).

Synonyms may be used to allow a given Originator to arbitrarily select from a set of possible Coupons for each switch connection. This helps to obfuscate the Message Header and makes multiple headers in a particular connection unique.

Aliases are a mechanism used within the network to allow parallel physical links between switches. This allows improved bandwidth to be added dynamically and independently from active connections or routes.

Physical ports on a switch may be configured either as Line Ports or Fabric Ports. Line Ports provide either Endpoint-to-Router or Router-to-Router links. Fabric Ports provide Switch-to-Switch links.

From an implementation standpoint the use of Fabric Ports allows additional Switches to be added to a Router to increase its capacity. It also allows Switches to be geographically distributed and still be part of a single logical Router.

Message Formats	Example
Disconnected	No signaling indicates disconnected cable or failed switch
Uninitialized Fabric Link	**STOP STOP STOP STOP**
Link Setup	**START START STOP** parameters **START STOP**
Idle Link	**START STOP START STOP** …….
Message	**START** prefix **START** coupon suffix **STOP** message **START STOP**
Fabric Routing	**START** coupon prefix **START** suffix **STOP** message **START STOP**
Illegal / Undefined Pattern	Idle waiting for marker

As defined here, an Idle Link can be indicated by a repeating sequence of **START STOP** markers. This is the simplest implementation but would make the link vulnerable to traffic analysis by making actual messages obvious. A proper Switch implementation would generate random bit sequences for Header and Message Data and send them over idle Links. The normal operation of the receiving Switch(es) would reveal this gibberish for what it is and discard the pseudo-Message. The use of pseudo-Messages will ensure that the ports appear busy but will not affect legitimate traffic or available bandwidth.

It is likely that the use of pseudo-Message generation will be suppressed on links to mobile or battery-powered devices due to power consumption and wireless shared-bandwidth constraints.

Message Headers

Conceptually a Message Header looks something like this sequence of Coupons:

A B C D d c b a

where **A**, **B**, **C** and **D** are coupons that route the Message to the Destination and **d**, **c**, **b** and **a** are coupons that route the response back to the Originator. A one-way datagram with no expected Response could have a header like:

A B C D

A Header may include Coupon Synonyms selected by the Originator randomly from the set of known values for each hop:

$\mathbf{A_n}$ $\mathbf{B_n}$ $\mathbf{C_n}$ $\mathbf{D_n}$ $\mathbf{d_n}$ $\mathbf{c_n}$ $\mathbf{b_n}$ $\mathbf{a_n}$

Each Coupon represents an encoded integer value. The number of bits required for each coupon will be variable and may only be determined by processing the bit stream in the correct order. It is possible to include (random) padding bits before and after the header sequence described so far. Padding on the beginning may be used to further randomize the header making it more difficult for an adversary to identify multiple messages over the same logical connection path.

pad A B C D d c b a pad

Padding on the end of the header will never be handled by a Switch, since (presumably) the response will be back at the Originator at this point. This padding may serve to obscure the actual length of the Header from an adversary. It may also contain any Message identification that the Originator might want to use to signal itself.

Markers

The Message Header is transmitted over the serial link, followed by the Message Data. The **START** and **STOP** markers are used as bracketing. The **START** and **STOP** markers are specially coded bit sequences that are recognizable immediately by the hardware because of violations of the Run-Length Limit rules for the particular physical media. The Message Header and Message Data fields are bit streams and the **START** and **STOP** markers must respect these bitwise boundaries. Each physical link will have a defined value of **L**, the run-length limit for normal operation. In these examples we use a value of **L=10**. Each physical link will use an appropriate value based on expected clock accuracy and jitter. The **START** and **STOP** markers will use **L+2** consecutive ones or zeroes as their link-escape indicator.

. . . . 0 111111111111 0 1 0 **x**	**STOP** marker after a zero
. . . . 1 000000000000 111111111111 0 1 0 **x**	**STOP** marker after a one
. . . . 0 111111111111 0 1 1 0 **x**	**START** marker after a zero
. . . . 1 000000000000 111111111111 0 1 1 0 **x**	**START** marker after a one

Note that the “long zero” pattern is used only to ensure that the end of the previous data is recognized at the correct bit.

The actual **START** and **STOP** markers are a “long one”, followed by a zero, followed by **N** ones, followed by a zero. This ensures that all defined markers end in a single zero. We have possible values of **N** in the range of **N**=1..**L**-1 where **N=1** is **STOP**, **N=2** is **START**. Additional values are reserved for future expansion.

Following any of the defined markers is the bit labeled **x** above. This bit is chosen to be the opposite polarity of the next bit of the following data. The **x** bit is discarded during decoding. It is present to ensure accurate resetting of the RLL mechanism in the Receiver. This guarantees that a Header can safely be inserted as needed in any Message Data stream.

Route Selection

The Message Header consists of a sequence of Coupons. At each hop the next Coupon is selected to choose the switch crosspoint. This requires an indicator of what is meant by the “next Coupon”.

