4

Flow Analysis

The results of requirements analysis—the requirements specification and requirements map—list and describe requirements that have been gathered and derived for a network. Many of these are performance requirements (capacity, delay, and RMA) for users, their applications, and devices. Such performance requirements, along with the locations of users, applications, and devices from the requirements map, form the basis for estimating where traffic will flow through the network. *Flow analysis* is the process of characterizing traffic flows for a network: where they are likely to occur and what levels of performance they will require.

The intent of flow analysis is not to show every possible flow in a network, but rather to show those flows that will have the greatest impact on the network architecture and design. This is often a small fraction of the total set of flows for a network.

4.1 Objectives

In this chapter you will learn how to identify and characterize traffic flows. We discuss individual, composite, and critical flows and how to determine when each applies. You will learn mechanisms to help identify and characterize flows, including data sources and sinks and flow models. Finally, you will learn about flow specifications, where performance requirements are combined for a flow or group of flows. Flow specifications are used as input to the network architecture and design processes, covered later in this book.

4.1.1 Preparation

To be able to understand and apply the concepts in this chapter, you should be familiar with client–server, peer-to-peer, distributed computing, and other models for how applications and devices interact with one another.

4.2 Background

Now that we have the user, application, device, and network requirements developed, listed, and mapped in the requirements specification, we further analyze these requirements based on their end-to-end characteristics. We use the concept of a *flow*, which, for an end-to-end connection, has constant addressing and service requirements, to combine performance requirements in a useful way. These characteristics are analyzed and combined, per flow, into a flow specification or flowspec. This is used for capacity and service planning.

This chapter introduces flows and flow concepts, data sources and sinks, and flow models which will help us to identify, size, and describe flows. We also develop cumulative performance requirements for each flow.

4.3 Flows

Flows (also known as *traffic flows* or *data flows*) are sets of network traffic (application, protocol, and control information) that have common attributes, such as source/destination address, type of information, directionality, or other end-to-end information.

Figure 4.1 illustrates this concept.

Information within a flow is transmitted during a single session of an application. Flows are end-to-end, between source and destination applications/devices/users. Since they can be identified by their end-to-end information, they can be directly linked to an application, device, or network, or associated with an end user. We can also examine flows on a link-by-link or network-by-network basis. This is useful when we want to combine flow requirements at the network or network-element levels.

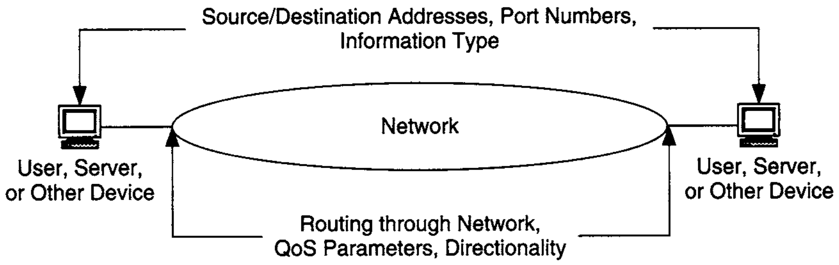

Source/Destination Addresses, Port Numbers, Information Type
Network
User, Server, or Other Device
User, Server, or Other Device
Routing through Network, QoS Parameters, Directionality

FIGURE 4.1 Flow Attributes Apply End-to-End and Throughout Network

Flow Characteristics	
Performance Requirements	Capacity (e.g., Bandwidth)
	Delay (e.g., Latency)
	Reliability (e.g., Availability)
	Quality of Service Levels
Importance/ Priority Levels	Business/Enterprise/Provider
	Political
Other	Directionality
	Common Sets of Users, Applications, Devices
	Scheduling (e.g., Time-of-Day)
	Protocols Used
	Addresses/Ports
	Security/Privacy Requirements

FIGURE 4.2 Common Flow Characteristics

Common flow characteristics are shown in Figure 4.2.

Flow analysis is an integral part of the overall analysis process. Flows are where performance requirements, services, and service metrics are combined with location information to show where performance and service are needed in the network. Flow analysis provides an end-to-end perspective on requirements and shows where requirements combine and interact. It also provides some insight into the degrees of hierarchy and diversity needed in the architecture and design. In addition, as we will see in the design process, this analysis also provides information that can be useful in choosing interconnection strategies, such as switching, routing, or hybrid mechanisms.

Most flows are bidirectional and can be represented as either a single, double-sided arrow with one or two sets of performance requirements, or as two separate flows, each with its own set of requirements. A single-sided arrow with one set of performance requirements represents a unidirectional flow. Figure 4.3 shows these cases.

Flows provide a different perspective on traffic movement in networks: they have logical as well as physical components; and they allow traffic to be coupled with users, applications, or devices. Flows are becoming increasingly important in the analysis, architecture, and design processes.

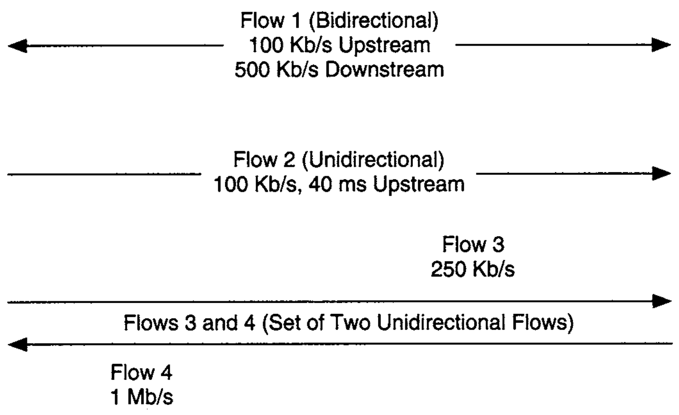


FIGURE 4.3 Flows Are Represented as Unidirectional or Bidirectional Arrows with Performance Requirements

We examine two types of flows: individual and composite. We will see that the aggregation of requirements and flows in the network due to hierarchy leads to composite flows, and that this can happen in the access network as well as in the backbone.

4.3.1 Individual and Composite Flows

An *individual flow* is the flow for a single session of an application. An individual flow is the basic unit of traffic flows in this book; they are either considered individually or combined into a composite flow. When an individual flow has guaranteed requirements, those requirements are usually left with the individual flow and are not consolidated with other requirements or flows into a composite flow (Figure 4.4).

This is done so that the flow's guaranteed requirements can be treated separately from the rest of the flows. Individual flows are derived directly from the requirements specification, or are estimated from our best knowledge about the application, users, devices, and their locations.

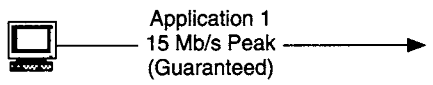


FIGURE 4.4 Individual Flow for a Single Application with Guaranteed Requirements

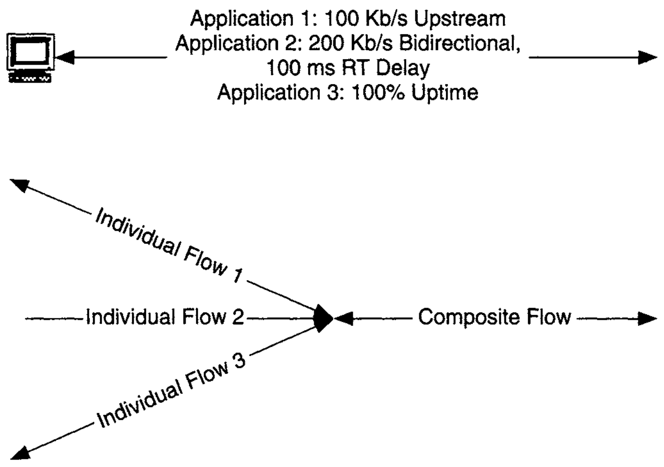

Application 1: 100 Kb/s Upstream
Application 2: 200 Kb/s Bidirectional,
100 ms RT Delay
Application 3: 100% Uptime
Individual Flow 1
Individual Flow 2
Composite Flow
Individual Flow 3

FIGURE 4.5 Example Composite Flows

A *composite flow* is a combination of requirements from multiple applications, or of individual flows, that share a common link, path, or network. Most flows in a network are composites (Figure 4.5).

More examples of flows are presented in Figure 4.6. The first example is an individual flow, consisting of a one-way delay requirement for a single session

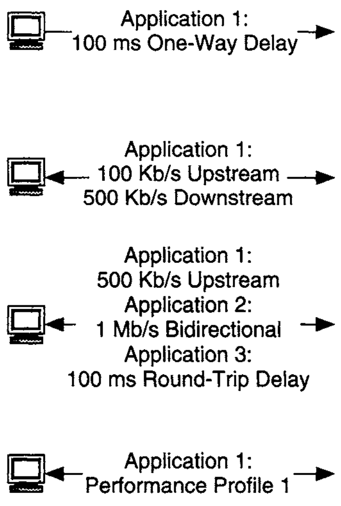


FIGURE 4.6 Flow Examples

of an application. Note that this flow is unidirectional. The second example is also of an individual flow, but in this case the capacity requirements are given in each direction. Since they are different, they are listed independently. The definitions of upstream and downstream as used here denote directionality based on the source and destination of the flow, where *upstream* indicates the direction toward the source and *downstream* is the direction toward the destination. Upstream is often toward the core of the network, while downstream is often toward the edge of the network, particularly in service provider networks. Another way to show a bidirectional flow is with two separate arrows, one upstream and the other downstream, each with its own performance requirement.

The third example is a composite flow, listing requirements from three separate applications at the same source. The last example uses a performance profile to describe the flow's performance requirements. A profile is often used when many flows have the same requirement, as this makes it easier to apply the same requirements to multiple flows: A pointer to the profile is sufficient to describe the requirements, instead of having them written out each time. This is especially useful when the requirements are long or when they are consistent across many flows. The flow in this example could be either an individual or a composite flow, depending on the contents of the profile.

Performance requirements for individual flows and composite flows are determined through development of a flow specification for the network, which is discussed at the end of this chapter.

4.3.2 Critical Flows

Some flows can be considered more important than others, in that they are higher in performance or have strict requirements (e.g., mission-critical, rate-critical, real-time, interactive, high-performance), while some flows may serve more important users, their applications, and devices. Such flows are called *critical flows*. In this chapter we examine how to determine when flows are critical. When prioritizing which flows get attention in the network architecture and design, critical flows usually come first. This is usually the case; however, individual flows with guaranteed requirements might also be considered first in the architecture and design. Prioritizing flows is discussed at the end of this chapter.

Flows are described in this fashion in order to make it easier to understand and combine requirements. Composite flows combine the requirements of individual flows, while individual flows can show guaranteed requirements that need to be considered throughout the end-to-end path of the flow. All of these flows are important in the architecture and design of the network. The descriptions of the types of flows that we develop in the flow specification help us to define the architecture and choose the technologies and services that best fit the customer's needs.

Throughout this chapter we analyze flow requirements to determine where they apply and when they contribute to composite flows, and, if they do, how to combine their performance requirements accordingly. Some networks have flows that indicate single-tier performance, others have flows that indicate multi-tier performance, and still others have predictable (stochastic) and/or guaranteed performance. Often the few flows that require high, predictable, and/or guaranteed performance are the ones that drive the architecture and design from a service (capacity, delay, and RMA) perspective, while all flows drive the architecture and design from a capacity perspective. The architecture and design processes accommodate both of these perspectives.

4.4 Identifying and Developing Flows

Flows can usually be identified and developed from information in the requirements specification: user, application, device, and network requirements; user and application behavior (usage patterns, models); user, application, and device location information; and performance requirements. The more thorough this information is, the better the resulting flows will be.

At this point in the analysis process we have sets of requirements and mappings of application and/or device locations. We have not made any choices of networking technologies, network devices, or vendors. It is important that during the flow analysis process we do not constrain flows to existing networks, topologies, or technologies. We want to be free in determining the compositions and locations of flows for our network, so that the flows drive our architectural and design choices. Thus, flows are determined based on the requirements and locations of the applications and devices that generate (source) or terminate (sink) each traffic flow.

The process for identifying and developing flows consists of identifying one or more applications and/or devices that you believe will generate and/or terminate traffic flows. Once you have chosen which applications and devices to focus on, you use their requirements from the requirements specification and their locations from the requirements map. Based on how and where each application and device is used, you may be able to determine which devices generate flows and which devices terminate flows (flow sources and sinks). Some common flow models are provided in Section 4.6 to help you with this process. Once you have identified each flow and determined its composition and location, you combine the performance requirements of flows into a flow specification. This process is shown in Figure 4.7.

From an application perspective, some common approaches to identifying flows include:

- Focusing on a particular application, application group, device, or function (e.g., videoconferencing or storage)
- Developing a "profile" of common or selected applications that can be applied across a user population
- Choosing the top *N* (e.g., 3, 5, 10, etc.) applications to be applied across the entire network

You may choose to consider some or all approaches for your network. Each approach is outlined in the following sections.

