
CHAPTER 2

The Mirage model

Mirage is an abstract model that describes latency in communication [To89], [To90a], [To91a]. The model consists of three components: a model of node state, a model of time and communication as state transformations, and a set of constraints on these transformations.

The model is designed to serve as a viewpoint for understanding existing protocols and for the development of new protocols. It is based on providing an abstract model against which protocol instances can be measured and compared. Prior work in protocol design and analysis is based on an alchemy of examining particular protocols; this work is designed to provide an initial framework for treating protocol research as a science.

Current research in gigabit wide area network protocols suggests a “clean sheet” approach is indicated. The transition from evolutionary design to needed new (revolutionary) protocols will provide a fresh opportunity for introspection into the mechanism of protocol models. Before suggesting new protocol designs, we consider how the problem of communication has changed, and develop a new model that incorporates these changes from the start. Mirage is an attempt to characterize the tradeoffs that emerge in the new domain of gigabit WANs.

2.1. Assumptions

As communication rate increases, fixed latency increases the amount of information in transit between interacting nodes. Increases in information separation alter the assumptions of the limiting factor in communication. We have already presented, in Chapter 1, our definition of a protocol and communication. We also assume a domain where latency is large, relative to the bandwidth of communication.

2.1.1. Initial description of channel utilization

Initially, Mirage can be described in terms of channel utilization. This describes the indeterminism of the model and provides a real performance measure that Mirage is intended to enhance.

Information in the channel can be modeled as a stream of data. In a transaction protocol, this stream consists of a single message, whereas, in a file transfer protocol, it consists of a linear sequence of characters. A linear stream can be extended to accommodate alternate possible streams; the stream of composite linear components is a *branching stream*.

Branching streams are common in interactive systems, where initial data messages indicate subsequent choices to be made by the receiver, which guides subsequent messages. The stream can be represented by a tree structure, where the *limbs* represent linear streams of data (similar to file transfer), and the *bifurcations* indicate choices among alternate streams (to be made by the receiver).

This work assumes that the latency is large compared to the bandwidth, such that the bandwidth delay product is large relative to the local storage available at nodes in the network. For current networks, this means a bandwidth-delay product in the order of tens of megabytes.

A minimum delay between participants in the protocol is also assumed, which in most cases is the propagation delay. For most equations, a fixed delay is assumed, although we show later (Chapter 4) how a variable delay can be accommodated.

To some extent, messages are assumed not to be lost, or, if they are, retransmission is required as in any existing protocol. We assume, as concluded earlier, that the limitations of existing protocols in this domain will be their inability to predict sufficiently enough information to occupy the round trip latency. Finally, Mirage shows

how these limitations can be circumnavigated, given enough information about the branching stream of data. This has important implications on the limitation of layering in protocols and abstraction in the layers of a protocol.

2.1.1.1. Phases

There have been three phases of network protocol design (Figure 2.1). In the first phase, characterized by the Network Control Protocol (NCP) [Ca70] (used in the ARPAnet at 56 Kbps), the information sent is treated as an unstructured block of data. This method sufficed where the partitioning of the data served no purpose; because NCP assumes lossless, ordered transmission, this was a reasonable design. NCP is a sufficient model for request/response protocols.

Sliding window protocols, e.g., TCP (used in the Internet at rates of 56 Kbps-45 Mbps), extended this view by partitioning the message stream into multiple blocks. The partitioning can be thought of as a *lookahead* into a time-ordered list of NCP block entities – individual bits in a block are assumed to be lossless and ordered, but the block unit may be lost or resequenced as a whole. In NCP only one block is sent at a time, whereas TCP looks multiple blocks ahead into the time-ordered list of NCP-like blocks. This *linear lookahead* permits TCP to accommodate latency (and introduces resequencing problems due to multiple outstanding packets), provided the delay associated with either is limited to that which can be accommodated by the lookahead.

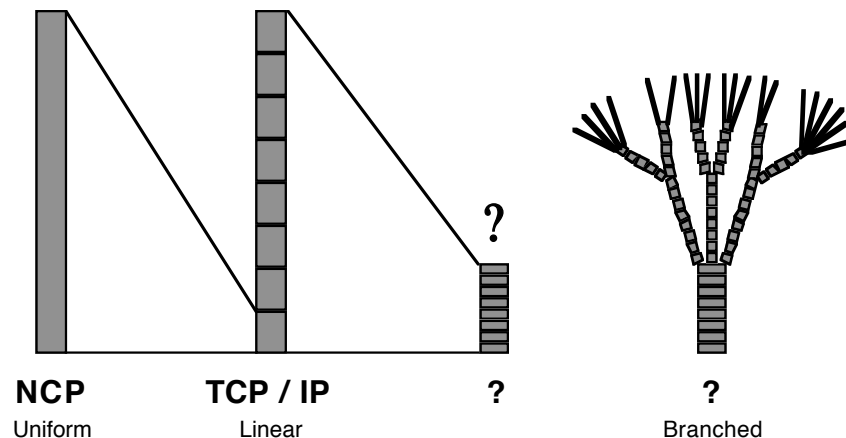


FIGURE 2.1

Kinds of protocol lookahead / branching

In the case of high bandwidth-delay product, linear lookahead will be insufficient to ‘fill the pipe’ with information, and thus utilize the channel efficiently. As the network bit rate increases, *there is a point at which the data required by the receiving end cannot deterministically be predicted and thus a TCP-like protocol will exhibit performance failure*. The time-ordered list of data blocks is limited to the amount of data that can be deterministically predicted to be required at the receiving end. This is where Mirage helps, by modeling the portion beyond the linear lookahead as branching (the tree in Figure 2.1). Mirage suggests transmission not only what is *known* to be needed, but also what *might be* needed.

2.1.1.2. Channel utilization: linear case

We define the efficiency of a protocol to be the ratio of communicated information to the channel bandwidth (percent utilization of the channel). Protocols are maximally efficient when the node buffer size¹ is less than the bandwidth-delay product (Equation 2.1). The formula is linear in the number of buffers, but inverse in the round trip time, so increases in latency may have severe effects on channel efficiency [Ta88]. There is a point at which the linear lookahead fails (i.e., cannot further anticipate the data stream required by the receiver), and utilization diminishes (Figure 2.2).

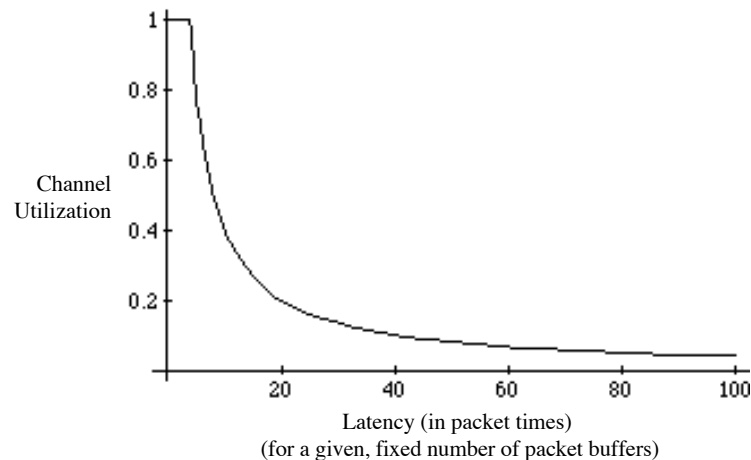


FIGURE 2.2
Utilization as latency increases (linear lookahead)

¹Node buffer size is defined as the total size of the state space of the communicating entity.

$$\text{Equation 2.1: } \%util = \begin{cases} \frac{B}{R} & \text{where } R > B \\ 1 & \text{where } R \leq B \end{cases}$$

where B = number of buffers in sliding window protocol

$R = \text{bandwidth} * \text{round_trip_time} =$ buffers used in one round trip

2.1.1.3. Channel utilization: sender-based anticipation

We will now demonstrate how a branching stream can increase the utilization of a channel in the presence of latency. Assuming that such branching is characteristic of the data stream being communicated, Mirage suggests a protocol that is capable of supporting labeling in the data stream. We call this labeling *guarded messages*, which will be described in greater detail later in this chapter.