We must preserve the Message Header so that the Destination can recognize fragmentation and reassemble a complete Message. It is not possible to simply trim off each Coupon as it is used. We also wish to ensure that only the correct, current Switch is able to recognize and act on a particular “next Coupon”. Therefore we need an unambiguous marker that can indicate the “next Coupon”. The chosen solution is to use a second **START** marker as follows:

START pad START A B C D d c b a pad STOP message **START** . . . .
START pad A START B C D d c b a pad STOP message **START** . . . .
START pad A B START C D d c b a pad STOP message **START** . . . .
START pad A B C START D d c b a pad STOP message **START** . . . .
START pad A B C D START d c b a pad STOP message **START** . . . .

The message has now arrived at the intended Destination. The remaining Coupons will be used to send the Response(s) back to the Originator.

The parts of a complete Message therefore look like this:

START prefix **START** suffix **STOP** message **START** . . . .

where prefix refers to the list of coupons that have already been used and suffix refers to the coupons controlling upcoming links.

At each hop, the Switch will identify the “next Coupon” as the one immediately following the second **START** marker, i.e. the first coupon in the suffix. This will used to configure the switch. When the output link is available the full Message Header will be sent with the second **START** marker in the next position. Thus the bit-length of each Coupon is determined only by the Switch that uses it. The prefix will grow and the suffix will shrink.

This completes Link-to-Link operation within a Switch.

If the operation of a particular Coupon requires the use of Fabric ports (switch-to-switch connection within a logical Router) , the Input Switch will insert a Fabric Coupon into the prefix after the first **START** marker. The Fabric Coupon will route the message across the required Fabric port(s) within the logical Router.

START pad A B START C D d c b a pad STOP message **START** . . . .
START FC pad A B C START D d c b a pad STOP message **START** . . . .
START pad A B C START D d c b a pad STOP message **START** . . . .

This radically simplifies and speeds up the operation of Fabric ports since the analysis of each ordinary Coupon need only be done once by the Input Link port. We have four possible situations within a Switch:

Link to Link	No Fabric Coupons involved
Link to Fabric	Create Fabric Coupon and insert it into Header
Fabric to Fabric	Preserve Fabric Coupon and forward it to next Switch
Fabric to Link	Remove Fabric Coupon

Port Polynomials and Router Polynomials

Every movement of serial data is logically modified by XORing with a pseudo-random bitstream that is unique to that operation. Unique values are used for Link-to-Link transfers between Routers, and for Port-to-Port transfers within each Router.

Every Router-to-Router Link uses a unique Polynomial and Polynomial Initializer that is negotiated during Link Setup. This is referred to as the Port Polynomial. The Output Port "encodes" the data stream and the Input port "decodes" it. Using the same Polynomial and Polynomial Initializer on each end ensures that this double-XOR recovers the correct data. This ensures that data transferred over every Link will have a pseudo-random component and will assist with data whitening for balanced data recovery. Using a different pseudo-random component for each Link will help to reduce the risk of casual adversarial snooping.

There is also a LFSR based pseudo-random sequence generator associated with every Link Input port. This is referred to as the Router Polynomial. Every ordinary Coupon has associated values for Polynomial and Polynomial Initializer. The pseudo-random sequence is applied to every Message that passes through the selected Crosspoint. Since these Polynomial and Polynomial Initializer values are associated with each Coupon they can be shared with the Originator during path setup. Knowledge of the resulting pseudo-random sequence used by each Router allows the message Originator to properly code the Coupon value into the Header.

Elastic Buffers and Rate Adaptation

Every port has an associated elastic bit buffer. The bitwise data stream received on an Input port has its run-length encoding removed. Detection of **START** and **STOP** marker patterns is performed. This will result in a Header and Message bits arriving at an essentially unpredictable rate. Analysis of the current Routing Coupon will allow determination of the intended Output port. There are three general possible situations: The nominal data rate of the Output port will be faster, equal to or slower than the Input port. The reality is rather complex since the effective data rates will be dynamic due to run-length limiting. Also there is no fixed clock rate

on the links so synchronization between Output and Input is not to be expected. The Input port will transparently adapt to the received bit rate.

Conceptually the data transfer from Input port to Output port could begin as soon as the routing Coupon is decoded. This ideal can be approached in the case of Fabric Coupons since the link properties should be well-known. For ordinary Link Coupon operations the Switch will be expected to buffer some amount of data before starting the Output operation. The minimum size before output will be an operating parameter. More buffering may be required if the desired Output Port is busy.

With the ability to create Messages of essentially unlimited length it becomes possible for a Message to occupy one or more Switch ports to the exclusion of all other traffic. To prevent this situation each Output port on a Switch maintains a count of Input ports with Messages pending for it. The number of pending Messages is used to establish a maximum allowed length for the Outgoing Message. When the Maximum length is exceeded the Output port switches, round-robin fashion, to the next pending Input source.