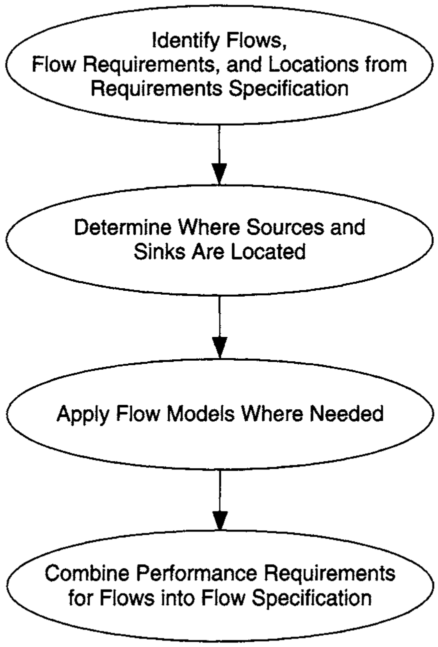

Identify Flows, Flow Requirements, and Locations from Requirements Specification
Determine Where Sources and Sinks Are Located
Apply Flow Models Where Needed
Combine Performance Requirements for Flows into Flow Specification

FIGURE 4.7 The Process for Identifying and Developing Flows

4.4.1 Focusing on a Particular Application

When focusing on an application, application group, device, or function, the idea here is to consider one or more applications that will likely drive the architecture and design—namely, those that are high performance, mission-critical, rate-critical, real-time, interactive, predictable, and/or guaranteed. By focusing on one or a few applications, you can spend more time determining their flows, as opposed to spreading your time out across many applications. Choose what to focus on and select the relevant information from the requirements specification.

Example: Data Migration

From requirements specification, for a single session of each application:

Application 1: Staging data from user devices

Capacity 100 Kb/s; Delay Unknown; Reliability 100%

Application 1: Migrating data between servers

Capacity 500 Kb/s; Delay Unknown; Reliability 100%

Application 2: Migration to remote (tertiary) storage

Capacity 10Mb/s; Delay N/A; Reliability 100%

You can use the location information to map out where the users, applications, and devices are, and develop a map if you don't already have one. Figure 4.8 is an example of such a map.

Using the information on user and application behavior, you determine or estimate where flows will occur. This can be between networks, device groups (better), or individual devices (best). Figure 4.9 shows where flows will occur between devices for Application 1.

Once flows are mapped, you apply performance information to each flow. Figure 4.10 shows this for the Central Campus portion of the storage application environment. Whereas Figure 4.9 shows flows between devices, Figure 4.10 simplifies this view by showing flows between buildings. Either method can be used, depending on the size of the environment and the number of flows you need to show. Usually the between-device method (Figure 4.9) is used first to estimate where flows will be, and then the second method (Figure 4.10) is used to simplify the flow diagram.

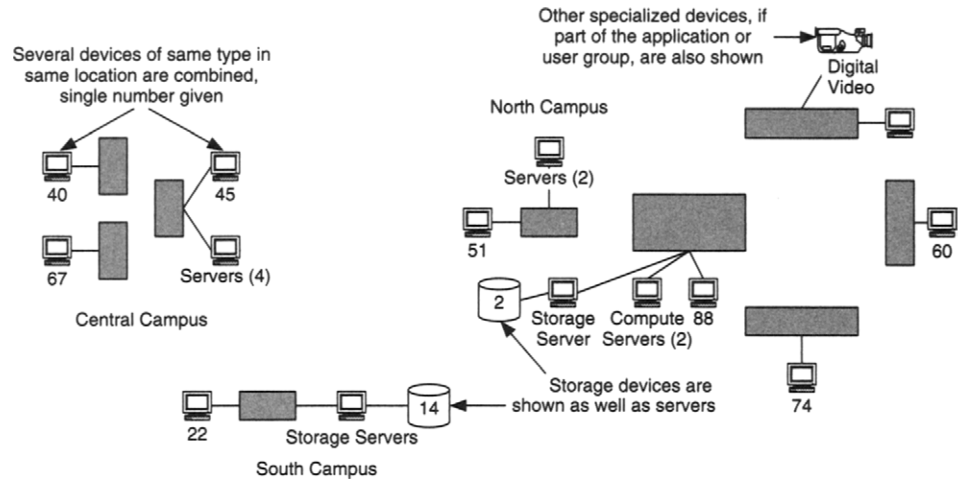

Several devices of same type in same location are combined, single number given
40
45
67
Servers (4)
Central Campus
Other specialized devices, if part of the application or user group, are also shown
Digital Video
North Campus
Servers (2)
51
60
2
Storage Server
Compute Servers (2)
88
Storage devices are shown as well as servers
74
22
14
Storage Servers
South Campus

FIGURE 4.8 A Map of Device Locations for a Network

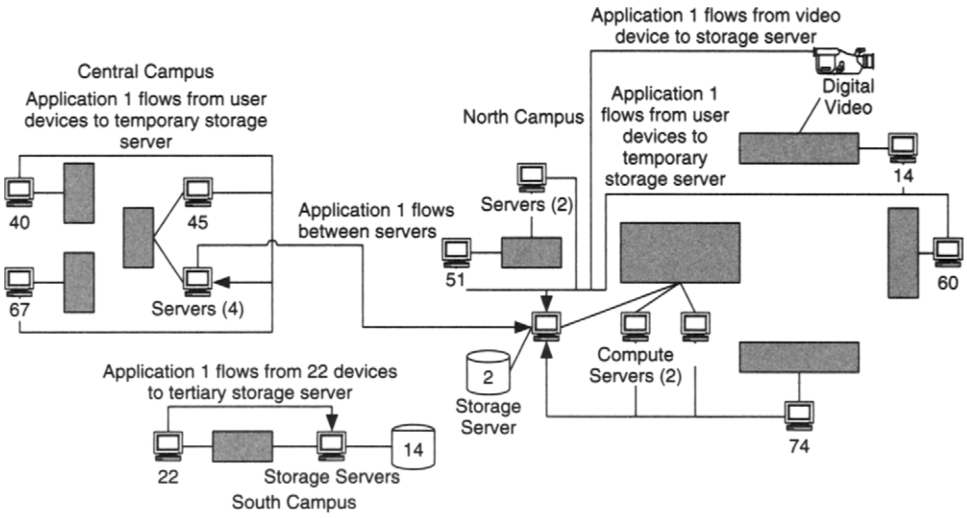

Application 1 flows from video device to storage server
Central Campus
Application 1 flows from user devices to temporary storage server
Digital Video
North Campus
Application 1 flows from user devices to temporary storage server
14
40
45
Application 1 flows between servers
Servers (2)
51
60
67
Servers (4)
Compute Servers (2)
Application 1 flows from 22 devices to tertiary storage server
2
Storage Server
74
22
14
Storage Servers
South Campus

FIGURE 4.9 Flows Estimated between Devices for Application 1

In Figure 4.10 flows F1, F2, and F3 represent the single-session performance requirement for each building for Application 1. At some point in this process the performance requirement will need to be modified to represent the estimated performance required by all users in each building (40, 67, and 45 users in the

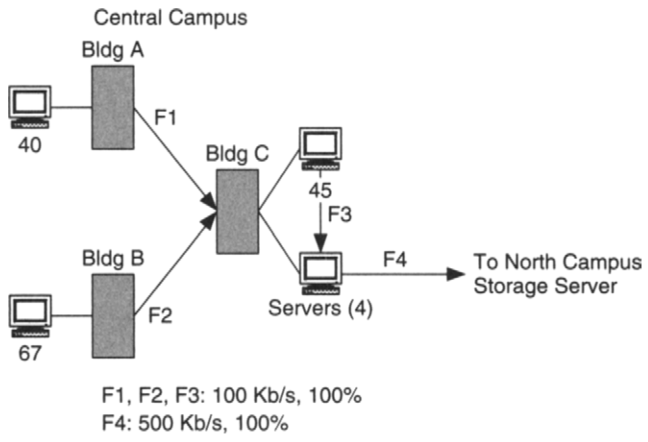


FIGURE 4.10 Performance Information Added to Central Campus Flows for Application 1

buildings at Central Campus). F4 represents the performance requirement for the server–server flow between Central and North Campuses.

Note that in this figure flows F1 and F2 are between Buildings A/B and C, while flow F3 is between devices. Since the 45 user devices and four servers are in the same building, we have to show the flows between devices. If we were showing flows within Building C, it would look like Figure 4.11.

This diagram also introduces a *flow aggregation point*, which allows us to show multiple flows being consolidated at a location. This is useful in the design process,

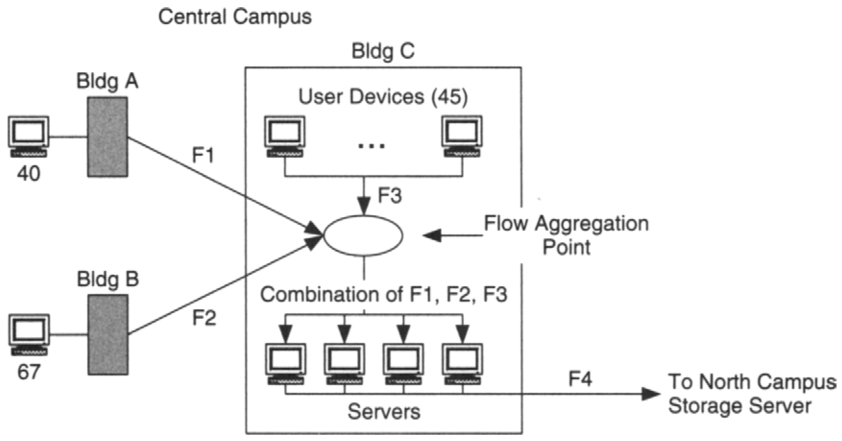


FIGURE 4.11 Central Campus Flows for Application 1 Expanded with Building C

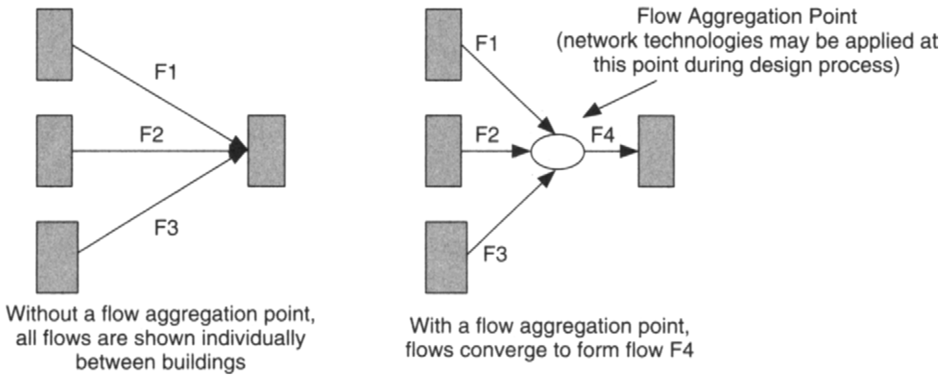

F1
F2
F3
Without a flow aggregation point, all flows are shown individually between buildings
F1
F2
F3
F4
Flow Aggregation Point (network technologies may be applied at this point during design process)
With a flow aggregation point, flows converge to form flow F4

FIGURE 4.12 Consolidating Flows Using a Flow Aggregation Point

when we are looking at technologies and interconnection strategies for such locations in the network. For example, Figure 4.12 shows what flows will look like between buildings with and without a flow aggregation point.

4.4.2 Developing a Profile

Sometimes a set of common applications apply to a group of users or to the entire set of users. When this is the case a profile or template can be developed for those applications, and each flow that fits the profile is identified with that profile's tag. Thus, instead of trying to replicate flow information for similar flows, you can use the profile, saving time and effort.

There are also times when flows have the same performance requirements, regardless of whether or not they share the same set of applications. When flows have the same performance requirements, a profile can be used to simplify their information as well.

Figure 4.13 shows a profile applied across the user population for Application 1. Instead of showing identical performance requirements at each flow, a common profile is shown for flows with the same performance requirements. This helps to reduce duplication of information on the map, also reducing clutter.

In this figure, P1 is the tag associated with flows having the following performance requirements: capacity = 100 Kb/s, reliability = 100%. There are six flows in this diagram with those performance requirements, all of which are tagged as P1. Other flows in this figure have different performance requirements. Flow F4 (also shown in Figures 4.10 and 4.11) has different performance requirements: capacity = 500 kb/s, reliability = 100%. Flow 5 combines the performance

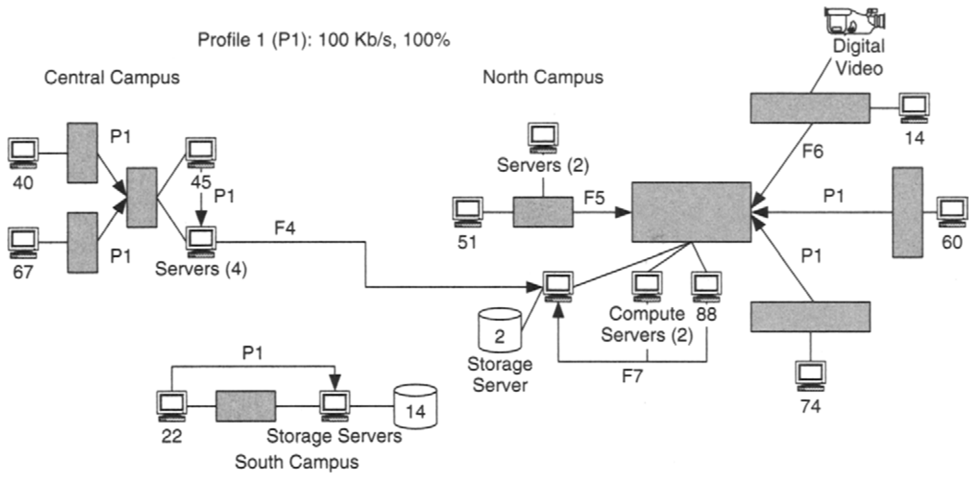


FIGURE 4.13 A Performance Profile (P1) Applied to Multiple Flows with the Same Performance Characteristics

requirements of 51 users (which, for Application 1, would be P1) with those of the two servers in that building. Flow F6 combines the performance requirements of 14 users (again, this would be P1) with that of the digital video device. F7, like F5, combines the performance requirements of users (88 in this building) with those of the two compute servers.