Guarded messages are labels on communication stream components, so that the sender can indicate the desired state of the receiver for a given message to be accepted. A set of messages with suitable guards can permit branching in communication stream. Branching of the stream permits the sender to ‘run ahead’ of a known (single) receiver state, into a set of receiver states, accommodating latency in the resolution of that state.

2.1.1.3.1. An example for illustration- the turtle

Consider a turtle moving on a two-dimensional map. The turtle moves at 5 spaces per minute. A message that directs the turtle takes 1 second to send (bandwidth), and the messages take 1 second for delivery (latency). In this case, traditional request/response communication suffices to direct the turtle and confine it to within 1 space.

If the messages are delivered with higher latency (e.g., 50 seconds, ~1 minute), or take less time to send (20 milliseconds), the turtle cannot be confined so precisely. The latency implies an error of at least 5 spaces in the turtle’s location.

Guarded messages permit use of the ‘possible messages in transit’ before knowledge of the receiver’s state is known. At most 50 messages can be sent in the latency. We send 1 message labeled with each of 50 points around the last known turtle location. Each message redirects the turtle to the desired goal of the communication, given each possible current turtle location.

Note that in this case, guarded messages accommodate the imprecision induced by latency, since an error of spaces of movement indicates an area of 50 spaces in a two-

dimensional grid¹. Unfortunately, guards consume communication bandwidth, so that less than 50 messages are possible in the given latency. Further constraint of the turtle's movement would be required to guarantee that a message would be received at each possible location. These constraints necessitate grouping locations (coarse partitioning of the turtle's location space), or restricting the turtle's movement so that less than 50 spaces are possible. These will be discussed later in this chapter as well.

2.1.1.3.2. Branching streams utilization

In branching streams, only part of the anticipated stream is actually used at the receiver. The revised channel utilization formula must account for the differences between sent data (multiple streams) and utilized data (a single stream). For channel utilization to increase, two forms of prediction are necessary – *predict enough* data to send, and *ensure the utilization of enough of the predicted data*.

If communication is limited to the linear portion of the predicted data, channel utilization will decrease as the latency increases (as before, in Figure 2.2). As latency increases, there is a point at which the amount of linear lookahead is insufficient to occupy the channel during the round trip time. In current protocols, as bandwidth increases there is often insufficient buffer space to accommodate further linear lookahead. As buffer space increases (i.e., as memory becomes cheaper and larger buffers can be accommodated), this will cease to be a primary issue; the linear portion will, at some point, no longer suffice to fill the buffers. The problem is a failure to predict which data stream to send, after an initial period of success. *The lack of buffer space is a short term issue; in the longer term, we expect the indeterminism of the data stream to dominate the problem.*

A simple model of the indeterminism of the data stream following some linear prefix assumes that the branching stream has finite *branch degree* (branching factor) and finite linearity before branching recurs (*limb length*, to extend the 'tree' analogy). Members of a branch are assumed to be **isopotent**, which we define to mean "information redundant." Redundancy usually refers to duplicate data used to protect against corruption. Isopotency indicates that members of a set affect the node similarly, and that only one member of the set has any effect.

¹If the turtle can move 5 spaces, then it can move 3 spaces left and 2 spaces up on the grid, for example. The total area is 10 by 10 on the diagonal, or 50 possible spaces.

For an example of isotopotency, consider the set of messages that directs a turtle to the center of a grid without its knowing that goal (this is a relative of the earlier turtle of Section 2.1.1.3.1.). Assume that the turtle can determine its own grid position, it goes 5 grid units in a unit time, and messages are delayed by 2 units of time. The turtle enters the grid from the north-west, at some distance.

The message “go south-east” can be sent for some time, but only until the turtle is within 10 units of the center. Any fixed messages sent can be incorrect in their assumption of the turtle’s current position. Instead, sets of messages are sent, guarded (in Dijkstra’s sense of guarded commands [Di76]) for each quadrant of the area around the center. These messages constrain the turtle to smaller and smaller areas, where it finally rests on the center. The messages are distinct, but the set of messages together has the same net effect, of directing the turtle towards the center of the grid. This is isotopotency.

Stated another way, the remote node is in some known set of states. The guards partition this set, and a message affects only the states within its indicated partition. After the set of messages has been received, the remote node is in some other set of states (the union of the original partitions, transformed by the messages), regardless of the original state. The guards all have the same effect (i.e., are isotopotent) – ensuring subsequent membership in this set, even though the messages are distinct.

Channel utilization is defined here as the percent usable messages per unit time, where all messages are of uniform length, time units are normalized at one message per unit time, both *branch degree* (D) and *limb length* (L) are fixed and finite, and that the branch alternates are equiprobable. The model for the exact formula of channel utilization accommodating branching is described here, under these assumptions.

The branching stream forms a tree, where the trunk represents the linear lookahead, the branch degree is the tree degree, and the limb length is the distance in messages between levels (Figure 2.3). The set of messages used by the receiver are the sum of the linear lookahead (all of which are used by the receiver), the number of full tree levels (because one path through these levels must be useful), and the probability of utilizing the number of leaves at the last, partially unfilled level (Equation 2.2).

$$\text{Equation 2.2: } \%util = \frac{P + L * full_tree_depth + extra_leaves * prob_leaves}{rtt}$$

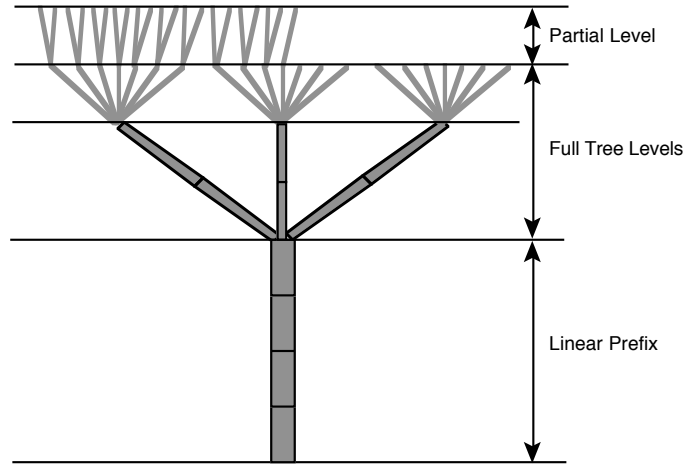


FIGURE 2.3
Tree levels

In a given round trip time (rtt), the sent messages include the linear prefix (P), some number of messages corresponding to the filled tree levels ($full_tree_depth$), and some remainder of leaf messages at the last level ($extra_leaves$) (Eq. 2.3). The number of messages in the filled tree levels requires knowledge of the branch degree of the tree (D) and the limb length (L). Using this and the identity of a summation of a power (Eq. 2.4), Eq. 2.3 can be rewritten as Eq. 2.5. Eq. 2.5 can be solved for $full_tree_depth$ (Eq. 2.6).