Every Input port maintains the ability to resend the complete Header - even after the Message Data has begun to be sent. When the Output port is needed for another Message the current Message transmission is simply suspended. When the round-robin selection returns to the suspended Message the correct Header is retransmitted, followed by the continuation of the Message Data.

The continuation of a fragmented message may start at a point zero or more bits before the last bit previously transmitted. This parameter is the Fragment Overlap value and may be used to assist the Destination in rapidly reassembling a fragmented Message.

Message Fragmentation

This port sharing process can cause any Message of sufficient length to possibly be broken into arbitrary Fragments at any Switch in the path. It is possible that persistent conflicts at congested ports will cause the operational maximum buffer size in a Switch to overflow. The buffer will continue to fill by discarding the oldest data from the bitstream. This will cause a gap in the data stream that must be recovered by additional protocol exchanges between Originator and Destination. There is no direct mechanism for signaling this type of error condition - we simply let the high-level protocols do their job.

Ensuring that the buffer handling at the Input port is unaffected by overflows allows for the next received Messages to be routed properly. Thus, there can be multiple pending Message fragments, destined for different Output ports, in a particular Input buffer at any given time.

Each Output port is tasked with democratically selecting Messages from every Input port that needs the particular Output port.

Every port will have an established parameter that sets the minimum size of a Fragment that it will create in this way. The Input and Output port buffers are coordinated to ensure that the

combination of Rate Adaptation and Port Sharing intelligently create near-optimum fragments as required.

A single message cannot block a port if other traffic is waiting. Every port will have an established parameter that sets the maximum size of a Fragment that it will send out a congested port. When breaking a Message due to port congestion, the switch will ensure that the trailing fragment will be at least the minimum length. Thus, the last fragment in a message may be slightly longer than the rest.

Once created, Message Fragments are conveyed to the Destination. Situations such as port Aliases using physical links with different data rates can cause Fragments to arrive at the Destination out of order. Out-of-order and missing-fragment recovery are accomplished at the Destination using a combination of Fragment Overlap and Phase Identification Bits within the Message. Fast and accurate reassembly requires fragments of a sufficient length.

The Practicality of Unlimited-Length Messages

We do not impose any limit on the length of the Message Data field. A Sender may transmit bits at the full capacity of a Link for as long as necessary. The Receiver can verify and process the data on the fly, without waiting for the end of the Message. There are no packet checksums or other design features that would force the Receiver to wait before processing incoming data.

Unlimited Message Length is a feature that is specifically intended for data center or High Performance Computing (HPC) implementation. These applications would benefit from eliminating packetization overhead on the communication links and the simplification of the protocol stacks on each end. Each data transfer would be able to approach the full available bandwidth of the Switch.

For wide-area networks which involve congested Routers, or paths with many different data rates, the value proposition is different. In these cases, Messages will almost always become fragmented along the path. Importantly, the fragmentation decisions will be made based on the instantaneous link condition - not as part of an arbitrary pre-set configuration. This means that the users can expect the best possible performance from the network at any given time.

Messages sent over any network may expect a certain non-zero bit-error-rate. Efforts will be made to reduce this rate, but errors must always be expected and tolerated. Long Messages will accumulate errors which may take the form of simple corruption, as well as insertion and deletion of a series of bits. Typical protocols that use packetization simply discard any packets that are not pristine, and invoke a recovery mechanism.

A design that supports unlimited-length messages cannot be so simplistic. Corruption within a long Message must be detectable and recovery must occur on the fly. The invalid portion of the Message must be identified with the finest resolution possible. Unaffected portions of the Message must be accepted and corrupted sections dealt with. The Receiver must be given the option of using the Forward Error Correction encoded within the Message to handle the situation. If the corrupted region is too large, or the FEC is otherwise unable to succeed, a

retransmission from the Sender may be requested. This retransmission may be made as specific as possible. These error handling decisions are strictly up to the Receiver, not a property of the network or intermediate nodes.

If the transfer is time-critical, the retransmission request would be made immediately. Upon receipt of a retransmission request the Sender would end the transmission of the long Message, send a Message with the retransmitted data, then resume the long Message. This sequence will look like any other fragmentation within the network and the Receiver will handle it accordingly.

If the transfer is not time-critical, the Receiver may accumulate a list of corrupted or missing areas and request retransmission in bulk. This is especially applicable to file transfers such as web page content.

As a matter of courtesy it is expected that the Receiver will periodically send a positive acknowledgement indicating successfully-received segments of a Message. This allows the Sender to release the buffers containing the compressed/encrypted/Phase-Identified content. This acknowledgement is at the discretion of the Receiver.

It is understood that Message Headers may be corrupted. In the case of a corrupted Header, that entire Message will become undeliverable and will be discarded by the network. This means that corruption of the Header of a very long Message cannot be detected or reported by the network. The Sender only recognizes the failure through the lack of acknowledgements. Correct behavior would likely be for the Sender to insert a new Header and continue the long Message. This should provoke the Receiver to send a retransmission request, which will ultimately result in the repair of the faulty/missing data.