Note that Figure 4.13 shows flows as arrows between buildings and between devices within a building (except for flow F4, which in this example is easier to show as a flow between devices than as a flow between buildings). We could also choose to show the flows only between devices, as in Figure 4.9. If you compare these two figures, you will see that showing arrows between buildings simplifies the diagram. However, you may need to complement such a diagram with diagrams of flows within buildings, such as that shown in Figure 4.11.

4.4.3 Choosing the Top *N* Applications

Finally, choosing the top *N* applications for your network is a combination of the first two approaches. It is similar to the first approach, however; instead of one particular application (or maybe two applications), you use three, five, or perhaps ten. It is also similar to the second approach, in that the result can also be an application profile. These applications are the "top *N*" in terms of helping with the success of that organization, which may be inferred by their degrees of usage, number of users, number of devices/servers, or performance requirements.

Example: Top 5 Applications

1. Web Browsing
2. Email
3. File Transfer
4. Word Processing
5. Database Transactions

This approach reduces the set of possible applications to a number that can be analyzed. However, whenever you eliminate applications from this set, you need to ask the question, "If I meet the requirements of these top *N* applications, will the requirements for those applications I eliminated be met?" The intent of this approach is to determine which applications represent the most important requirements for that network. These "top *N*" applications are likely to be the performance drivers for the network. As such, if you meet the needs of these applications, you are likely to meet the needs of all applications for that network. This is described in more detail during the development of the flow specification.

The list of "top *N*" applications should be as precise as you can make it. For example, Application 5 above (Database Transactions) may actually be time-card data entry into an ERP database. This would be a more precise description, resulting in more precise flow composition and location information. Whenever possible, include usage scenarios for each application.

You may also use different approaches for different parts of the network. For example, it is common to include the top *N* applications that apply everywhere as well as profiles for selected locations and to focus on an application, device, or group in other locations, as in Figure 4.14.

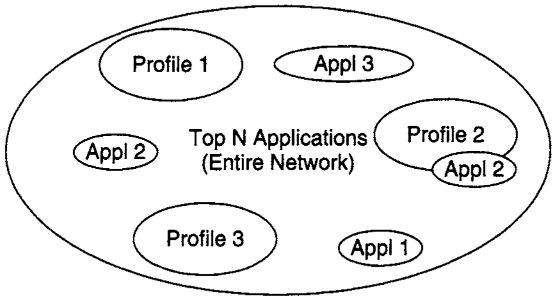


FIGURE 4.14 A Project May Incorporate Multiple Approaches in Choosing Applications

4.5 Data Sources and Sinks

Data sources and sinks can help provide directionality to flows. A *data source* generates a traffic flow, and a *data sink* terminates a traffic flow. To help show data sources and sinks in a diagram, the convention shown in Figure 4.15 is used. Data sources are represented as a circle with a dot in the center, and a data sink is represented as a circle with a cross (i.e., star or asterisk) in the center. These are two-dimensional representations of a plane with an arrow coming out of it, as in traffic flowing out of a source, or a plane with an arrow going into it, as in traffic flowing into a sink. By using these symbols we can show data sources and sinks on a two-dimensional map, without the need for arrows.

Almost all devices on a network produce and accept data, acting as both data sources and sinks, and there are some devices that typically act as either a source or sink. In addition, a device may be primarily a data source or sink for a particular application.

Some examples of data sources are devices that do a lot of computing or processing and generate large amounts of information, such as computing servers, mainframes, parallel systems, or computing clusters. Other (specialized) devices, like cameras, video production equipment, application servers, and medical instruments, do not necessarily do a lot of computing (in the traditional sense) but can still generate a lot of data, video, and audio that will be transmitted on the network (Figure 4.16).

A good example of a data sink is a data storage or archival device. This may be a single device, acting as a front end for groups of disks or tape devices. Devices that manipulate or display large quantities of information, such as video editing or display devices, also act as data sinks (Figure 4.17).

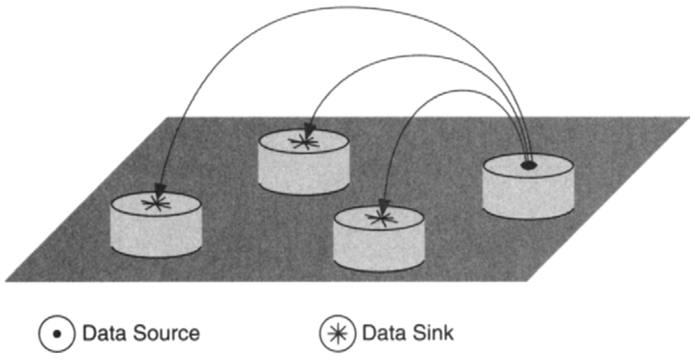


FIGURE 4.15 Conventions for Data Sources and Sinks

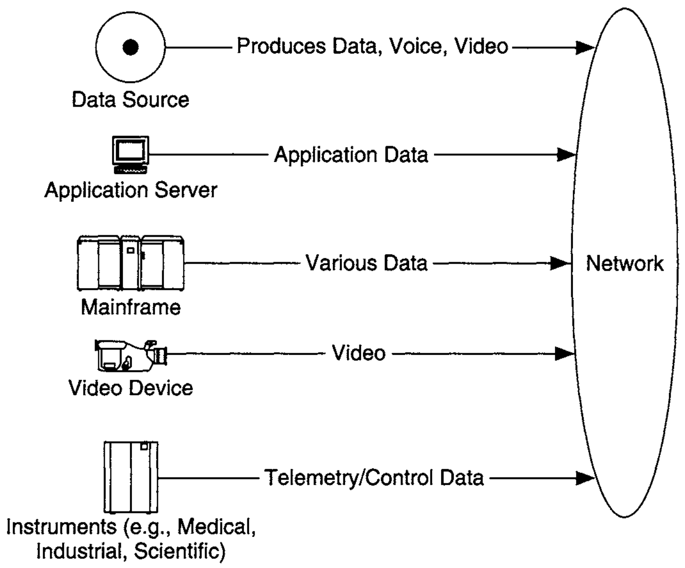


FIGURE 4.16 Example Data Sources

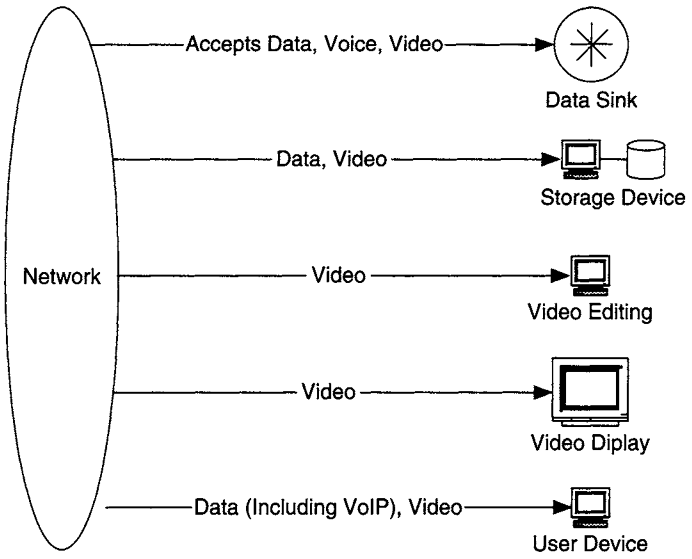


FIGURE 4.17 Example Data Sinks

Example 4.1. Data Sources and Sinks for Data Migration Applications.

As an example, let's consider the data migration applications. Recall that Figure 4.8 shows device location information for these applications. Shown on this map are storage and compute servers, a video source, and groups of desktop devices for each building. Note that a total number of desktop devices are given; this is done to simplify the map and flows. If more detail is needed, this group could be separated into multiple groups, based on their requirements, or single devices can be separated out. If necessary, a new map could be generated for just that building, with substantially more detail.

This service has two applications. Application 1 is the frequent migration of data on users' desktops, servers, and other devices, to storage servers at each campus. As part of this application, data are migrated from the server at Central Campus to the server at North Campus.

The server at South Campus is also the archival (or long-term) server for these campuses. Application 2 is the migration of data that have been collected over a period of time (e.g., 1 day) from the server at Central Campus to the archival server at South Campus.

When, for Application 1, we add the data sources and sinks to Figure 4.13, we get the diagram in Figure 4.18. In this figure all devices are labeled as a data source, sink, or both. All of the user devices at each campus are data sources. The servers at Central Campus act as a data sink for flows from user devices at that campus, and as a data source when data migrates to the server at North Campus (flow F4). The server at South Campus is a

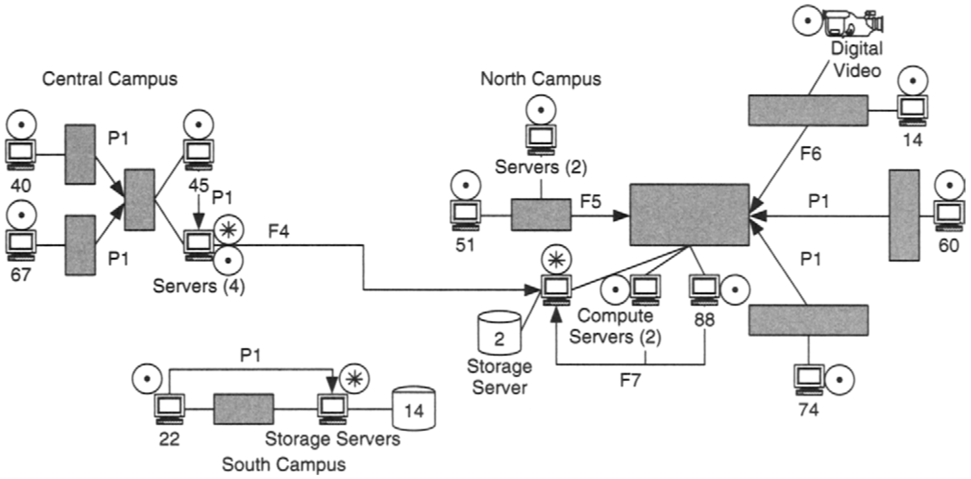


FIGURE 4.18 Data Sources, Sinks, and Flows Added to the First Part of the Application

data sink for user flows at that campus, and the servers at North Campus are data sinks for all devices at that campus, including servers and digital video, as well as for flow F4 from Central Campus.

With the sources and sinks on the map, along with arrows between them to indicate potential traffic flows, we are beginning to have an idea of where flows occur in this network for this application.

We do the same thing for Application 2, with the result shown in Figure 4.19. For this part of the application the desktop devices and video source are not involved, so they are not shown as either sources or sinks of data. The server at Central Campus that was a data sink for Application 1 now becomes a data source for Application 2, sending data to the archival server (shown as a data sink). The flows for this part of the application are much simpler, merely a single flow between the storage server and archival device. This is shown in two ways: as a flow between buildings (Option 1, the standard convention used so far in this example), and as a flow between devices at separate buildings (Option 2). While Option 2 is not common practice, it is used here since there are only two devices (and one flow) for this application.

These applications are shown as separate events in order to clarify the flows and the roles of the storage servers. We could, however, put both parts of the application together on the same map. We would want to make sure that the resultant map is still clear about when devices are data sources and sinks.