$$\text{Equation 2.3: } number_sent = rtt = P + \sum_{i=1}^{full_tree_depth} D^i * L + extra_leaves$$

where P = linear lookahead length

L = limb length (number of messages between branchings)

D = branch degree (used in later equations for $full_tree_depth$, etc.)

rtt = round trip time

$$\text{Equation 2.4: } \sum_{i=1}^n x^i = \frac{x * (x^n - 1)}{x - 1}$$

$$\text{Equation 2.5: } number_sent = P + \frac{D * (D^{full_tree_depth} - 1)}{D - 1} * L + extra_leaves$$

$$\text{Equation 2.6: } full_tree_depth = \lceil tree_depth \rceil = \left\lceil \log_D \left(\frac{(rtt - P) * (D - 1)}{L * D} + 1 \right) \right\rceil$$

$$\text{where } tree_depth = \log_D \left(\frac{(rtt - P) * (D - 1)}{L * D} + 1 \right)$$

Similarly, the number of leaves remaining in the partially filled level can be represented (Eq. 2.7), and using the identities in Eqs. 2.8, 2.9 and the notation of ‘fractional part’ in Eq. 2.10) a more simplified overall utilization formula is derived (Eq. 2.11).

$$\text{Equation 2.7: } extra_leaves * prob_leaves = \frac{(rtt - P) - \frac{D * (D^{full_tree_depth} - 1)}{D - 1} * L}{D^{full_tree_depth} + 1}$$

$$\text{Equation 2.8: } rtt - L = \frac{D * (D^{full_tree_depth} * D^{F[tree_depth]} - 1) * L}{D - 1}$$

$$\text{Equation 2.9: } extra_leaves * prob_leaves = \frac{D^{F[tree_depth]} - 1}{D - 1} * D$$

$$\text{Equation 2.10: } F[x] = x - \lfloor x \rfloor$$

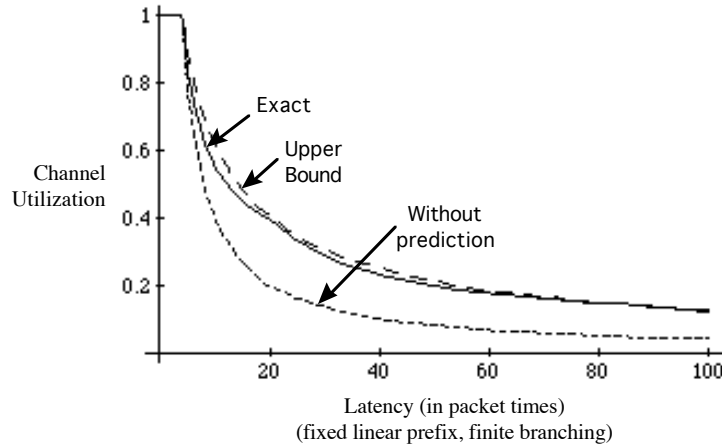


FIGURE 2.4
Utilization of branching lookahead (P=5, D=2, L=4)

$$\text{Equation 2.11: } \%util = \frac{L + L * \lfloor tree_depth \rfloor + \frac{D^{\lfloor tree_depth \rfloor} - 1}{D - 1} * L}{rtt}$$

$$\text{Equation 2.12: } \%util = \frac{P + tree_depth * L}{rtt}$$

Equation 2.11 is a discontinuous curve (Figure 2.4, Exact). The upper bound of this curve uses *tree-level* in its original (continuous) form, rather than its discontinuous floor function (Eq. 2.12, Figure 2.4, Upper Bound) (see also Appendix C). Comparing the linear stream curve to the branching stream exact curve and branching stream upper bound, a utilization increase is shown, due to anticipation of receiver requests which branching stream alternates permit. The utilization of the channel, relative to using linear lookahead only, increases without bound (logarithmically) (Figure 2.5). The decrease in channel utilization is due to indeterminism in the data; the channel is full of data, but not all the data sent is actually useful. For a given limb length and branch degree, this is an upper bound on channel utilization, given limited prediction capability (i.e., limited to the linear lookahead).

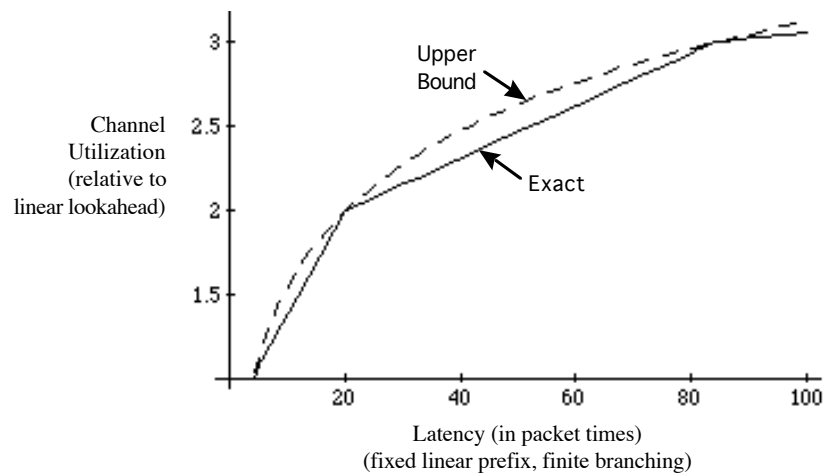


FIGURE 2.5
Channel utilization (relative to linear) ($P=5, D=2, L=4$)

2.1.2. Conclusions of the branching model

The previous discussion describes how a branching stream model of communication can permit increased utilization of the channel in the presence of latency. Branching streams presented thus far are a simple model in which a state space evolves according to static, context independent rules. A more complete model includes context sensitivity, so that the branching and limb lengths are irregular; this model is provided by Mirage in Section 2.2. of this chapter.

In the case where branching is regular and context independent, the channel utilization increases logarithmically with an increase in its bit-latency. These results are verified in a real example in Chapter 5, in which this model is applied to processor-memory interaction as a communication protocol. Measurements indicating the real values of the branching and limb lengths are presented in Chapter 6, in the analysis of the feasibility of the implementation of the designs suggested by this application of Mirage's principles.

Before proceeding, some tests can be done to verify the reality of the branching stream model. These include tests of the limiting cases, i.e., considering the boundary conditions of the equations.

2.1.3. Some reality checks

Given the above model for a channel with imprecision, consider the characteristics of these formulae as various parameters are taken to their limits. These limiting characterizations should match their real-world counterparts.

As limb lengths increase, the indeterminism of the receiver is postponed at each decision. There is less indeterminism in the stream (Figure 2.6), and thus communication holds to the limb paths longer. For example, on the first branch, as the arm length increases, the utilization approaches $1/\text{branch-degree}$, because utilization is dominated by the first choice in the branching stream. In the case where branching is binary, the limit of the utilization as the branch lengths approach infinity is 50%.

A linear stream exhibits determinism during the linearity, and complete indeterminism thereafter; an infinite limb length stream has a linear portion followed by a single branch point. The communication after the branch point dominates the channel

utilization, and the branching never recurs, both due to the extreme length of the limb. In other words, a linear stream exhibits no choices (complete determinism), whereas a branching stream with an infinite limb length exhibits one choice among *branch-degree* alternates.