There is no specified limit to the length of a Message Header. Message Headers cannot be fragmented and must remain contiguous. Switches that detect Headers that would monopolize a port, or cause unnecessary congestion, may discard the Header and Data.

It is recognized that there may be applications for signaling using data encoded in portions of the Header prefix or suffix that are not actually involved in network routing. This is supported, but must be implemented in a network-friendly manner.

Limitations of Reliable Communication

In general the goal of a communication network is to provide reliable, secure transfers of data between endpoints. There are instances, however, in which the timeliness of data reception is the overriding concern. Streaming Media and Voice Over IP (VoIP) are consumer-grade examples.

There are other instances that provide great value to ultra-high-speed data transfers. A fire-and-forget style of messaging can be quite effective, even if individual data does not always reach its destination. Examples include High-Performance Computing applications such as finite-element analysis for things like Computational Fluid Dynamics.

Interprocessor messaging in compute clusters should be generally reliable, but the occasional dropped message in a Peta-Flop environment will be lost in the computational noise. The penalties for guaranteed reliability, including retransmission delays or excessive error-correction overhead would slow the system to a crawl.

Big-Data statistical analysis and Machine Learning are also quite tolerant of the occasional dropped data-point. The overall accuracy of the result is unaffected and the performance benefits are overwhelming.

Phase Identification Bits

In an ideal world we could send any Message, with an unlimited length, from an Originator to Destination with only one Message Header. Reality dictates that we handle situations such as

1. The Link runs faster than the Originator can prepare the data,
2. Rate adaptation along the path overflows or underflows the elastic buffering
3. Congestion at a particular Link requires sharing with other traffic,
4. Link Aliases become available along the path to speed traffic using multiple physical Links.

All of these situations can cause a single conceptual Message to be broken into multiple Fragments at unknown and unknowable points. We require a mechanism to allow the Destination to accurately reassemble multiple Fragments into the original Message.

We cannot use any sequence numbering scheme since we do not know the nature or location of the fragmentation. We are specifically concerned with data recovery at the Destination. No intermediate "Transport Layer" solution will work.

Consider the following data stream with an inserted pseudo-random sequence of bits:

...F...F...F...F...F...F...F...F...F...F...F...F...F...

The "**.**" represent Message data bits, the "**F**"s are inserted bits from a pseudo-random sequence known to the Originator and Destination. A digital correlator can be used to identify the location within the pseudo-random sequence that corresponds to this Message section. This will allow the Destination to determine with high probability the location of a particular Message Fragment within the overall Message.

For practical reasons, the pseudo-random sequence will be of finite length and will repeat at an interval controlled by the length of the polynomial in the LFSR. Therefore we modify the approach by adding another pseudo-random sequence at a lower data rate like this:

...F...F...F...M...F...F...F...M...F...F...F...M...F...F...

The "**M**"s represent a PRN sequence applied at a lower data rate from the "**F**"s. Even though the "**M**"s obscure some of the "**F**"s our correlator will still lock to the "**F**"s with high probability. We will then be able to use a second correlator to search for the position of the "**M**"s in the sequence.

Next, we can repeat the process to add an even slower-rate sequence:

...F...F...F...M...S...F...F...M...F...S...F...M...F...F...

This gives us an overlapping sequence of Fast, Medium and Slow sequences. If the insertion rates are mutually prime we can be assured that the length of the overall repeating sequence is now on the order of N^3, where N is the length of the individual sequences. Additionally, we ensure that the cycle lengths of the three sequences are not the same: for example 1024, 1023 and 1022.

Using just three overlapping PRN sequences in this manner should allow rapid correlation of the position of Fragments within very long Messages. The location of the inserted PI bits can be rapidly determined by correlating the Fast cycle sequence. Then the PI bits can be examined to synchronize the Medium cycle and Slow cycle sequences. This yields three numbers that, when combined, result in the precise location within the bitstream.

Combining this technique with Fragment overlap bits generated by intermediate Switches will allow very rapid reassembly of fragmented Messages at the Destination.

The use of this multiple-correlation technique will also allow precise identification of missing Fragments. This precise location-and-length information allows the Destination to accurately request any necessary retransmission from the Originator.

The overhead for this capability with a $\mathbf{PI_{RATE}}$ of 131 (insert one PI bit for every 131 data bits) is less than one percent.

Using only three sequences with a PRN length of 1024 (a ten-bit LFSR) would result in a Phase Identification resolution of one bit in more than 130 Gb.

Actual implementation of the Phase Identification would almost certainly include a pseudo-random bit stream with an extremely long period, yet with an easy-to-correlate structure of multiple overlapping short sequences with unique periods and lengths. The **F**, **M** and **S** insertion described above (for clarity) is vulnerable to detection by an adversary due to the presence of repeating the **F** sequences in the clear. In an actual implementation, the value of the inserted bit would be the XOR of the three phase values at that point. We must protect the bit-insertion location, the interval, the PRN pattern and the repeat intervals from casual detection.