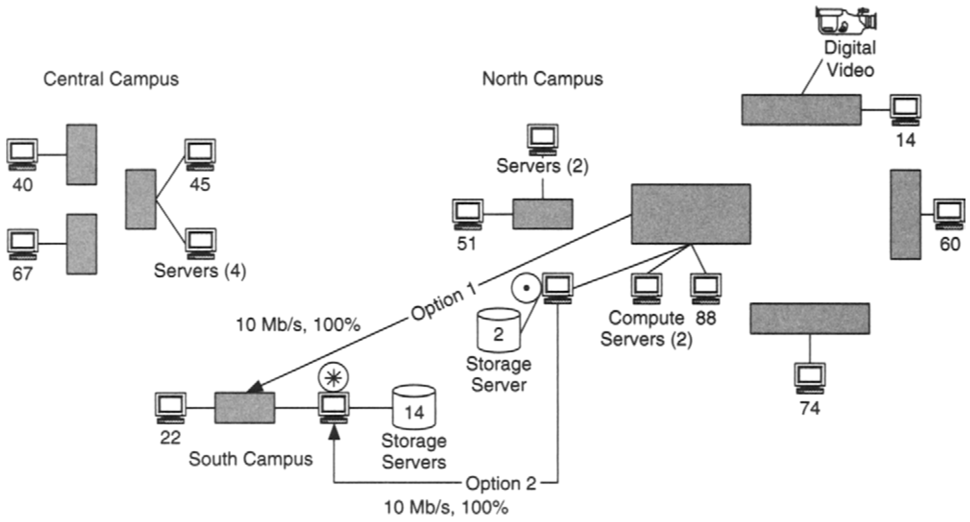


FIGURE 4.19 Data Sources, Sinks, and Flows Added to Application 2 (Two Options Shown)

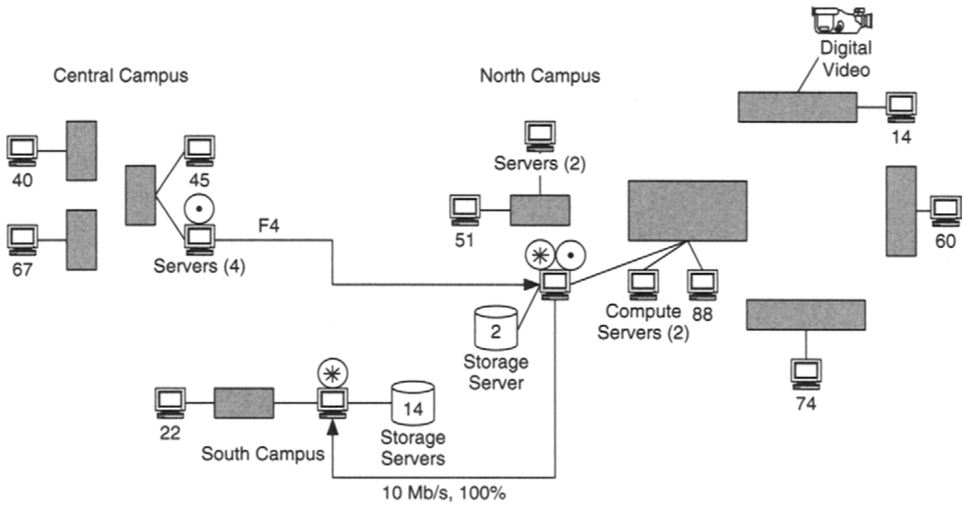


FIGURE 4.20 Data Migration Application with Server–Server Flows Isolated

One way to simplify this map is to focus only on flows between storage servers, data sources, and sinks. Figure 4.20 shows this case, with both applications on the same map.

You may have noticed in our example that flows occur at different times of the day, and can vary in schedule (when they occur), frequency (how often they occur), and duration (how long they last). The occurrence of traffic flows can be cyclic (e.g., every Monday morning before 10 a.m., at the end of each month, or during the end-of-the-year closeout). Understanding when flows occur and how they impact your customer's business or operation is critical. For example, if 90% of the revenue for a business is generated during the last four weeks before Christmas, you may need to focus flow development on what is happening to applications during that time.

This leads to the concept of developing worst-case and typical-usage scenarios. This is similar to developing for single-tier or multi-tier performance, as discussed in Chapter 3. Worst-case scenarios are like the aforementioned one—for a business whose revenue is highly seasonal. This is similar to developing for the highest-performance applications and devices, in that you focus on times when the network requires its greatest performance to support the work being done. Architecting/designing a network for the worst-case scenario can be expensive and will result in over-engineering the network for most of its life cycle. Therefore, the customer must be fully aware of the reasons for, and supportive of, building toward a worst-case scenario.

Typical-usage scenarios describe an average (typical) workload for the network. This is similar to developing for single-tier performance, in that you focus on times when average, everyday work is being done. Recall from Chapter 3 that single-tier performance focuses on those applications and devices that are in general use and that make up a background level of traffic on the network. Often both worst-case and typical-usage scenarios are developed for a network.

Using data sources and sinks to help map flows for an application provides a great start to generating the flows for your network. Flow models, described next, are another great tool that you can use to help describe flows.

4.6 Flow Models

Another method to help describe flows in the network is to compare them to general, well-known flow models. *Flow models* are groups of flows that exhibit specific, consistent behavior characteristics. The flows within a flow model apply to a single application. Directionality, hierarchy, and diversity are the primary characteristics of flow models. *Directionality* describes the preference (of lack thereof) of a flow to have more requirements in one direction than another. Hierarchy and diversity are based on the definitions from Chapter 1.

While network architectures and designs typically treat traffic flows as having equal requirements in each direction, we find that many flows (especially from newer applications) have substantially different requirements in each direction. Most flows are asymmetric, and some access and transmission technologies (such as digital subscriber loop [xDSL] or WDM) can be optimized for such flows.

Flow models help describe the degrees of hierarchy and diversity of flows for applications. They show where flows combine, where they can be grouped together, and where flows occur between *peers*, which are devices at the same level in the hierarchy. They also can help us to identify which flows are critical flows; these, as you may recall, are considered more important than others, in that they are higher in performance, have strict requirements, or serve more important users, applications, and devices.

Flow models can also be useful to help quickly identify and categorize flows in an environment, so that we may easily recognize its flow characteristics. In this way they are like application groups, discussed in Chapter 2.

Flow models that we examine are:

- Peer-to-peer
- Client–server
- Hierarchical client–server
- Distributed computing

For each model we consider the directionality and hierarchy of its flows. We also, when possible, identify which flows in each model are critical, or important, flows.

These flow models are a subset of many possible models that you could use to characterize the flows for your network. We could include, for example, real-time (or near real-time or interactive) flows as a model, as well as streaming media. You are encouraged to develop an exhaustive list of flow models for your network projects.

4.6.1 Peer-to-Peer

Our first flow model, *peer-to-peer*, is one where the users and applications are fairly consistent in their flow behaviors throughout the network. They are, in effect, peers, in that they act at the same level in the hierarchy. Since they (users and/or applications) are fairly consistent, their flows are also fairly consistent. Thus, we can consider the flows in a peer-to-peer flow model to be equivalent (Figure 4.21). This has two important implications:

- We cannot distinguish between flows in this model. Therefore, either all of the flows or none of the flows is critical
- Since the flows are equivalent, they can be described by a single specification (e.g., profile)

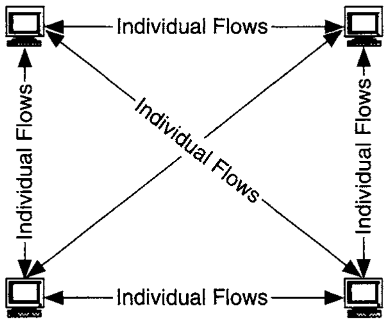


FIGURE 4.21 Peer-to-Peer Flow Model

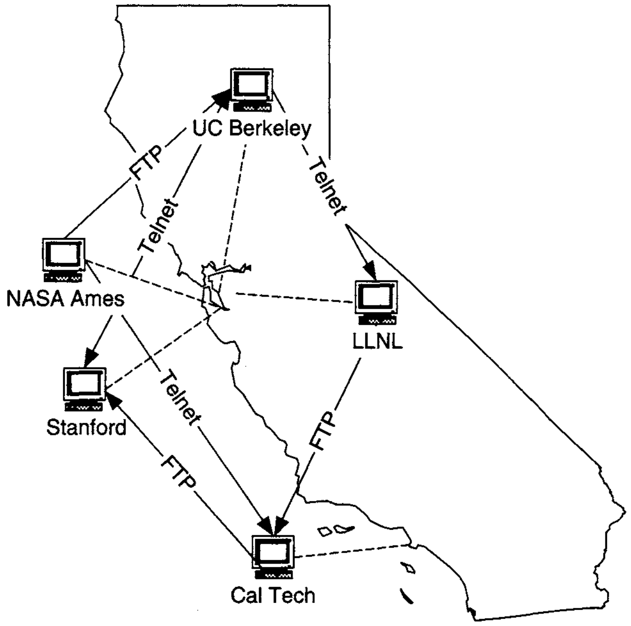


FIGURE 4.22 An Example of Peer-to-Peer Flows in the Early Internet

There are several examples of peer-to-peer flow behavior. The first is the early Internet, a portion of which is shown in Figure 4.22. In the early Internet, applications like FTP and telnet were predominant, and each device on the network was a potential source and destination of the flows for these applications. Another example is of file-sharing applications on the Internet. Basically, anywhere devices communicate directly with each other is considered peer-to-peer.

The peer-to-peer flow model is our default when we do not have any other information about the flows in our network. In a sense, it is part of the degenerate case for our requirements map (since there are no flow-specific requirements that can be determined). This flow model can also be used to describe flows when all users in a group need equal access to one another for an application. In addition to the file-sharing and remote access applications already described, this could also be a multimedia, tele*services application (e.g., teleseminars, telelearning, teleconferencing), where any of the participants may source or sink data to or from any other participant. Although each of the tele*services just noted can be considered a one-to-many application, there are components of each that can be applied as a set of one-to-one conversations. For example, while telelearning consists of a number of users (students) receiving and transmitting from and to a teacher, another

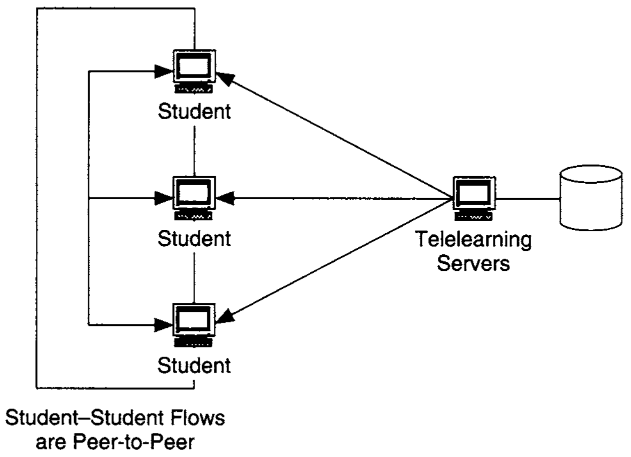


FIGURE 4.23 Peer-to-Peer Flows in a Telelearning Environment

component to this application is the ability to have side conversations among students. This part of the application is peer-to-peer (Figure 4.23). This flow model is useful, in that it indicates that a performance profile may be used for those flows, and that there are no (or several) critical flows for that application.

4.6.2 Client–Server

The client–server flow model is currently the most generally applicable model. This model has both directionality and hierarchy. Flows in this model are bidirectional, between clients and the server, in the form of requests and responses. This flow model is *client–server* in that the flows are asymmetric and hierarchically focused toward the client. Thus, requests tend to be small relative to responses. Depending on the type of application, the flows may be considered almost unidirectional, from the server to the clients. Figure 4.24 illustrates the client–server flow model.

Since the flows in the client–server model are asymmetric, with the predominant or important flows in the direction from the server to the clients, the server can be considered a data source, and the clients are data sinks. The server would be shown as a data source on the requirements map, with flows generating from it to its clients on other areas of the map. Since the predominant or important flows are from the server to the clients, these are the critical flows for this model. When there is a requirement to transmit information to multiple clients concurrently, multicasting at some layer in the network must be considered to optimize flows for this model.

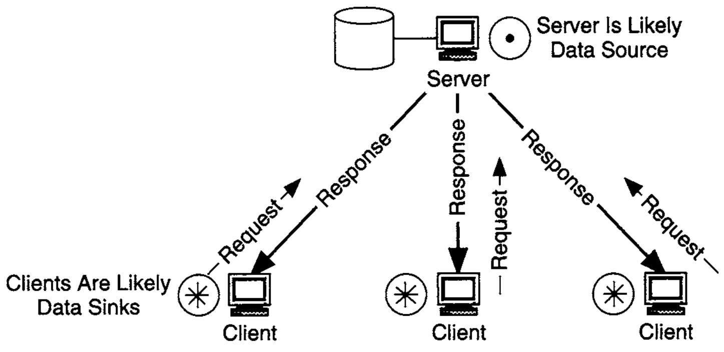


FIGURE 4.24 Client–Server Flow Model

This model is the traditional view of client–server operations, exemplified by ERP applications such as SAP, and e-commerce applications. Applications such as these are highly dependent on network performance when they are configured to work across significant distances, as when a company is dispersed across multiple locations, spanning cities, states, or countries. If the network does not properly support client–server applications, the customer must resort to distributing the client–server architecture (e.g., distributed ERP), which can be expensive.

Video editing exemplifies a client–server flow model. A video server can store video to be edited, and clients make requests to that server for video to edit. The server passes video to the clients, which may be sent back up to the server upon completion, or may be sent elsewhere for more processing (Figure 4.25).