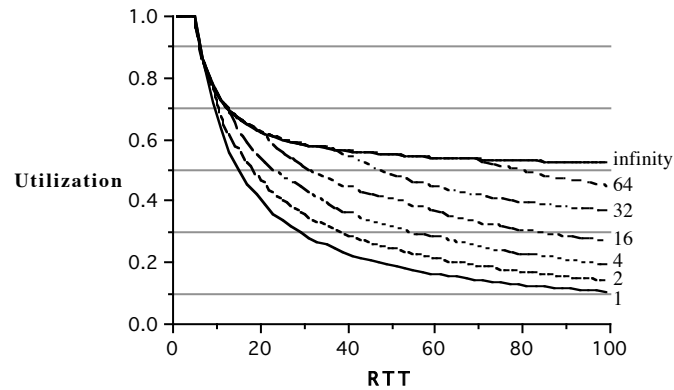


FIGURE 2.6

Utilization vs. branch arm length (geometric)

As the branch degree increases, utilization approaches the linear stream case (Figure 2.7). Sender anticipation is increased when behavior of the receiver is predictable, i.e., when the branch degree is minimal. A larger branch degree represents more indeterminism in the branching stream. As the branch degree approaches infinity, the linear stream case results, because a linear stream consists of a fixed linear lookahead followed by infinite indeterminism, i.e., no prediction of data subsequent to the known lookahead.

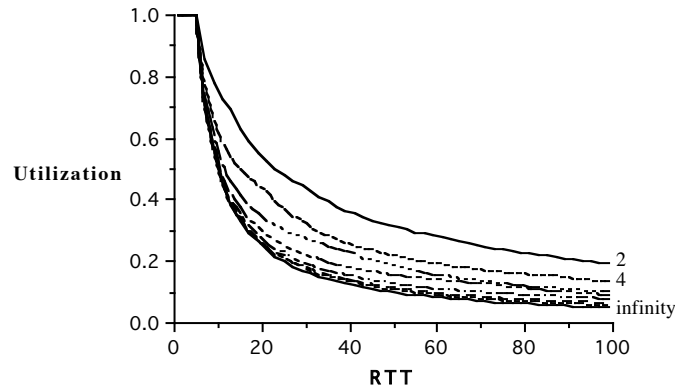


FIGURE 2.7

Utilization vs. branch degree (geometric)

A conventional TCP-like stream is the result of a limb length or branch degree of zero, (i.e., no tree beyond the linear lookahead), where the utilization is 1 whenever the buffers can accommodate the round trip latency via linear lookahead.

These examinations are the effect of varying limb length and branch degree, and consider the consequences in the resulting channel utilization. Limb length and branch degree are characteristics of the communication stream, determined by the nature of the communication and the protocol facilitating that communication. They represent the indeterminism in the communication, and are modeled by discrete finite indeterminism increases (branching degree) recurring at known intervals (limb length).

2.1.4. Some common sense

Channel utilization has been discussed, along with a method that accommodates data stream imprecision (i.e., multiple possible data streams) allowing a higher channel utilization than conventional linear streams. We believe that existing protocols will exhibit performance failures in gigabit WANs because the linear lookahead will be insufficient to occupy the channel during the round trip time.

Before discussing the particulars of the Mirage model, and how these bifurcating streams are accommodated within it, some common sense rules of protocols should be mentioned. These are commonly known constraints, but which are rarely included in protocol models.

A remote node with a highly fluctuating state requires a higher bandwidth or lower bit-latency to communicate effectively, because its requests are more unpredictable. Making pre-existing rules that restrict the fluctuation is the only way to overcome this limitation. We live with these rules daily. A human parent requires a high bandwidth and low bit-latency channel to his infant child, because there are very few assumptions that the parent can make about the safety of the child. Low latency is provided by proximity to the child. The child represents the highly fluctuating remote node, with respect to the parent.

As the child is moved away from the parent, either the constraints increase (via the addition of a sitter to the infant) or the bandwidth increases (e.g., via a nursery monitor). A nearby infant is not attended to as intensely as a monitor, because the increase in latency (time before parental intervention is possible) necessitates an increased anticipation of imprecision. A nearby crying baby is often ignored in the short term,

because the low latency (proximity to the parent and low time until the parent can attend the child) permits the parent to postpone action until absolutely required. A crying baby in the next room causes the parent to act on each cry, in anticipation of a more serious event, which would require more time to act upon.

The same parent may talk to his child once a week when the child is in college, because by that time there is sufficient knowledge of constraints in the child. Known fluctuation constraints permit a relaxation of the latency limitations¹.

A gigabit WAN has too much information in transit to manage. Additional constraints are required which describe how the stream branches (bifurcates) and which denote redundancy in the branches, in order to permit effective communication at high bandwidths. The system needs to be sufficiently predictable to utilize the bandwidth, but not so predictable that communication is obviated. The Mirage Model provides measures for being “*predictable enough*”.

2.2. The Mirage model

The abstract Mirage model is based on representing remote nodes as volumes in state space where data transmission and reception, as well as time evolution, are modeled as transformations on those subspace volumes. State space volumes represent imprecision in state, i.e., the volume is the subspace that contains the set of possible states.

Inherent in the Mirage model is the notion of measurable latency. Shannon’s model of a communication channel [Sh63] can be extended by including latency measurements (Figure 2.8). Latency is assumed to be either constant or predictable (effectively computable). Flow in the Shannon model is described as the motion of a volume along a communication pipe, and latency is the length of that pipe. As such, incorporation of latency into the model reveals a spatial aspect to the formerly topographic Shannon model².

¹Experience of my advisor, David J. Farber, suggests that these constraints may require failure compensation mechanisms, or at least a meta-communication that negotiates and monitors such constraints.

²Further discussion of the relationship between Mirage and Shannon’s model, along with a brief discussion of the latter, appears in Appendix A

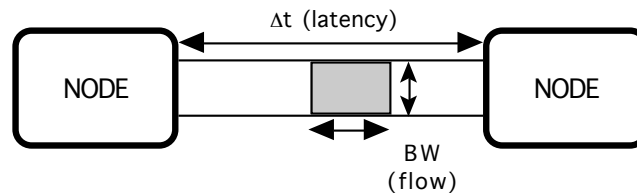


FIGURE 2.8
Mirage communication channel

The Mirage model describes communication among similar components of a network, called nodes. These nodes are separated by some minimum time lag, and through some maximum bandwidth communication channel; this separation characterizes the network for our purposes. The minimum time lag is representative of speed-of-light signal propagation delay, and the maximum bandwidth is representative of physical limitations on signal power per unit time. The nodes consist of some finite information storage; here the connectivity among the nodes and algorithmic power of each node is considered inconsequential to this preliminary analysis.

2.2.1. More definitions

The basis for this protocol model begins with state space transformations, extended to account for latency. This can be considered an extension of the FSM model, but with some exceptions.

First, FSMs usually describe a system in state space, whereas Mirage uses the powerset of this space. State spaces permit only a single value for each dimension in space, i.e., they accommodate only an individual point in the space. Mirage uses sets of these points, or volumes (ensembles, in either case) to describe the indeterminism of the knowledge of data in a remote node.

Mirage models the nondeterministic operation of a remote FSM, just as a deterministic finite automaton (DFA) models a nondeterministic finite automaton (NFA). A DFA state represents a set of states in the NFA, just as a Mirage state volume represents a set of states of a remote node. Mirage permits these volumes to vary, whereas the DFA model of an NFA fixes the state sets when the model is computed.