	Fast	Medium	Slow
Rate of Bit Changes			
Generating Polynomial			
Repeat Length			
Insertion Interval			

Over a sufficiently long sample, the correlation with any one of the phase sequences will be expected to be exactly 50% positively correlated and 50% negatively correlated. This property is

expected to be easily and rapidly recognized, especially with hardware assistance. It should also be quite difficult to identify by a casual adversary.

The inclusion of Phase Identification as part of every Message enables enhanced, high-level error recovery. A Recipient can determine the location and size of a missing or corrupt Fragment and request retransmission from the Sender. In addition, one could easily request multiple, disjoint Fragments. The retransmission in this case would consist of multiple fragments. Upon receipt it would be automatic to have the Fragments assembled in the correct places to reconstitute the original Message. All of this can be done with high confidence by ensuring that the PI phase sequence matches across all Fragment boundaries.

The Phase Identification concept and selective retransmission requests combine to create a robust random-access file transfer mechanism. In particular, it would be possible for a Recipient to be only interested in selected portions of a (very large) original Message. Only the parts of interest need be transferred over the network.

This feature can be extended to encoding of streaming media. Instead of having H.264 style interspersed data streams (video, audio, captioning, etc.) each of these can be separated at different offsets within the file. An Originator can explicitly request snippets from anywhere in the file. This allows transmission of only the actually required data and also supports accurate, near instantaneous scrubbing within the video. The Recipient uses the PI mechanism to unambiguously recognize the particular Fragments as they are received.

Since the Phase Identification bits are the final item inserted into a Message stream by the Sender, and the first removed by the Recipient, they make for a mechanism to simplify controlled subscriber access to content. Each subscriber, connection, or connection path could have its own PI parameters (or offset within the PI space) for recovering content. A given streaming video user might see PI parameters that change periodically during his viewing session. This feature would radically simplify the operations on the server side. Video streams would not need to be re-encoded or re-encrypted for every customer and session.

The Phase Identification parameters must be shared between Sender and Recipient. These include the bit-insertion rates, polynomials and PRN lengths. Without this knowledge the Message content will be unrecoverable. Although these parameters would not be difficult to discover for a skilled adversary, it does act as an additional security feature.

It is expected that the required bit-insertion, correlation, bit-extraction and PRN generation will ultimately be performed by very high speed, low power dedicated hardware.

Pseudo-Random Number Generation

Every data transfer over the network uses a pseudo-random sequence of bits to alter the data stream and make it difficult for an adversary to recognize specific messages as they traverse the network. We recommend a linear-feedback register implementation controlled by two integer values: **POLY** and $\mathbf{POLY_{INIT}}$. The length of the shift register and the location of the XOR taps are specified by POLY and a sequence of bits defined by $\mathbf{POLY_{INIT}}$ are then used in chaining mode to

establish the starting condition for the random number generation. A repeating sequence of pseudo-random bits is then generated by feeding the LFSR with zero bits as input and using the output to XOR with each Message data bit.

This approach ensures that the recipient, armed with the same **POLY** and $\mathbf{POLY_{INIT}}$ values will be able to directly decode the Message by removing the randomization.

Properly selected **POLY** values will create a shift register **N** bits long and will create a repeating pattern of 2^N bits in length.

The LFSR will be reset to the initial state at every **START** or **STOP** marker.

Every port on a Router has an initialization sequence that it performs as a handshake to establish a Link with the connected Router or Endpoint. An obvious result is that the connected ports agree on a preferred bit rate for the Link. The negotiation also establishes a **POLY** and $\mathbf{POLY_{INIT}}$ for each direction. These are used to "encode" and "decode" traffic in each direction over the physical Link. The actual Message content is not affected by this - it is purely for making data streams unique among different Router ports and Links. Think of this as a Data Whitening operation that is unique to each physical Link, and that changes each time the Link is reset.

Every Routing Coupon has an associated **POLY** and $\mathbf{POLY_{INIT}}$ that is used to modify the Message Header as it passes through the Router. These values are created by the Routing Processor (RP) and used to control the Switch setup tables that allow the Switch to recognize the Coupon. When an Originator requests a route to a particular Destination from the Routing Processor, the values of a (set of) Routing Coupons are returned. With each Coupon value are also **POLY** and $\mathbf{POLY_{INIT}}$ values. These are retained by the Originator and used as part of the Message Header Origination process.

Unlike the Link data whitening which is added and removed at each end of a single link, this randomization is fully applied by the Originator and removed, one step at a time, by each Router in the path. This ensures that an obfuscated Message Header changes at every hop and is only revealed at the single Switch that acts on the Coupon.

Response Messages are created by a Destination element and follow the sequence of Return Coupons, created by the Originator, back along a path to the Originator. Thus the Originator has full control of the path followed by all traffic that it sends or receives during a session. And no Switch or Router along the way knows anything at all about the path, apart from for its immediate physical connections.