Another view of client–server flows is with Web applications. While the early Internet started with peer-to-peer flows from applications such as FTP and telnet, this usage evolved to become more client–server-like with the use of FTP servers, followed by applications such as gopher and Archie. Now, with the widespread use of Web applications, many flows in the Internet are between Web servers and their clients. As TCP/IP assumes more network operating system (NOS) roles, print and file services across enterprise networks and the Internet will become more client–server oriented. For example, in the early days, a person who wanted to access information from an organization would have used the application FTP to a known site to download information. Then this changed into accessing an FTP or gopher server, and then to accessing a Web server. Today a person may access large quantities of information from an organization without ever entering that organization's network, through accessing external Web servers.

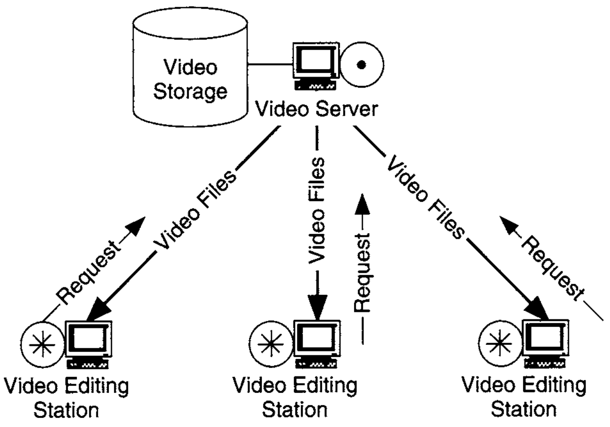


FIGURE 4.25 An Example of Client–Server Flows

However, a better flow model for Web traffic has multiple levels of tiers, with traffic flows within some of the tiers. This is the hierarchical client–server model, described in the following section.

4.6.3 Hierarchical Client–Server

As the flows within a client–server flow model become more hierarchical, in terms of adding layers, or tiers, to the flows, then their behavior can be represented as a hierarchical client–server flow model. A *hierarchical client–server flow model* has the characteristics of a client–server flow model but also has multiple layers, or tiers, between servers. In this model there may also be flows from a server to a support server or management device, as shown in Figure 4.26. These flows (server-to-server and server-to-manager) may be considered critical, in addition to the server-to-client flows. With the additional layer(s) of hierarchy in this model the servers can now be either data sources or sinks (or both). More information may be needed about the application(s) in order to determine the status of these servers. This model is important in that it recognizes server-to-server and server-to-manager flows.

A hierarchical client–server flow model is indicated when multiple applications work together and share information to accomplish a task, or when multiple client–server applications (or multiple sessions of the same client–server application) are managed by a higher-level application. An operations support system (OSS) managing several back-office applications may often be modeled in this fashion.

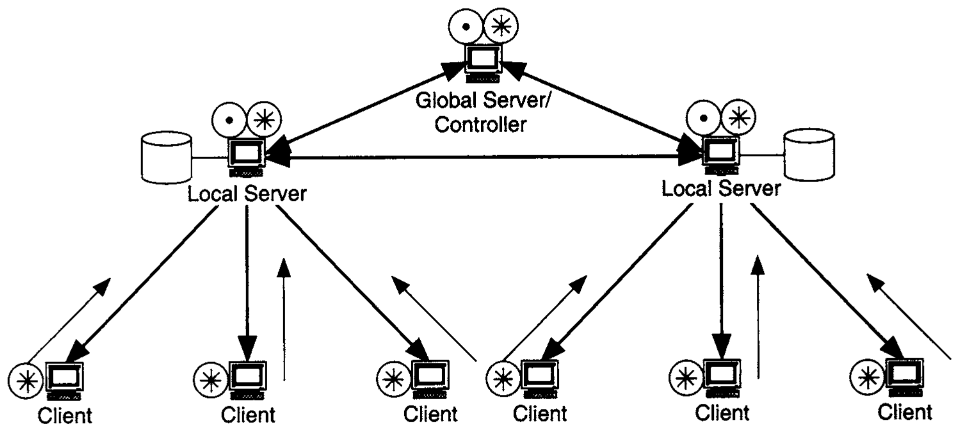


FIGURE 4.26 A Hierarchical Client–Server Flow Model

Critical flows for this model are dependent on application behavior. If the applications are inherently client–server, and the servers are there merely to support multiple sessions, then the client–server flows may be the only critical flows. However, when the servers communicate with each other (e.g., to update a common database or share data between applications), then the server-to-server flows may be critical, possibly in addition to the client–server flows. And when there is communication to a manager (e.g., to synchronize processing or information), then the server-to-manager flows may also be critical.

We have shown that the Internet has evolved from an early peer-to-peer flow model to a client–server flow model with the acceptance of Web applications. As client–server traffic has grown, however, Web servers have been replicated and distributed across the Internet, resulting in an increase in server-to-server flows. This is being done to increase the effectiveness of Web access, in part by spreading access across multiple devices and in part by locating devices closer to the access portion of the Internet, thus bypassing part or all of the core. As a result, the Internet is evolving to more of a hierarchical client–server flow model.

Such hierarchical Web services are shown in Figure 4.27. In this figure, content delivery networks (CDN) and mirrors (introduced in Chapter 1) are used to migrate Web content between servers and provide local access to content for users.

In this figure, the servers can provide the same function or different functions. For example, Web-based three-tier application servers can run application and Web services on the same device, while running database services on another device. In this case the flows between servers (application/Web and database) are critical for operation.

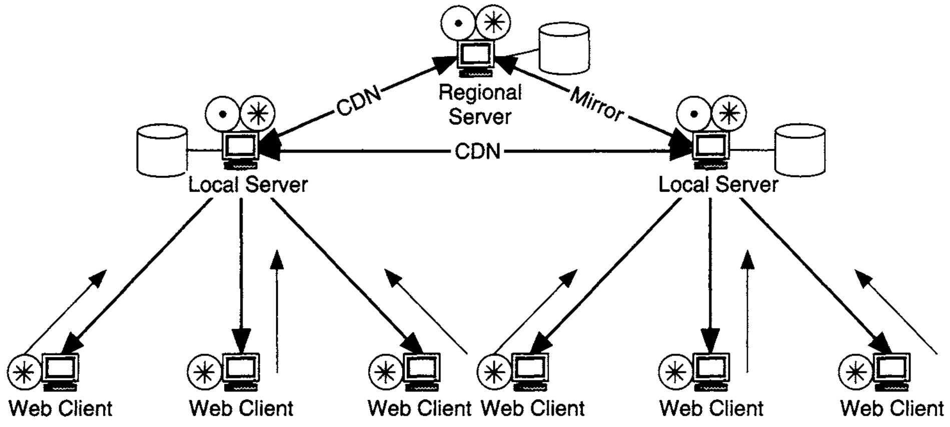


FIGURE 4.27 Web Services Modeled Using a Hierarchical Client-Server Flow Model

This type of flow model can also be seen with the visualization application group described in Chapter 2. An example of this is in the visualization of scientific simulations. Consider the simulations of a multi-part problem. These can be found in climate modeling, fluid flow analysis, structural analysis, and others.

In climate modeling there may be a simulation consisting of multiple parts—atmosphere, earth, and ocean—as shown in Figure 4.28. Each part of the simulation in this figure may be developed on a separate computing device, probably at different locations (based on where the various scientists are located). Since each component affects the others, at the boundaries between atmosphere, earth, and ocean, data must be passed between the computing/visualization servers for each part. The flows would look like those in Figure 4.29. In this figure, if the parts of

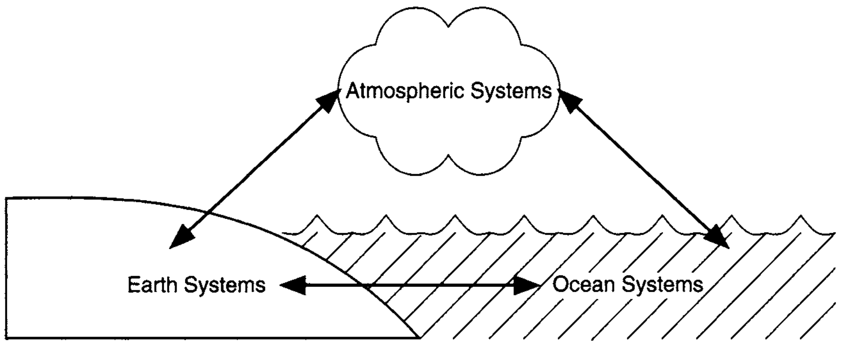


FIGURE 4.28 Components of a Climate Modeling Problem

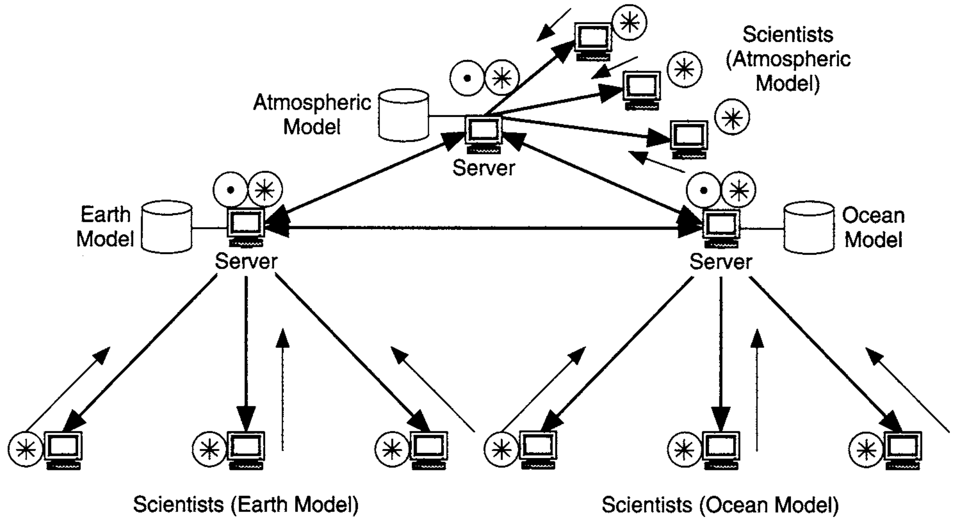


FIGURE 4.29 A Hierarchical Client–Server Model for Scientific Visualization

the simulation are being solved at different locations, then the server-to-server flows may cross long (WAN) distances, impacting both local- and wide-area networks.

4.6.4 Distributed-Computing

The distributed-computing flow model, shown in Figure 4.30, is the most specialized of the flow models. A distributed-computing flow model can have the inverse of the characteristics of the client–server flow model, or a hybrid of peer-to-peer and client–server flow models. In this model, flows may be primarily between a task manager and its computing devices (like a client–server model) or between the computing devices (like a peer-to-peer model). The type of model depends on how the distributed computing is done. The important characteristics of this model are that the flows can be client–server but are reversed in direction, and that the computing devices may have strict performance requirements.

We can make distinctions in the distributed-computing flow model based on the relationship between the task manager and the computing devices and what the task is. This relationship can result in the computing devices being closely coupled, where there are frequent transfers of information between devices, or loosely coupled, where there may be little to no transfer of information between computing devices. Tasks may range from having a coarse granularity, where each task is dedicated to a single computing device, to having a fine granularity, where a task is subdivided among several devices and the computing is done concurrently.

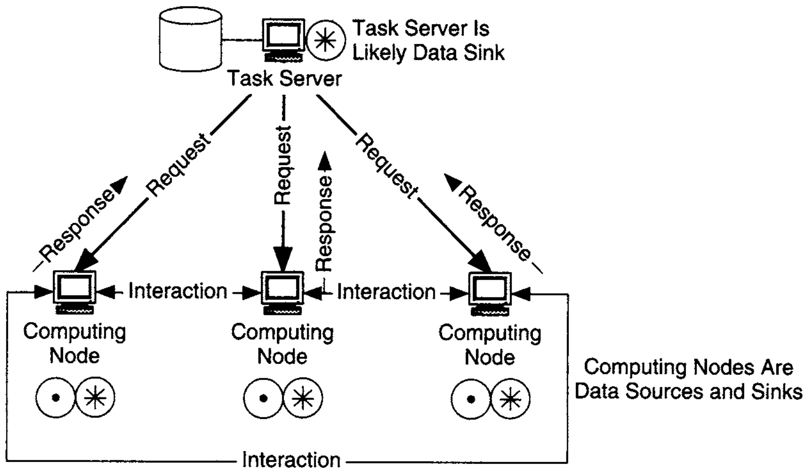


FIGURE 4.30 A Distributed-Computing Flow Model

When the task has a coarse granularity and the computing device relationship is loosely coupled, then the distributed-computing flow model takes the form of a computing cluster or computing resource management system, where tasks are allocated to each computing device based on resource availability. Thus, each computing device communicates with the cluster server or resource manager. Figure 4.31 shows the flows for an example of a computing cluster.