Mirage provides a way to describe a Turing machine system with explicit indeterminism and information delays. Traditional temporal extensions to FSMs describe the time bounds between transition transformations by describing the length of the time

arcs in the transition sequence. Mirage is concerned with the number of simultaneous arcs in the transition (i.e., the number of possible FSMs of the remote node), and so describes the transition as a function of time, not time as a function of the transition.

Mirage uses state space volumes as they are used to describe error correcting codes, because error induced by latency is similar to that caused by true corruption of the data. These analogs, to NFA-to-DFA transformations, to error correcting codes, and to temporal FSMs are all described in further detail as prior work in Chapter 3.

Because Mirage uses state space volumes to describe the possible states of a remote node, a set-based description of Mirage is most direct. This description is elaborated in Appendix E. Another example of its description in terms of Petri Nets is provided in Appendix F. The Mirage model is more general than either of these examples, i.e., it represents a model of which set notation and Petri Nets are instances.

Consider a set of nodes in the network. These nodes are considered completely connected, each pair connected with a finite maximum communication bandwidth and a finite minimum communication delay. Nodes possess finite **storage**. This storage is used both to denote the node's dedicated local storage and perceptions of the storage of remote nodes. The local **state** of a node is the local component of its storage.

One node has a **perception** of another (remote) node, which represents a subset of the possible states of the remote node. A node's **view** of the network consists of its own local state and the set of perceptions of the other nodes in the network. Mutually recursive knowledge is permitted, provided that the recursion is bounded and finite, as required by the fixed size of local storage at each node.

2.2.2. A description of the model

Like every good model, Mirage has some governing principles. Some of these can be considered axiomatic, i.e., self-evident truths used as the basis of the model. Some are tenets, i.e., beliefs common to a group (the network research community), but as yet unproven. Some are thesis statements, to be proven by this discussion. The common thread among these statements is that they are generally believed, but to date no model existed in which their truth could be debated. Mirage presents such a model. As such, we believe the most appropriate label is 'tenet'. The following are a few we think are important.

TENET 1: Communication is logical information synchrony among information separated entities

TENET 2: A protocol is a mechanism for maintaining communication

TENET 3: Information separated entities are separated in time*space, in units of *pending-information*

TENET 4: Bandwidth-delay product is a measure of *information separation*

When the system begins, each node is modeled as a point in state space, a particular individual state. As the system progresses, this single state evolves into a set of states. Which of these states exactly describes the remote node is not known; it is known that the state of the remote node is in this set. Thus from the initial individual point, volumes in state space result corresponding to the set of possible states of a node.

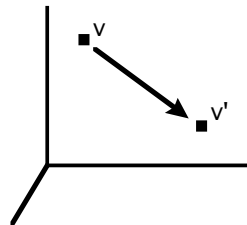


FIGURE 2.9

Shannon's model: states are points, transformations move points.

In Shannon's model [Sh63], information about remote nodes is modeled as a point in state space, and any operations that affect this information translate the point in space (Figure 2.9)¹. In Mirage, a remote node is modeled as a volume in state space, where operations become transformations of that space. Volume before the transform is denoted by a thin outline, volume after the transform is denoted by gray shading, and the action of the message or time is denoted by the thick outline. Time expands the volume of a space, transmission yields the union of the original volume (thin outline) with the transformed

¹In Shannon's model, the receiver gets a single message, and back-calculates the state of the sender (i.e., what was sent) from this. The goal is to determine, from a sequence of messages, the exact states of the sender. The sender does not model the receiver.

copy (thick outline), and reception collapses the volume to a sub-volume (Figure 2.10). These transformations are derived from the progression of time and the action of messages, as they leave the originating node and as they arrive at the destination. These are further discussed below.

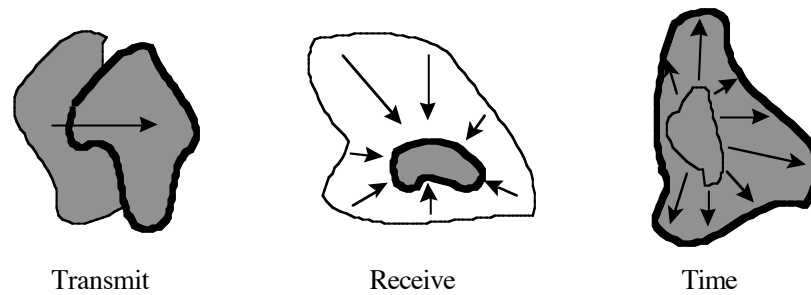


FIGURE 2.10
Visualization of state space volume transformations

2.2.2.1. Time

Time is modeled by the expansion of the state space volume. A node's local state does not grow in volume over time, because there is no imprecision in local state information, i.e., a node knows itself. The perceptions of remote nodes become less precise over time; it is this imprecision which the time transformation models.

Local state volumes cannot expand over time, because this would imply a violation of the Liouville thermodynamic theorem. The theorem indicates that local state spaces in physical systems cannot expand. This is not an issue here, because local state remains a point model; only remote state expands, and the theorem does not apply to perceptions (see Appendix D).

The transformation of a remote node's representation over a time interval is represented by a function that describes the known bounds on the variation of state space evolution over time. The extent to which the remote node is correctly modeled depends on the precision of this function, which is characterized by the amount of state space expansion per unit time, a form of induced entropy¹. The imprecision describes the difference between the node actual state and the perception of that state. The entropy

¹The notion that state reduction and volume ratios are related to entropy is not new; it has been discussed before, in [Sh63] and [Ha28].

change per unit time is a measure of the minimum bandwidth required to compensate for the entropy change, and can be bounded by the ratio of the volumes of state imprecision at the beginning and end of the time interval. Formulae expressing these relationships are described in Appendix E.

The computation function, which describes the evolution of the space over time as viewed at a distance, is a combination of the remote space evolving over time and the messages that it can receive over that time. Analysis of this function can be complex, because all possible permutations of messages and computing intervals must be accounted for. The computation function encodes known internal computation in the remote node, known bounds on the information received by that node, and known message emissions from that node (i.e., all known constraints on the remote node).

In the case where the time transformation is expressed by a probability density function (pdf), the computation function reduces to a convolution of the entire set of remote nodes over the set of probability density functions (pdfs) of the transformations of individual messages that can be received and a time transformation pdf. This reduction to convolutions requires that the time transformation is time invariant, i.e., it depends on the interval of elapsed time, but not the absolute time at which the interval occurs.

2.2.2.2. Receive

Receiving information collapses the perception volume of a remote node to a subspace of its former volume. Consider the case where a node receives data from a remote node. The received message affects part of the view that models the source of the message (the perception of the sender). There is a limit to the amount to which the message can reduce the volume of the perception (on average), because the volume reduction caused by the incoming information is bounded by the information content of that message, and because volume reduction is equivalent to reduction in entropy.

2.2.2.3. Transmit

Transmitting messages expands the perception of a remote node's state, similarly to the expansion induced by time. Rather than accounting for the temporal transformation of the remote space, the message itself causes the transformation of the space. The expanded space is logically OR'd with the entire original space, because the message may be received later, or lost, and both cases must be accounted for. The message affects a node's view by transforming its perception of the remote node to which the message is

sent. Again, the information contained in the sent message is limited by the transformation it effects, in terms of relative volumes of state spaces indicated (i.e., entropy).