Coupon Creation

During creation of a Message Header the Originator must obtain the necessary Coupons. Coupons are created exclusively by Routing Processors in each Router along the path.

The Originator makes a request to each Routing Processor in turn. The Routing Processor looks up the requested entry for the next Router, determines the correct crosspoint number to connect the requester's incoming port to the desired target output port. Then the RP finds the (set of) Routing Coupons that are currently loaded into its Switch that match the desired crosspoint. One or more of these coupons are returned to the Originator as candidates to use for its Message Header.

A Coupon is a series of bits that will be recognized by the Switch. We make no other assertions about their length or content. The 5-bit RLL representation of integers described here is one possibility. Each Router in a network may use different representations without compromising interoperability.

Every Router input port has an associated pseudo-random sequence defined by a **POLY** and **$POLY_{INIT}$** . These values are returned to the Originator along with the candidate Coupons. The computed PRN sequence must be used in the construction of the Message Header and is used by the Switch to decode received Headers.

In addition to the Coupon(s), **POLY** and **$POLY_{INIT}$** to connect to the requested Router, the response will also include the same set of information for the reverse path. This is used by the Originator to build the continuation of the Message Header that is used by the Destination to route a response back to the Originator.

Route Discovery

When an Originator wishes to connect to a new Destination it must discover the required route. The target Name or unique ID is used in a request to a Routing Processor. The RP looks for Names on a best-match basis; IDs matches must be exact.

All Routing Processors act as (at least minimal) Name Servers. They must be able to recognize each of the Routers or Endpoints connected directly to each of their ports and respond appropriately. In addition they will be able to direct an Originator to more comprehensive Name Server(s). Each Router may make decisions as to how complete it wants its own directory to be. This creates the desired peer-to-peer Name Server architecture.

Name Servers

Any Router or Endpoint may request to be listed in any Router's Name Server. Name Server entries may be more comprehensive than current DNS entries, and are more like a combination of DNS, routing tables, keychain and WHOIS entries. In order to be listed, the entry must contain a verifiable path from the Name Server to the listed entity. The Name Server verifies the path and the credentials of the target entity before allowing the listing.

The reported path consists of a list of Router IDs - not Coupons. This ensures the path from Name Server to target entity will remain (more) stable over time and can be used from anywhere on the network.

The identity of the Originator, the Target being requested and the subsequent use of that path are all independent operations. The network will obscure Originator's request to the Name Server.

The Originator will make subsequent requests to each of the Routers in the path to obtain the necessary Coupons for each hop. This ensures that the Originator knows the published IDs for each hop, and that the Router that generates requested Coupons can be authenticated.

Path Trimming

After obtaining a path to a Destination, the Originator will have a list of the intermediate Routers along that path. Communication may commence as soon as the required Coupons have been obtained.

As an autonomous, independent process, the Originator may continue interrogating Name Servers along the path. Requesting routing information for each of the intermediate routers along a path may allow discovery of shortcuts or alternative paths.

The use of alternative paths for individual Messages is completely at the discretion of the Originator. Concurrent use of alternate paths may be used to evaluate path performance. Path diversity allows increased security and fault tolerance.

Firewalls and Private Networks

Requests for routing information and Coupons may be made to any Router. The choice of whether or not to respond, and what information to return, is at the discretion of the Routing Processor. Normally a Routing Processor will return a standardized set of responses matching the request.

Additional rules relating to functionality for Firewalls and Private Networks may alter the response. For example, a particular Router may require authentication on requests coming from particular ports. This could come in the form of signed or encrypted requests.

There is no requirement that a private Router provide the same response to each requester. E.g., the router may have multiple identities and reveal none, one or a subset to a particular requester. Variations on this theme allow secure remote administration of a Routing Processor. Features that are normally not allowed (such as enumerating all connected routers and ports) might be necessary for network administration and might simply be restricted to connections with the correct credentials.

The simplest implementation would be to simply not issue routing Coupons to requests that are not authorized.

Private networks can be implemented over shared Routers by restricting the issuance of Coupons relating to particular ports to authorized users. Private Routers can be safely

connected to any port of a public network. Improperly authorized requests for Coupons will simply be ignored. No traffic can pass through the Router without presenting a valid Coupon.

Glossary

Alias: Mechanism to allow a particular Routing Coupon value to select from a set of effectively identical connections between Switches or Routers. Used to allow added bandwidth via ports with parallel cabling.

Balanced Data: A bit stream with evenly distributed zeroes and ones. Helps the data discriminator to accurately resolve the bits in the presence of low signal to noise ratios.

Bridging Router: A Router that acts as both a legacy Router and a set of virtual Endpoints. Allows interoperability with legacy networks, but with the same security vulnerabilities as a VPN access point.

Connectivity Request: Message used to ask a Routing Processor for the necessary information to route a Message to a particular Router or Endpoint.

Crossbar Switch: A switch consisting of M inputs and N outputs that allows connections to transfer data at any of the MxN crosspoints. Used here we are talking about a symmetric Switch with **P** full-duplex ports and $\mathbf{P}^2$ crosspoints.