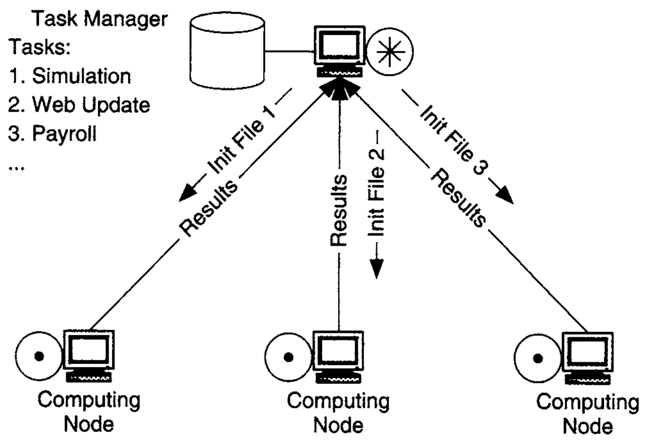


FIGURE 4.31 Flows for a Computing Cluster

The flows in this type of distributed-computing flow model are similar to those in the client–server flow model, where communications are primarily between each client and the server. A difference here is that the direction of the flows is not necessarily from the computing server to its clients. In fact, the size of the task initialization file (which is, in a sense, a request) sent from the server to each computing device may be much smaller than the size of the results of the computation, which is sent from the computing device to the server. In this model the flow directionality is asymmetric, but in the opposite direction from the client–server flow model. Also, each of the flows between the computing devices and their server is independent of the other flows. There is no synchronization among individual flows. The critical flows for this model are from the computing devices to their server. Since the flows for this model are asymmetric, in the direction toward the server, the server acts as a data sink, while the computing devices act as data sources.

When the task has a fine granularity and the computing node relationship is closely coupled, then the distributed-computing flow model behaves like a simplified parallel processing system, where each task is subdivided, based on the degree of parallelism in the application and the topology of the problem, among several computing devices. These devices work concurrently on the problem, exchanging information with neighbor devices and expecting (and waiting for) updated information. The task manager sets up the computing devices and starts the task with an initialization file, as in Figure 4.32.

Flows in this type of distributed-computing flow model can have the most stringent performance requirements of any of the models. Since computing devices may block (halt their computations) while waiting for information from neighbor devices, the timing of information transfer between computing devices becomes critical. This has a direct impact on the delay and delay variation requirements for the network connecting the devices. Although each individual flow has directionality, collectively there is little or no overall directionality. Individual flows in this model can be grouped to indicate which neighbor devices a computing device will communicate with for a given problem or topology. For example, a problem may be configured such that a computing device will communicate with one, two, four, or six of its closest neighbors.

For this model, critical flows are between computing devices. When a device will transfer the same information to several neighbors simultaneously, multicasting should be considered to optimize flow performance. There are no clear data sources or sinks for this model. The climate-modeling problem shown in Figure 4.28 could

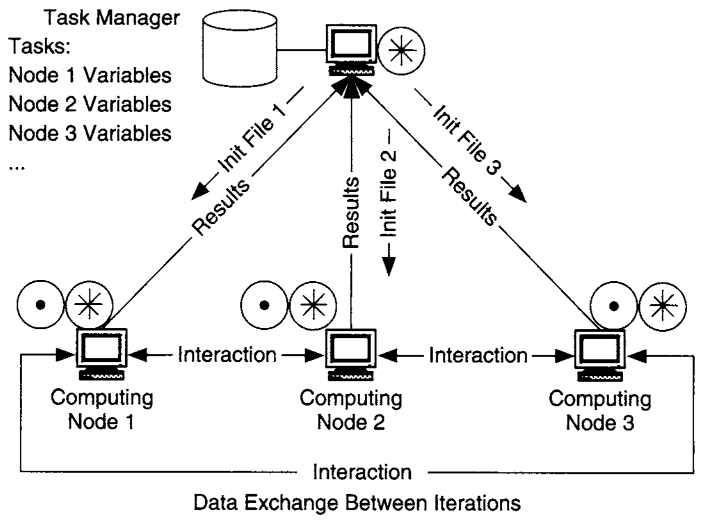


FIGURE 4.32 Flows for Parallel Computing

also be considered with a distributed-computing flow model, depending on the task granularity and degree of coupling within the system.

Flow requirements will vary between the computing cluster and parallel system models, depending on the degrees of coupling and the granularity in the task. Depending on the application and amount of analysis you want to put into this model, you can use the computing cluster and parallel system models as they are, or modify the task granularity and degree of coupling to suit your needs.

4.7 Flow Prioritization

In developing and describing flows for your network, you may find it useful to prioritize flows. Flow prioritization means ranking flows based on their importance, which can be described in various ways, depending on your environment. Flows can be prioritized according to importance, based on the characteristics shown in Figure 4.2. Some common prioritizations include:

- Business objectives and the impact of a flow on the customer's business
- Political objectives
- One or more of the performance requirements of the flow (a subset of capacity, delay, RMA, and quality of service).

- Security requirements for each flow
- The numbers of users, applications, and/or devices that a flow serves

The purpose for prioritizing flows is to determine which flows get the most resources or which flows get resources first. Typically, the primary resource is funding. This is an entirely new way of looking at how to allocate funding for parts of the network. By basing resource allocations on flows, which are directly representative of the users, their applications, and their devices, the resulting network architecture and design more accurately represent user, application, and device requirements.

You may use more than one parameter for prioritization, creating multiple levels of priority. It is common to start by prioritizing based on the number of users that a flow supports, and then adding another parameter, such as a performance requirement (e.g., capacity). Some example prioritizations are presented in Figures 4.33 and 4.34.

In Figure 4.34 the flows are prioritized by the number of users the flow supports. This information would have been determined from the requirements specification and requirements map, by comparing user, application, and device requirements, and mapping them to common flows. For this network there is a funding level ($750 K), which is distributed to each flow based on the number of users supported.

Another example prioritizes flows based on performance requirements, in this case reliability. Figure 4.35 shows a set of flows that have been prioritized this way. In this example there are three levels of reliability: 99.95%, 99.5%, and N/A

Flow ID	Performance Requirements			Number of Users
	Reliability	Capacity	Delay	
F1	N/A	1.2 Mb/s	10 ms	1200
F2	99.5%	100 Kb/s	N/A	550
F3	99.5%	15 Kb/s	100 ms	100
CF1	99.95%	500 Kb/s	100 ms	1750
CF2	N/A	100 Kb/s	100 ms	2100
CF3	N/A	3 Mb/s	100 ms	50

Total Budget for Network Project: $750 K

FIGURE 4.33 An Example of Flow Information for Prioritization

Flow ID	Performance Requirements			Number of Users	Budget	Priority
	Reliability	Capacity	Delay			
CF2	N/A	100 Kb/s	100 ms	2100	$274 K	1
CF1	99.95%	500 Kb/s	100 ms	1750	$228 K	2
F1	N/A	1.2 Mb/s	10 ms	1200	$157 K	3
F2	99.5%	100 Kb/s	N/A	550	$72 K	4
F3	99.5%	15 Kb/s	100 ms	100	$13 K	5
CF3	N/A	3 Mb/s	100 ms	50	$6 K	6

Total Budget for Network Project: $750 K

FIGURE 4.34 Flows Prioritized by the Number of Users Served

Flow ID	Performance Requirements			Number of Users	Budget	Priority
	Reliability	Capacity	Delay			
CF1	99.95%	500 Kb/s	100 ms	1750	$375 K	1
F2	99.5%	100 Kb/s	N/A	550	$141 K	2
F3	99.5%	15 Kb/s	100 ms	100	$141 K	2
F1	N/A	1.2 Mb/s	10 ms	1200	$31 K	3
CF2	N/A	100 Kb/s	100 ms	2100	$31 K	3
CF3	N/A	3 Mb/s	100 ms	50	$31 K	3

Total Budget for Network Project: $750 K

FIGURE 4.35 Flows Prioritized by Reliability

(not applicable). Budget allocations for each level are: highest level, 1/2 of budget; middle level, 3/8 of budget; and lowest level, 1/8 of budget.

Funding can be applied to this list of flows in a variety of ways. It can be applied based on the level of reliability, in which case all flows with equal levels of reliability get equal amounts of funding. This allocation can then be refined by including another parameter, such as the users or applications supported.

How per-flow funding relates to purchasing equipment is discussed in the network design chapter of this book (Chapter 10).

4.8 The Flow Specification

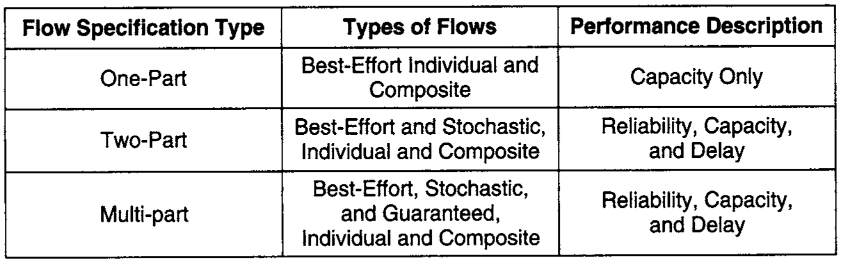
The results of identifying, defining, and describing flows are combined into a flow specification, or flowspec. A flow specification lists the flows for a network, along with their performance requirements and priority levels (if any). Flow specifications describe flows with best-effort, predictable, and guaranteed requirements, including mission-critical, rate-critical, real-time, interactive, and low and high performance. The flow specification combines performance requirements for composite flows, when there are multiple applications requirements within the flow. It can also be used to combine requirements for all flows in a section of a path. There is much information embedded within a flow specification.

Flow specifications can take one of three types: one-part, or unitary; two-part; or multi-part. Each type of flowspec has a different level of detail, based on whether the flows have best-effort, predictable, and/or guaranteed requirements.

A *one-part flowspec* describes flows that have only best-effort requirements. A *two-part flowspec* describes flows that have predictable requirements and may include flows that have best-effort requirements. A *multi-part flowspec* describes flows that have guaranteed requirements and may include flows that have predictable and/or best-effort requirements (Figure 4.36).

These flow specifications range in complexity. One-part and two-part flowspecs can be relatively straightforward, whereas multi-part flowspecs can be quite complex. Two-part flowspecs are usually a good balance between ease of development and amount of detail. Many networks can be adequately represented with a one-part flowspec, when performance requirements and flows are not well understood.

As networks become integrated into the rest of the system, however, flows will incorporate more reliability and delay requirements, and the two-part and multi-part flowspecs can better represent the network. This is the case today for many networks. In developing the flowspec we use the information in the requirements specification and requirements map as the basis for flows, and apply the methods described in this chapter to identify and describe flows.



Flow Specification Type	Types of Flows	Performance Description
One-Part	Best-Effort Individual and Composite	Capacity Only
Two-Part	Best-Effort and Stochastic, Individual and Composite	Reliability, Capacity, and Delay
Multi-part	Best-Effort, Stochastic, and Guaranteed, Individual and Composite	Reliability, Capacity, and Delay

FIGURE 4.36 Descriptions of Flow Specifications

4.8.1 Flowspec Algorithm

Flowspecs are used to combine performance requirements of multiple applications for a composite flow or multiple flows in a section of a path. The *flowspec algorithm* is a mechanism to combine these performance requirements (capacity, delay, and RMA) for flows in such a way as to describe the optimal composite performance for that flow or group of flows.

The flowspec algorithm applies the following rules:

1. Best-effort flows consist only of capacity requirements; therefore, only capacities are used in best-effort calculations.
2. For flows with predictable requirements we use all available performance requirements (capacity, delay, and RMA) in the calculations. Performance requirements are combined for each characteristic so as to maximize the overall performance of each flow.
3. For flows with guaranteed requirements we list each individual requirement (as an individual flow), not combining them with other requirements.

The first condition is based on the nature of best-effort traffic—that it is unpredictable and unreliable. RMA and delay requirements cannot be supported in a best-effort environment. The best that can be expected is that capacity requirements may be supported through capacity planning (also known as traffic engineering) or by over-engineering the capacity of the network to support these requirements.

The second condition is at the heart of the flowspec—that for each performance characteristic, capacity, delay, and RMA, the requirements are combined to maximize the performance of the flow or group of flows. How the requirements are combined to maximize performance is discussed later in this chapter.

The third condition is based on the nature of guaranteed requirements. Since flows with such requirements must be supported end-to-end, their requirements are kept separate so that we can identify them in the network architecture and design.

When a one-part flowspec is developed (for flows with best-effort requirements), then capacities of the flows are combined. There should be no RMA or delay requirements for these flows. Capacities are added together, forming a total best-effort capacity (C_{BE}), as shown in Figure 4.37.

A two-part flowspec builds on a one-part flowspec, adding predictable capacities, delay, and RMA. When a two-part flowspec is developed (for flows with best-effort and predictable requirements), the best-effort flows are combined in

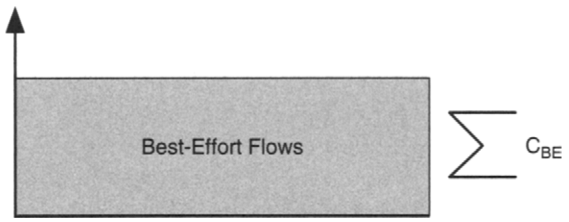


FIGURE 4.37 A One-Part Flow Specification

the same way as for the one-part flowspec. For the predictable requirements, the capacities are added together as with the best-effort flows, so that the flowspec has a total capacity for best-effort flows (C_{BE}) and another capacity for predictable flows (C_P). For the delay and RMA requirements for predictable flows, the goal is to maximize each requirement. For delay, the minimum delay (i.e., the highest-performance delay) of all of the delay requirements is taken as the predictable delay (D_P) for the flowspec, and the maximum RMA (i.e., the highest-performance RMA) of all of the RMA requirements is taken as the predictable RMA (R_P) for the flowspec. Figure 4.38 illustrates this for a two-part flowspec.