Lost messages increase the state space of the perception, which is collapsed either when the state of the remote node indicates, or when the sender decides that the message loss can be ignored (i.e., a time-out that forces perception collapse). The use of time-outs to force collapse denotes the potential conflict with message loss assumption (i.e., that the reappearance of the message can cause an inconsistency in the perception).

2.2.3. Implications of the model

Several constraint conditions have already been presented, relating message effects on state space volume transformations and entropy limitations. Other correctness criteria have been presented relating received state to a subset of prior state. There are other constraints that are implied by the model, when further considered.

2.2.3.1. Lag and stability

The Mirage transformations can be considered with respect to stability. There are two variations on the definition of control stability. The first assumes that the state space reaches some fixed-point value, i.e., that it focuses on a specific point, within some variation, and remains there. The second maintains that the state space *entropy* is the value that becomes stable, i.e., that the imprecision of remote information reaches some fixed point, rather than the value of the state itself.

Consider the system whose state space is A . Over time, the space evolves to A' , whereas given communication (traditionally feedback/feedforward), it would become A'' (Figure 2.11). **Stability** is defined as A'' being a subset of A (after some minimum time¹) (Eq. 2.13), whereas **entropic stability** is defined as the volume of A'' being smaller or the same as the volume of A (Eq. 2.14). Neither criterion applies to the uncontrolled state A' .

Equation 2.13: $\forall_{\Delta t > t_{\min}} (A'' \subseteq A)$

¹Feedback stability commonly requires a minimum time lag, in order that the requisite circularity of information and action exists.

Equation 2.14: $\forall_{\Delta t > t_{\min}} (|A''| \leq |A|)$

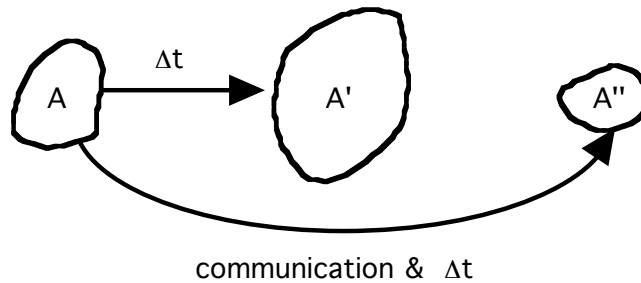


FIGURE 2.11
Control space evolution

2.2.3.2. Communicability

We can now express the most important formulae in the description of Mirage that defines the goal of the model. The space of a node consists of a finite number points, so a message could be sent, suitably guarded, for each point in this space. Assuming each message is of arbitrary length, destination state space can be transformed as precisely as desired, guaranteeing either stability criterion.

The trick is to send messages that are short enough, and to partition the space coarsely, to send as few of these messages as possible (each with a similarly brief guard), otherwise the required bandwidth would be unmanageable. The ultimate goal is a suitably efficient partition K (i.e., smallest number of component partitions in K) that satisfies these bandwidth criteria, and that ensures stability over all time frames beyond some minimum (Eq. 2.15)¹.

In Equation 2.15, A denotes the state of the remote node (i.e., the perception to be stably modeled), K denotes the partition, A'' denotes the perception thus stabilized, and $A.M$ denotes the state A as transformed by the set of delivered messages M . The goal is that for all intervals larger than some minimum, the system is entropically stable. The goal also includes ensuring that the smallest message set M be chosen, and that the set of

¹The number of components in the partition is the branch degree which alters the graphs (Figure 2.7). The message length is related to the limb length mentioned before and reflects the directed state path between branchings.

messages can be communicated in the time given. One message m_i is sent to each component of the partition K .

Here we denote this condition as a predicate, called *communicability*. The predicate holds where entropic stability is permitted by a given partition under the given communication bandwidth and latency parameters. This predicate can be used to specify the bandwidth and latency for a given partition, or to govern the search for a minimal partition, using bandwidth or latency as a measure.

$$\text{Equation 2.15: Given: } \left\{ \begin{array}{l} K - \text{a partition of A's perception at B,} \\ \quad \text{i.e., in a set of guarded messages } M \\ t_{\min} - \text{latency} \\ BW - \text{bandwidth} \end{array} \right.$$

Node A is COMMUNICABLE from node B if and only if:

$$\left\{ \bigvee_{\Delta t > t_{\min}} (|A''| \leq |A|) \text{ where } A'' = A: M \right\} \text{ and } \left\{ |M| \leq BW * t_{\min} \right\}$$

The condition under which such a partition exists, called *communicability*, represents the ability to communicate sufficiently with a remote node so as to ensure its stability, within the given bandwidth and latency criteria.

Stability need not be strictly guaranteed; in many cases, it is sufficient that the stability be highly probable. By assigning probabilities to each path in the state space evolution, the expected entropy change, rather than worst-case, can be considered. In this way statistical methods can be incorporated into the realization of the model. The volumes of state space become probability density functions (pdfs) in that space. Set operations on these volumes then become compositions of the pdfs. One case where similar statistical methods have already proven useful is clock synchronization [Cr89]; this is discussed in Chapter 4.

One important result of this description is that protocols which operate in high-latency environments require sufficient constraints on control space evolution. The stability of the system relies not only on the messages sent, but on the existing constraints of the computation function as well, because the state space is constrained by the interaction between the two.

The communicability formula is easiest to compute as a test. Given a fixed latency, does there exist a partitioning that results in a set of messages that can be sent during the round trip time and that result in stability in the perception of all remote spaces? If the time transformations of the remote nodes are sufficiently constrained, and if sufficient bandwidth exists to overcome any remaining imprecision via controlling messages, then the answer is “yes”.

Another way to view the situation is that error and lag are conjugate variables. A communication system that requires zero error thus requires infinite lag, to collect arbitrarily precise information about a remote node before making a decision. A system that tolerates infinite error also tolerates zero lag – the instant a query is asked, a reply is given. The lag can be zero, because the answer is allowed to be arbitrarily wrong.

2.2.3.3. Guarded messages

Thus far, the description of a protocol as a set of state space transformations is very similar to conventional statistical communication theory. Rather than using the state space volumes merely to describe or analyze the protocol, Mirage uses guarded messages to manipulate portions of these volumes, as part of the control mechanism [To89].

Guarded messages are similar to guarded commands as used in programming languages [Di76]. Prior to executing a set of guarded commands, the state of the machine is within some set of states (the union of the states specified by the guards); afterwards the state of the machine is within another set of states (the union of the states resulting from each guarded command)¹.

Guarded commands are used during programming to counter the uncertainty in the machine state during execution of the program; guarded messages do the same for communicating nodes. Guarded commands account for the latency between coding and execution (and uncertainties that arise in that interval), whereas guarded messages do the same for transmission latency.

Guarded messages permit the transmission of multiple sets of information to a remote node (Figures 2.12, 2.13). The perception of the remote node can be a volume in state space, so messages can be sent that are labelled with various regions of that volume.

¹ See the discussion on isotopy.

The remote node compares its current local state¹ to the label of the incoming message, and acts on the received information only if the two match.