Crosspoint: The connection within a crossbar switch that connects any Input port to any selected Output port. Each crosspoint within a Router is assigned a unique number, which is part of what is implied by Routing Coupons in the Message Header.

Data Whitening: Use of a pseudo-random bit sequence to help ensure balanced data on a physical link. The output of a linear-feedback shift register is XORed with each successive bit of the data stream. The LFSR is reinitialized with each **START** or **STOP** marker.

Destination: The final target node for a Message from an Originator. Will usually create a response Message and send it using a continuation of the same Message Header (which will be the Return Routing Coupons) to ultimately reach the Originator.

Endpoint: Either the Originator or Destination of a Message. May be the Routing Processor inside a Router.

Fabric Port: Bidirectional physical serial port that connects between Switches within a Router.

Fragment: Pieces of a conceptual Message, each with a copy of the Header, either created by the Originator or dynamically by intermediate Switches along the Path.

Identification Request: Message used to discover the unique identification of a Routing Processor (i.e. Router) or an Endpoint.

Line Port: Bidirectional physical serial port that connects Router to Router or Router to Endpoint

Linear Feedback Shift Register (LFSR): Simple hardware implementation that can be used to generate a pseudo-random sequence of bits. The conceptual length and feedback polynomial can be expressed as a small integer.

Message: Sequence of zero or more bits sent from an Originator to a Destination. Preceded by a Message Header that describes the route to the Destination.

Message Header: Sequence of Routing Coupons that fully describes the sequence of Routers to use to send a Message from the Originator to the Destination and back to the Originator.

Node: Element of a connected network consisting of either an Endpoint or Router. All nodes can respond to Identification and Connectivity Requests (if security allows).

Originator: The node that creates a Request Message and prepares the Message Header.

Overlap: The number (zero or more) of bits that are duplicated in a Message when dynamic fragmentation occurs within a Switch. Used by the Destination to assist in assembling received Fragments in the correct order.

Path: Sequence of routing hops that Message traverses from Originator to Destination and back.

Phase Identification Bits: A sequence of pseudo-random bits inserted at specified intervals into a Data Message. PRN sequences inserted at different, relatively prime, intervals allow recognition of unique locations within extremely long bit streams. This is a variation of bit sequence correlation similar to that used in gene-sequencing.

Polynomial: Numeric value used to configure a Linear Feedback Shift Register (LFSR). Establishes a pseudo-random sequence of bits that may be unique to data transfer within a Router or between Routers. Correctly chosen Polynomial values will create a LFSR containing **M** bits with a repeating period of 2^M unique values.

Polynomial Initializer: Numeric value fed into a Linear Feedback Shift Register prior to transmitted or received data. Used to effectively set the starting offset within the pseudo-random bit stream.

Prefix: The sequence of Routing Coupons in a Message Header that have already been processed. The bits between the first and second **START** markers.

Pseudo-Messages: Randomly generated Message Header and Message Data transmitted over idle Links to defeat traffic analysis by an adversary.

Rate Adaptation: The ability of data received by a Switch Input port at one speed to be transmitted to an Output port at a faster or slower rate. Uses elastic buffering for optimum efficiency.

Request Message: Message sent from the Originator to the Destination.

Response Message: Message sent from the Destination back to the Originator.

Routing Processor: Computing element within a Router that responds to Identification and Connectivity Request Messages. Performs setup and initialization of one or more connected Switches.

Router: Combination of a Routing Processor (RP) and one or more Switches.

Routing Coupon - A sequence of bits that select a router switch connection (crosspoint) within an overall path. Multiple coupons are concatenated to form a Message Header. The Header describes the forward and return path for routing the request and response.

Run-Length Limit: Preventing sequences of more than **L** consecutive ones or zeroes in a data stream. During transmission when **L** identical bits are detected a bit of the opposite polarity is artificially inserted. During reception the pattern of **L** identical bits followed by the opposite causes the opposite bit to be discarded. Violations of this rule are used to encode the **START** and **STOP** markers required by the protocol.

Sequential Relative Routing (SRR) - A message header that contains a list of port-to-port connection numbers describing how to configure each consecutive switch in a network.

Store and Forward: The basis of current IP routers. Requires every packet to be received in its entirety at each router before it is analyzed and transmitted out the desired port. What we are specifically trying to avoid.

Suffix: The sequence of Routing Coupons yet to be processed. The bits between the second **START** and the **STOP** marker in the Message Header.

Switch: Hardware device with multiple bidirectional physical ports. Capable of transferring binary data messages from any input port to any output port under control of the Routing Coupons in the Message Headers.

Synonym: Multiple Routing Coupon values that resolve to the same connection within a Switch. Used to allow randomization within a Message Header for security purposes.

Appendix: Operating Parameters

The operation of the proposed message routing system can be fully characterized using the set of parameters described below. Origination and transfer of messages can be accomplished in a fully interoperable manner.