A multi-part flowspec is the most complex of the flowspecs, building on a two-part flowspec to add guaranteed requirements. Best-effort capacity, along with predictable capacity, delay, and RMA, is generated in the same fashion as for a two-part flowspec, and each set (i) of guaranteed performance requirements is added individually (shown as C_i, R_i, D_i) to the flowspec, as shown in Figure 4.39.

Sets of guaranteed performance requirements (C_i, R_i, D_i) are listed individually in the flowspec to show that they will be supported as individual requirements and not grouped with other requirements. This is necessary for us to be able to fully support each guaranteed requirement throughout the network.

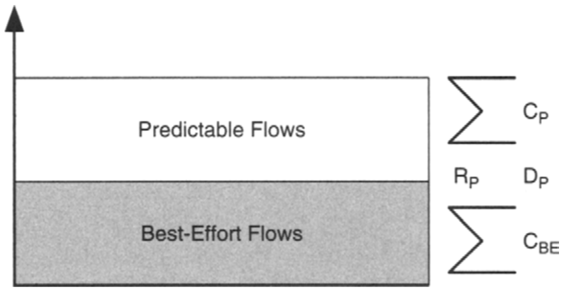


FIGURE 4.38 A Two-Part Flow Specification

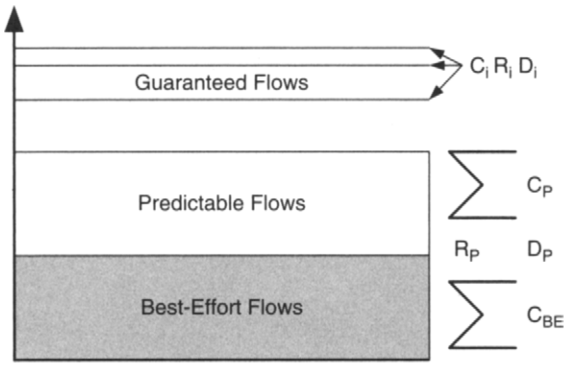


FIGURE 4.39 A Multi-Part Flow Specification

4.8.2 Capacity and Service Planning

Capacity and service plans are written descriptions of the network performance required for the flows described in the flowspec. A *capacity plan* describes network performance in terms of capacity only. It is used in conjunction with a one-part flowspec. A *service plan* describes network performance in terms of sets of capacity, delay, and RMA. It is used in conjunction with two-part and multi-part flowspecs.

While a flowspec lists flows and combines their performance requirements, capacity and service plans describe what may be needed in order to support such requirements. As we see in the chapter on performance (Chapter 8), there are many mechanisms we may choose to support performance requirements in the network.

4.9 Example Application of Flow Analysis

We now bring the concepts of flow analysis together for an example network, in this case a network to support computing and storage management. For this network project the computing and storage devices already exist in multiple buildings on a campus. The buildings and devices that will be using this network for computing and storage management are shown in Figure 4.40.

From the requirements analysis process, we have been able to determine that there are four types of flows for this network:

Type 1: Flows between high-performance computing devices. There are compute servers that are the high-performance devices for this network. The first type of flow consists of traffic flows between these devices. Flows are sometimes (approximately

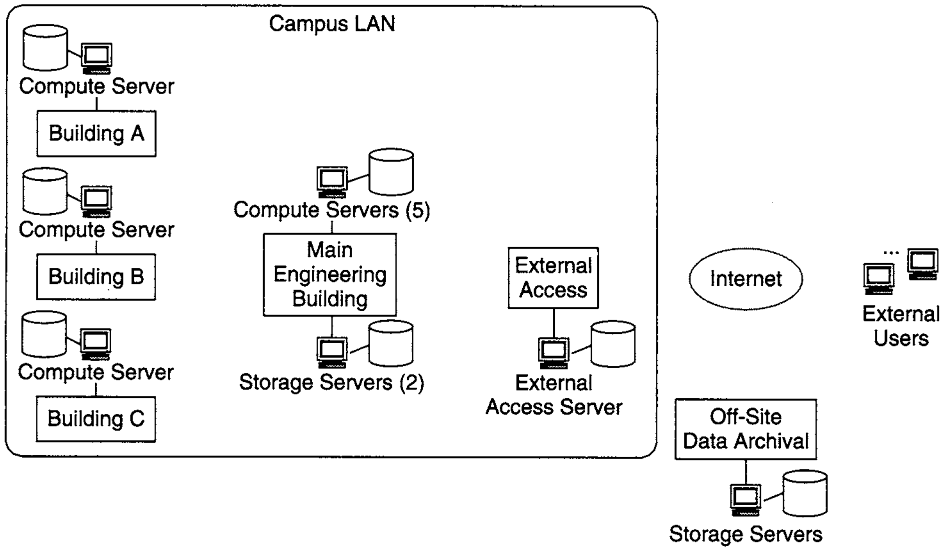


FIGURE 4.40 Building and Device Locations for the Example

10% of the time) synchronized between pairs of devices. At other times these devices may draw data from the local data store of another computing device, from the storage server at the Main Engineering building, or request a computing or data management task from a device. Most computing tasks are controlled from Main Engineering.

Type 2: Data migration from computing to storage at Main Engineering. These flows may be from each compute server as its task is completed; from the local data store at each compute server at a specific time; or from the compute server at Main Engineering at the completion of a combined (multi-server) task.

Type 3: Data migration to external server. Final data sets, or extracts of final data sets, are pushed from the Main Engineering storage server to a storage server at a building where external (Internet) access is managed. Data at this external access server are used to feed other campuses and for remote (off-site) data archival.

Type 4: Data archival and Internet access. These are flows from the external access server to users on the Internet, as well as flows to off-site archival servers.

These flow types are added to our map, as shown in Figure 4.41.

From discussions with various users of these applications, we learned that flow usually runs in this order: type 1–type 2–type 3–type 4. For flow type 1, a Main Engineering compute server may act as a server for the compute servers in Buildings A–C and for getting data from the storage servers in Main Engineering. This follows a hierarchical client–server flow model. Compute servers from Buildings

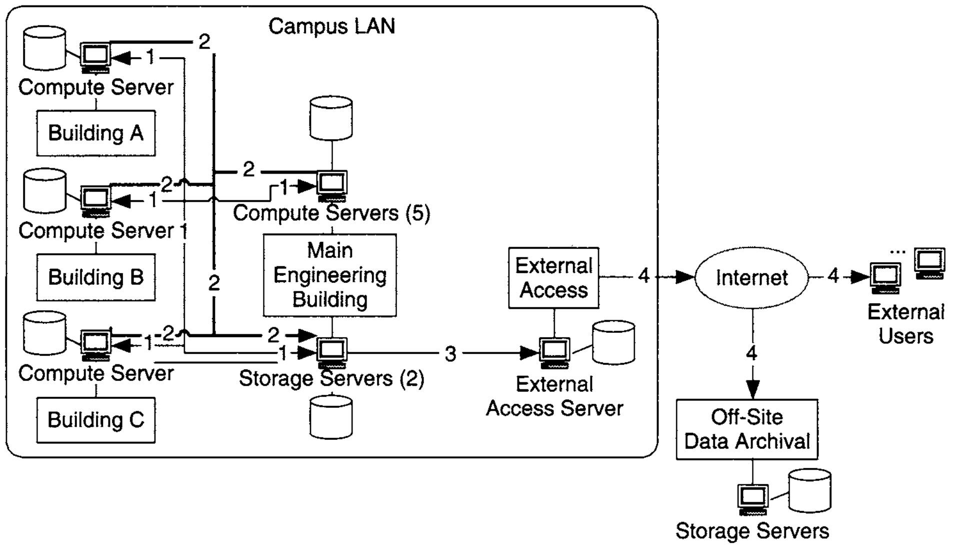


FIGURE 4.41 The Map with Flow Types Added

A–C and Main Engineering may also act as synchronized peers, using a distributed-computing flow model.

From discussions with engineering users we found that the computing application runs in either batch or interactive mode, from about 10 minutes to several hours, generating files of sizes ranging from 1 to 2 MB (synchronization or sync files), 10 to 100 MB (interactive updates), and 100 MB to over 1 GB for final data sets.

Interactivity for the computing application is needed to steer or direct the computation. This requires synchronization on the order of HRT (100 ms), and updates on the order of 1 second. Users expect to have up to two tasks running concurrently.

For flow type 2, data are stored at the storage servers in Main Engineering. Data can be from interactive updates, final data sets, and extracts (selected subsets of the final data set). Data also migrate from local stores at each computer server, usually every few hours.

For flow type 3, full data sets as well as extracts of the data sets are migrated to the external access server. Extracts are approximately 80% of the size of a full data set. Data sets are migrated hourly.

For flow type 4, users from other campuses, via the Internet, access data. Data sets are archived at an off-site facility. The system is expected to support the download of a full final data set within a few minutes.

Performance Envelope from Requirements Analysis

Characteristics of flow type 1. Flows of type 1 involve the frequent passing of 1–2 MB sync files with delays on the order of HRT, 10–100 MB update files on the order of 1 second, and final data sets of 500 MB–1 GB on the order of minutes to hours, with up to two tasks running concurrently. From this information we can estimate a range for capacity performance for these flows. Each of these flows is multiplied by 2 for concurrency.

Sync files: (1 to 2 MB)(8 b/B)(2 concurrent tasks)/10^{-1} s = 160 to 320 Mb/s

Update files: (10 to 100 MB)(8 b/B)(2 concurrent tasks)/1 s = 160 Mb/s to 1.6 Gb/s

Final data sets: (500 to 1000 MB)(8 b/B)(2 concurrent tasks)/(10^2 to 10^4 s) = 800 Kb/s to 160 Mb/s

Characteristics of flow type 2. These flows involve migrating (pushing) updates, final data sets, and extracts. The delay characteristics of these flows are much less strict than for the computing function, with delays ranging from 10 to 10^4 seconds.

Update files: (10 to 100 MB)(8 b/B)/(10 to 10^4 s) = 8 Kb/s to 80 Mb/s

Final data sets: Same as for flow type 1

The performance envelope for final data sets, updates, and synchronization files is shown in Figure 4.42.

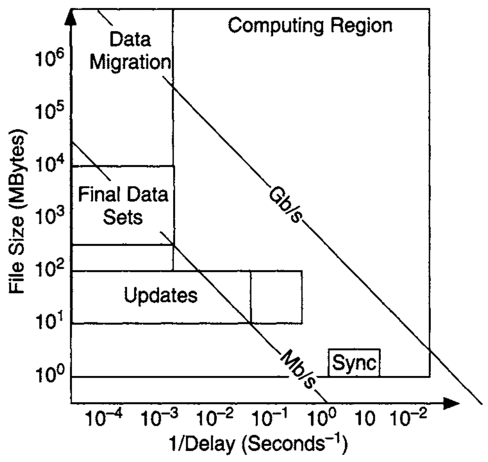


FIGURE 4.42 The Performance Envelope for the Example

Flow Models

For flow type 1 between compute servers and the Main Engineering storage servers, flows can follow distributed-computing and hierarchical client–server computing flow models, as shown in Figure 4.43.

In the distributed-computing model each device can act as a data source and sink, and data transfer is synchronized between devices at about 100 ms. In the hierarchical client–server model, data sets can flow from the storage server to the compute servers in Main Engineering, which then pass down to compute servers in Buildings A–C. There is no synchronization for this model.

Flow type 2 consists of data pushes from each compute server to the storage server in Main Engineering. Each compute server is a data source, while the Main Engineering storage server is a data sink (Figure 4.44).

For flow type 3, the storage servers in Main Engineering are data sources, and the external access server is a data sink (Figure 4.45).

For flow type 4 a client–server flow model exists between external users of the data, including off-site archival, and the external access server (Figure 4.46).