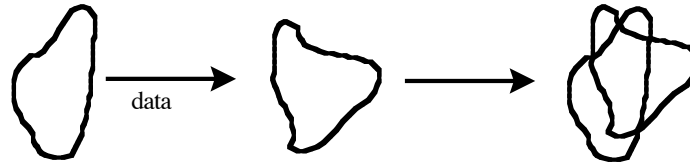


FIGURE 2.12

Entire space affected by unguarded message

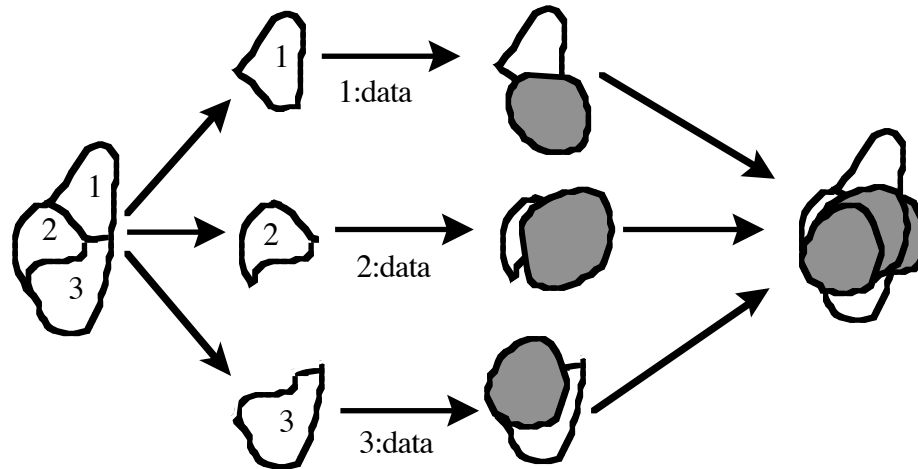


FIGURE 2.13

Guarded messages affecting partitions only

2.2.3.4. Isopotent sets

Isopotency describes a set of messages whose actions are equivalent, albeit by different actions on separate partitions of the state space. Guards differentiate the component messages of an isopotent set, to partition the space as desired.

The notion of isopotency leads to the distinction between a *physical message* and a *logical message*. A physical message is conveyed by the unguarded data, whereas a logical message is the message indicated by an isopotent set. *Isopotency* denoted the

¹ A node's local state is known as a point in state space. The imprecision in the perception of the state of a remote node causes the volume to be introduced.

single effect on the whole state space, as indicated by its component messages and their corresponding guards. The union of these actions is the logical message.

2.3. Discussion

There are several results to this abstract model, which are the consequence of this view of protocol models. Mirage is a model for a channel that accounts for imprecision in the communication, as introduced by latency.

We also have shown some formulae for the limitations of the ways in which latency can be accommodated (communicability). Most importantly, these formulae depend on the ways in which the state space can be partitioned, which in turn depends on semantic information about the state space. *The result is that protocol layering prohibits this partitioning, by hiding the semantic structure of the space.* Layering prevents the effective partitioning of the state space, and thus prevents any accommodation that could have occurred by sender-based anticipation using logical messages.

2.3.1. A channel with imprecision

The concept of a channel with imprecision can be elaborated. The limitation of existing protocols in gigabit WANs is due to an increase in the bit-latency. The increased amount of pending communication, i.e., information in the channel, requires modeling to permit channel utilization to increase. Further, this modeling can be performed only where prediction is possible, where the layering does not completely obscure some structure of the time transformation.

In Mirage, the characteristics of the data stream that are required in order to permit sender-based anticipation can be specified. The linearities in the stream express sender determinism, so that, regardless of the information communicated in the data of the stream, the sender knows which data to emit. Branching allows indeterminism in the sender, where the data sent depends on some unknown state of the receiver, permitting context sensitivity of the data stream.

The conclusion is that there are limitations to the utilization of the channel, and that these limitations can be overcome only if the internal structure of the data stream is examined. The sender can predict the next required information only if it knows what to expect. If these expectations are not fulfilled, round-trip delay penalties are incurred, in order to resynchronize the sender to the receiver's state.

We note the imperfection of the simulation. Although the Holodeck was successful in most details of the simulation, it lags when we strayed from the expected.

- Star Trek, the Next Generation
"Future Imperfect"

Some observations include the equivalence between infinite linearity and TCP-like existing protocols, and between infinite branching and NCP-like request/response protocols.

2.3.2. Looking into the structure of the stream

The structure of the stream can be described more completely in diagrams (Figures 2.14, 2.15). In the first case, bit-foreshortening (via increasing the channel transmission rate) causes the channel to be utilized less effectively. The result of the bit-foreshortening is an increase in the amount of the data stream that is "looked into" (fetched for transmission) during a round-trip time. So long as this stream continues to be linear, current protocols accommodate the lookahead (given sufficient buffer space).

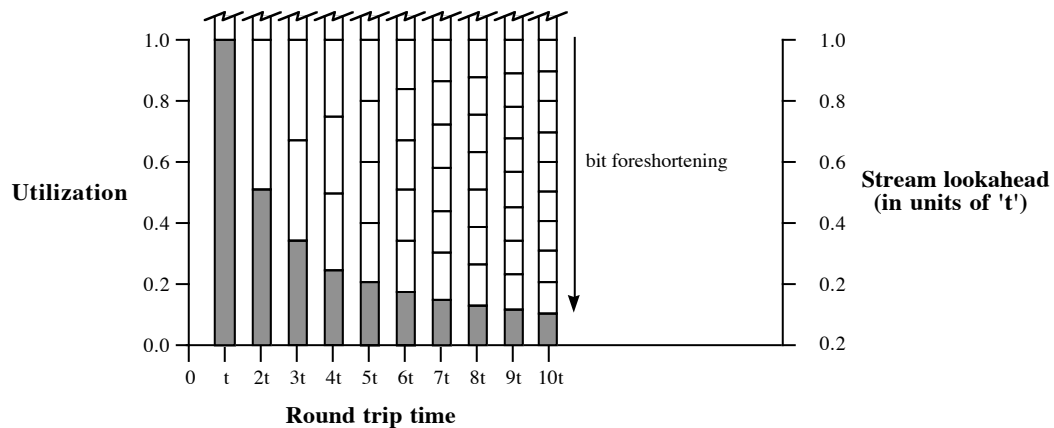
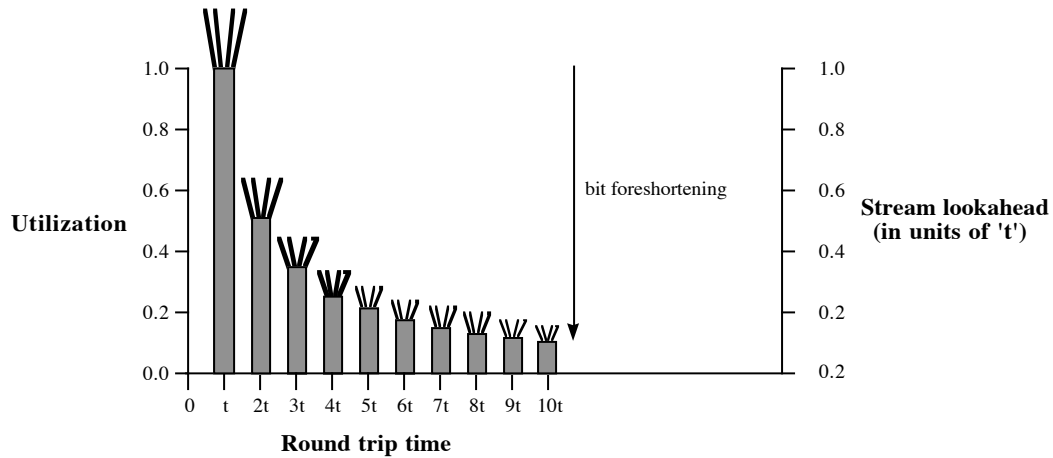


FIGURE 2.14

Bit foreshortening and its effect on lookahead / utilization

In the second case, bifurcations in the data stream cause the channel utilization to drop, because lookahead is permitted only until the first branching. By permitting the protocol to accommodate the branching, the remainder of the stream can be anticipated, albeit less effectively than the initial linear portion. This is presented in more detail in Chapter 5, in the discussion of the processor architectural implications of the protocol analysis of a processor-memory interaction.