These represent suggested values. In most cases negotiations may adjust the values during initialization of links, Switches, Routers, Endpoints and Connections.

Network Parameters

Parameter	Value	Description
RLL_{INT}	5	Run-length Limit for integer encoding. Used in Coupons and Polynomials
RLL_{MSG}	10	Run-length Limit for message encoding. Used to create markers for Header and Message Data
$MARKER_{START}$	**1**	Bit sequence in Marker that identifies it a **START** symbol
$MARKER_{STOP}$	**11**	Bit sequence in Marker that identifies it a **STOP** symbol
POLY		Polynomial configuration for a Linear-Feedback Shift Register
$POLY_{INIT}$		Bit sequence fed through a LFSR prior to using it as a PRN sequence generator
PI_{RATE}	131	Interval between inserted Phase Identification bits. Probably should be a prime number.
PI_{POLY}		Polynomial for generation of pseudo-random sequence used for Phase Identification
$FRAG_{MIN}$	10,000	Minimum number of bits in an auto-generated Message Fragment. Helps to ensure sufficient Phase Identification bits for successful Message reassembly at the Destination.
$FRAG_{MAX}$	100,000	Maximum number of bits in a Message Fragment created when there are other pending Messages for a specific Output port. Helps to ensure equitable sharing of congested ports.
$FRAG_{OVERLAP}$	12	Number of overlapping (redundant) Message bits sent in an auto-generated Message Fragment

Shared Parameters

The following parameters are shared between an Originator and Destination. Usually this occurs once during the setup of a logical connection. Knowledge of these parameters allows secure communication between the two endpoints.

Parameter	Description
Compression Algorithm	Algorithm used to perform bulk compression of the Plain-Text data message
Encryption Algorithm	Encryption algorithm used to secure the compressed data
Encryption Keys	Cryptographic keys necessary to encode and decode Messages
Phase Identification Parameters	Values used to control the insertion and detection of Phase Identification bits into the encrypted Message

Bibliography

The following documents provide background and additional information on the topics discussed here.

RFC 791 Internet Protocol -
https://www.rfc-editor.org/rfc/rfc791.txt

RFC 8200 Internet Protocol, Version 6 (IPv6) Specification -
https://www.rfc-editor.org/rfc/rfc8200.txt

RFC 792 Internet Control Message Protocol -
https://www.rfc-editor.org/rfc/rfc792.txt

RFC 793 Transmission Control Protocol -
https://www.rfc-editor.org/rfc/rfc793.txt

RFC 1034 Domain Names - Concepts and Facilities -
https://www.rfc-editor.org/rfc/rfc1034.txt

RFC 1035 Domain Names - Implementation and Specification -
https://www.rfc-editor.org/rfc/rfc1034.txt

RFC 1060 Assigned Numbers -
http://www.rfc-editor.org/rfc/rfc1060.txt

RFC 2131 Dynamic Host Configuration Protocol -
https://www.rfc-editor.org/rfc/rfc2131.txt

RFC 1771 A Border Gateway Protocol 4 (BGP-4) -
https://www.rfc-editor.org/rfc/rfc1771.txt

RFC 5246 The Transport Layer Security (TLS) Protocol Version 1.2 -
https://www.rfc-editor.org/rfc/rfc5246.txt

RFC 6101 The Secure Sockets Layer (SSL) Protocol Version 3.0 -
https://www.rfc-editor.org/rfc/rfc6101.txt

ISO/IEC 13239-2002 High-Level Data Link Control (HDLC) Procedures -
https://www.iso.org/obp/ui/#iso:std:iso-iec:13239:ed-3:v1:en
This is behind a paywall.

High-bandwidth Digital Content Protection System Revision 1.4 -
https://www.digital-cp.com/sites/default/files/specifications/HDCP%20Specification%20Rev1_4_Secure.pdf

Bellcore GR-246-CORE Specification of Signalling System Number 7 -
http://www.ece.virginia.edu/~mv/standards/telcordia/gr246.pdf

Open Systems Interconnection Model (OSI Model) -
https://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip
This is a copyrighted PDF inside a ZIP file and requires accepting a license agreement to download.

EU General Data Protection Regulation (GDPR) -
https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN

The Onion Router (TOR) Specification -
https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt

Internet Corporation for Assigned Names and Numbers (ICANN) -
https://www.icann.org

Internet Assigned Numbers Authority -
https://www.iana.org

North American Numbering Plan Administration -
https://www.nationalnanpa.com

Contact the Author

Richard Feynman said that ***precision is the enemy of clarity***.

My goal for this document is to clearly describe limitations of current message protocols and to propose meaningful alternatives. To that end, I have left many "implementation details" to be filled in by the discerning reader. I welcome comments, especially those that indicate where I might have given a misleading impression, overlooked an important aspect, or made a statement that is incorrect. Suggestions to improve the clarity for all readers are appreciated.

Brian McMillin brian@bkmcm.com