**FIGURE 2.15**

Bit foreshortening and branching effecting utilization

2.3.3. Implementations

Mirage is an abstract model, using state space set transformations to describe the stream with imprecision. There are various ways to implement the model, such that the implementations are equivalent to the abstract form. Some of these implementations are direct analogs of the abstract model, suitably collapsed or condensed to permit their realization. A more specific example is investigated in a later chapter (Chapter 5).

2.3.3.1. Projections

The Mirage model is based on state space transformations, so one obvious implementation is the realization of a projection of the model, where some dimensions of the model are ignored, or groups of dimensions are collapsed into one.

The complete form of the model incorporates not only sets of points in state space, but also probabilities for each state. Where probabilities are not known, the worst case is assumed, which in information theory is the case where each possible state is equiprobable. This results in a uniform distribution among members of the set.

Each point in the state space is assigned a probability, where omitted points (points where the receiver's state cannot lie) have zero probability, so probability density functions can be used to express the distributions as a function of state value. Transformations of the state space in Mirage are then most completely expressed as pdf transformations, which are the convolutions of the individual pdfs.

These pdfs can be restricted to ease their implementation. For example, the state space volumes can be limited to be uniform and orthogonal, such that the value of the pdf (i.e., probability of the state space value being accurate) is independent in the dimensions of the variables of the state space. This is equivalent to a range-value system, where we can express the pdf as high/low values, between which the probability is uniform, and outside of which the state cannot lie. The result is an implementation that tests only bounds of the state space, rather than true likelihood. Such a system would be useful in real-time systems, or fault-tolerant systems.

The pdf can also be replaced with an average/standard deviation pair, but only where the pdf is orthogonal and has an internal structure that is adequately approximated by these first order statistics. This is useful in ‘aiming’ protocols, where boundaries are not an issue, but localization of a shared value is; such is the case in clock synchronization protocols.

2.3.3.2. Granularity

Another consequence of the Mirage model is an acknowledgment of the desired dichotomy between the state space of the receiver and that same space as modeled in the sender, for the purposes of controlling data anticipation.

The receiver has a state space that is a fine partitioning of the state space, fine enough to express the limit of the granularity of the space. The fineness of the granularity of this space is defined by the degree to which the receiver partitions (or does not partition) it.

The sender’s view of the receiver is a more coarse partitioning of this space. The coarseness of the granularity reflects the ‘need to know’ principle of this model – the sender models the state of the receiver only so explicitly as it needs to, in order to permit effective use of the channel. From the equations of stability and control, a larger granularity means that fewer messages need to be sent to anticipate the partitions of the receiver’s state, which in turn allows the individual messages (to each component of the partition) to be longer.

The result is a system in which the sender models the receiver only to the extent that it must in order to send data in anticipation, and the receiver completes the structure of the partition down to the level of each state value. The sender needs to model only so far as to anticipate, but the receiver will use the sent data along with local information to achieve the desired computation.

2.4. Insights

There are a few useful insights from the investigation of this abstract model. There are several types of information in a system: direct, indirect, and a new kind, virtual. Error and latency are related, as conjugates. Finally, entropy and communication have been discussed in more abstract terms.

2.4.1. Kinds of information

In distributed systems, two kinds of information are usually described: direct, and indirect. *Direct* information is data about another node which that node explicitly sent. *Indirect* information is inferred data, sometimes called ‘common knowledge’[Ha84], [Go88], which is information about another node that is inferred from global constraints and direct information from the rest of the system. Indirect communication occurs when we know a-priori that 3 of 5 nodes hold a copy of a single datum, and we have received 2 replies where the datum is absent; we can immediately conclude from the global constraint and the received information that the remaining 3 nodes contain the datum.

Mirage suggests another kind of communication, that of virtual data. *Virtual* communication is not the result of any direct communication; it is the consequence of the lack of communication over time, and some specific constraints about the node ‘not heard from’. Virtual communication results from the time transformation, i.e., how the state space of the remote node behaves over time, unless otherwise heard from.

2.4.2. Error and latency as conjugates

Mirage indicates ways in which error and latency are conjugates. C. Shannon noted that error could be reduced as small as desired by encoding information over a long enough sequence [Sh63]; the process of encoding induces a latency of the length of the encoding sequence. Mirage shows how to reduce latency by anticipation, with a corresponding increase in the error of the perceived state of the remote node.

2.4.3. Entropy

When the state space volumes were described as corresponding to entropy, a set of constraints was introduced, by analogy. If the log of the state volume is entropy, then all the conventional physical constraints on entropy should apply.

For example, in physical systems, entropy always increases. In this system, entropy increases with time and with emitted messages (i.e., entropy increases in the sender when it sends data to the receiver). Mirage permits the collapse of these volumes, when information is received, contrary to traditional physical laws (i.e., entropy decreases). However, because data is created in the nodes of a network, as has been claimed in biological systems [Ja55], the state space volume may reduce.

2.4.4. Constraints

There is an interaction between error, latency, communication, and the need for constraints about the ways in which the receiver traverses the state space.

Time and space and thought aren't the separate things they appear to be.
These things are dangerous to say.
- Star Trek, the Next Generation
"Where no one has gone before"

2.4.5. Contrasts & comparisons

There are comparisons between the abstract Mirage model and aspects of forward error correction (FEC). FEC uses the same kinds of state space volumes, where the state space is divided into equivalence classes, so that when any point in the class is received, the canonical member of the class is presumed to have been sent. Mirage uses what can be considered a dynamically reconfiguring FEC scheme to communicate the remote state.

Mirage also uses multiple possible messages, in the isopotent set. It explores the state space in a breadth-first sequence (BFS), rather than depth first, as in conventional receiver-based anticipation. The use of BFS techniques removes the need for sender rollback, because all possible states are covered with the traversal of each level of the tree of possibilities of communication.

Other anticipatory schemes sometimes use replication of multiple independent DFS explorations, in which the results are collected upon termination of each probe [Sm89]. Although there are analogies to our method, Mirage relies specifically on the differences which BFS affords; specifically those removing the need for rollback. Further, there is a difference in the characterization of the state space in these two methods. [Sm89] assumes one of the DFS paths will terminate before others, and that which path is shortest is not computable before the actual probing of the space. In Mirage, the space is computable, with sufficient delay. Mirage tries to get around the delay of communicating exact state by permitting the states to be mere approximations.

There are direct correspondences between the way in which the receiver filters messages according to guards, specifically related to the Universal Receiver Protocol (URP) [Fr89] and the Knockout switch [Ye87]. These similarities, as well as other prior work, are discussed in Chapter 3.

Finally, there is an interesting comparison between the implications of Mirage and the selection of optimal buffer sizes in sliding-window protocols. The optimum buffer size for communication is the size of the round trip bandwidth-delay product, and inefficiencies result if the buffer size available is less than this product, addressing the TCP/sliding-window protocol situation. Mirage asks the question ‘what happens if the buffer size indicated is negative’, i.e., if the round-trip time is much larger than the maximum possible window (i.e., linear lookahead in the data stream). In this case, Mirage suggests an advantage.