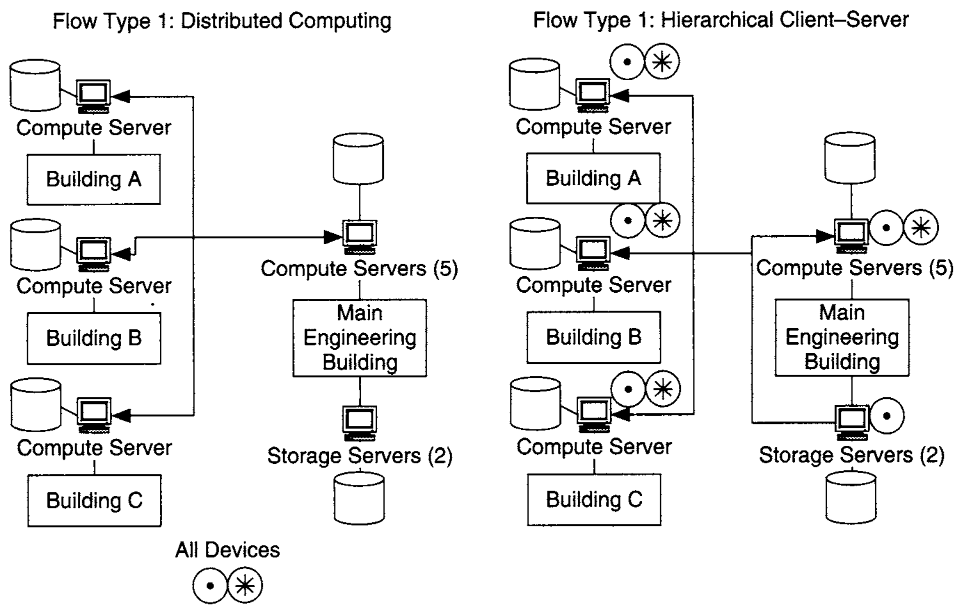


FIGURE 4.43 Flow Models for Flow Type 1

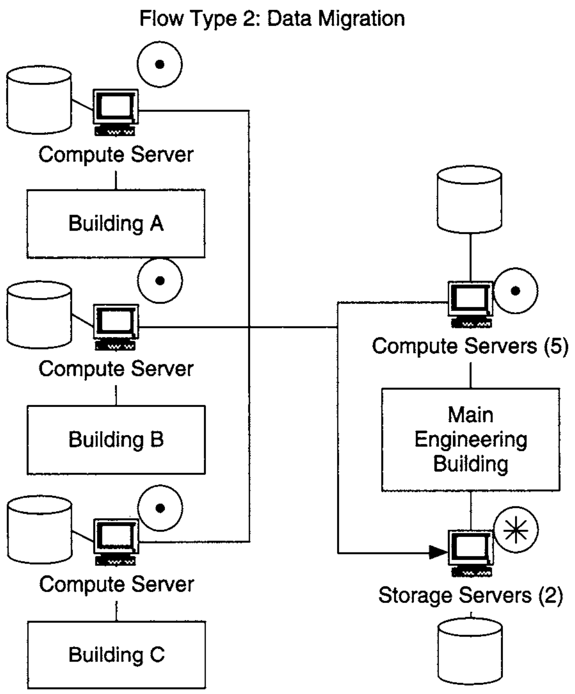


FIGURE 4.44 Flow Model for Flow Type 2

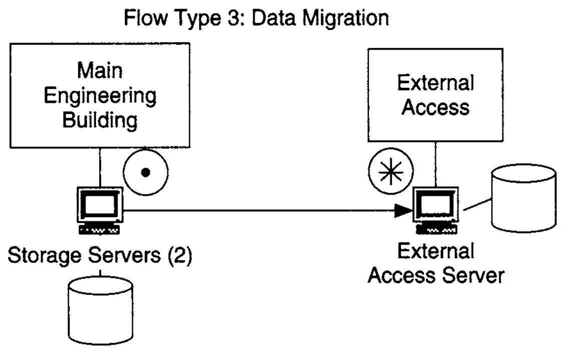


FIGURE 4.45 Flow Model for Flow Type 3

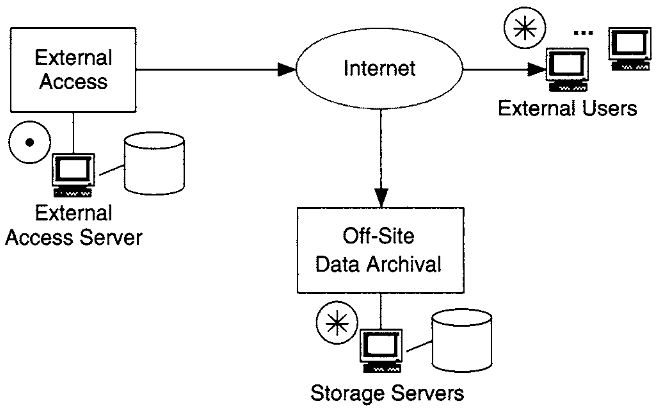


FIGURE 4.46 Flow Model for Flow Type 4

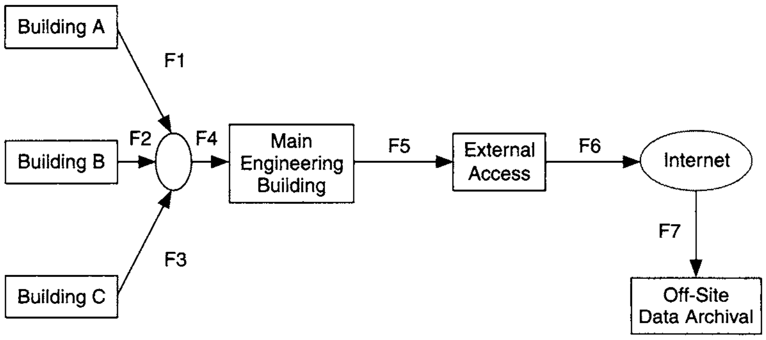


FIGURE 4.47 A Flow Map for the Example

Flow Map

Figure 4.47 is an example of a flow map that describes flows between buildings. Note that all devices have been removed from this map. This is often done for larger networks, as the number of devices on a map can become unwieldy. Also, a flow aggregation point has been added between Buildings A–C and Main Engineering. This is done to show the aggregated performance requirements at Main Engineering (flow F4).

For this flow map, flows F1, F2, and F3 have the same performance requirements, consisting of flow types 1 and 2. Flow F4 is an aggregate of flows F1, F2, and F3 (flow types 1 and 2). Flow F5 consists of flow type 3, and flows F6 and F7 are flows of flow type 4.

Next we combine the performance requirements from each of the flow types and apply them to flows F1 through F7, as shown in Figure 4.48.

Flow ID	Performance Requirements	
	Capacity (Mb/s)	**Delay (ms)**
F1: Flow Type 1		
Synchronization Files	320	100
Update Files	1600	1000
Final Files	160	10^5
Result for Flow Type 1	1600	100
F1: Flow Type 2		
Update Files	80	10^4
Final Files	160	10^5
Result for Flow Type 2	160	10^4
Result for F1	**1760**	**100**
Result for F2	**1760**	**100**
Result for F3	**1760**	**100**
F4: Flow Type 1	1600	100
F4: Flow Type 2		
Update Files	320	10^4
Final Files	640	10^5
Result for Flow Type 2	640	10^4
Result for F4	**2240**	**100**
Result for F5	**16**	**10^3**
Result for F6	**80**	**10^2**
Result for F7	**16**	**10^3**

FIGURE 4.48 Performance Requirements for Flows

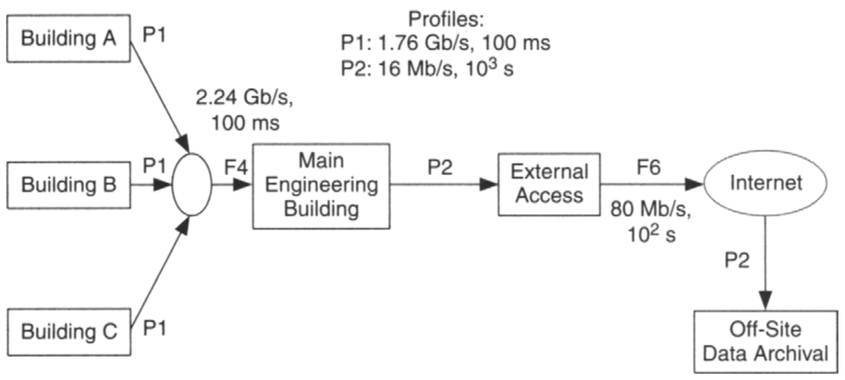


FIGURE 4.49 Performance Requirements Added to the Flow Map

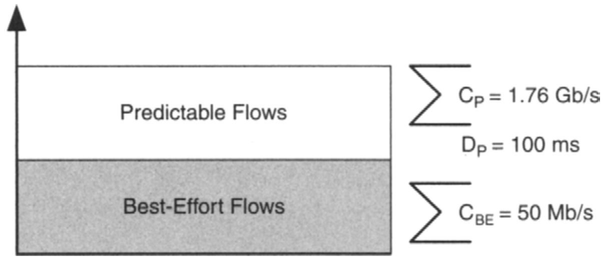


FIGURE 4.50 Two-Part Flowspec for Each Flow with Performance Profile P1

When these performance requirements are added to the flow map from Figure 4.47, we get Figure 4.49. Two performance profiles were generated for multiple flows: P1 for flows F1, F2, and F3; and P2 for flows F5 and F7.

A two-part flowspec for each flow that has a performance profile of P1 (F1, F2, and F3) would look like Figure 4.50.

4.10 Conclusions

Flow analysis takes an end-to-end perspective of network performance requirements, combining capacity, delay, and RMA requirements into a specification that is used as input to the network architecture and design, to evaluate and help select technologies and diversity strategies for the network. In building the flow specification we use various techniques, including data sources and sinks and flow models, to identify and determine individual and composite flows as well as critical flows.

Flow analysis is the final part of the analysis process. We began this process by gathering, deriving, managing, and tracking requirements for the network, from users, applications, devices, and networks that will be part of the planned network. In developing requirements for the network, we considered performance requirements (in terms of capacity, delay, and RMA) and the many ways to categorize requirements for users, applications, devices, and networks. This information, along with initial conditions, problem definitions, and goals, was collected in the requirements specification and mapped out in a requirements map.

Performance requirements, on a per-application basis or grouped by user, application, device, or network, are added to the directionality, hierarchy, and diversity of traffic flows to characterize them. Some tools, such as data sources and sinks, flow models, and flow aggregation points, can be used to help us determine which flows are important in a network and where flows are likely to occur. You are encouraged to develop other tools to aid in analyzing flows, or modify those presented in this book to fit your needs.

While flow analysis is presented here as part of the overall analysis process, in preparation to architect and design a network, it should be noted that flow analysis can be performed on any network, regardless of what state it is in. Notice that throughout the flow analysis process, no network technologies, topologies, or underlying infrastructures were shown or mentioned. Flow analysis allows us to separate traffic movement and performance requirements from an existing network, giving us the freedom to determine what flows should look like when the network does not restrict movement or performance. If you analyze flows on an existing network (regardless of whether or not you are developing a new network or upgrading the existing network), the results of this analysis will indicate if the existing network needs to be modified to fit the traffic flows.

Now that we have an idea of what to expect of the network in terms of requirements and flows, we are prepared to begin the process of network architecture.

4.11 Exercises

1. **Show flows for each set of devices and applications below. Label each as either a unidirectional or bidirectional flow.**
   a. **Client–server application: Downstream (from server to client): 1.2 Mb/s capacity; upstream (from client to server): 15 Kb/s capacity.**
   b. **Streaming video (UDP) from video server to a subscriber's PC: 300 Kb/s capacity, 40 ms delay (one-way).**

  c. Downloading pages from the Web: Downstream: 250 Kb/s capacity, 5 second delay; upstream: 100 Kb/s capacity.
  d. Transaction processing from point-of-sale machine to server: Upstream (from PoS machine to server): 30 Kb/s capacity, 100 ms round-trip delay; downstream: 50 Kb/s capacity.

2. Devices can act as both data sources and sinks, depending on the application and flow. Which of the following devices (for the applications given) are data sinks? Data sources?
   a. A storage device receiving streaming video from a camera
   b. A video editing unit, using video from the storage device in (a)
   c. A Web server and its clients
   d. A storage disk farm

3. Which flow models apply to each set of flows described below?
   a. Users on the Internet accessing the same Web server
   b. Forty workstations processing batch jobs overnight, managed by a central mainframe
   c. Email use across the Internet
   d. A transaction-processing application, authorizing credit card transactions between a company's retail stores and its headquarters

4. For each of the examples in Exercise 3, give the most likely direction(s) for the flows described by each flow model.

5. Develop a flow model for real-time/near-real-time flows. How would you characterize the flows for this model? What are likely data sources and sinks? Apply your model to a videoconferencing application.

6. You are developing a network for a company's online transaction processing (OLTP) application (e.g., a retail sales network). Its current system is a mainframe that has several terminals connected to it, either directly or through a terminal server, as in Figure 4.51. It is moving to a hierarchical client–server network, where there will be

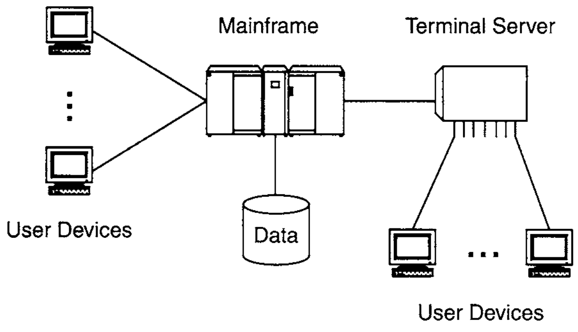


FIGURE 4.51 A Mainframe Environment for an OLTP Application

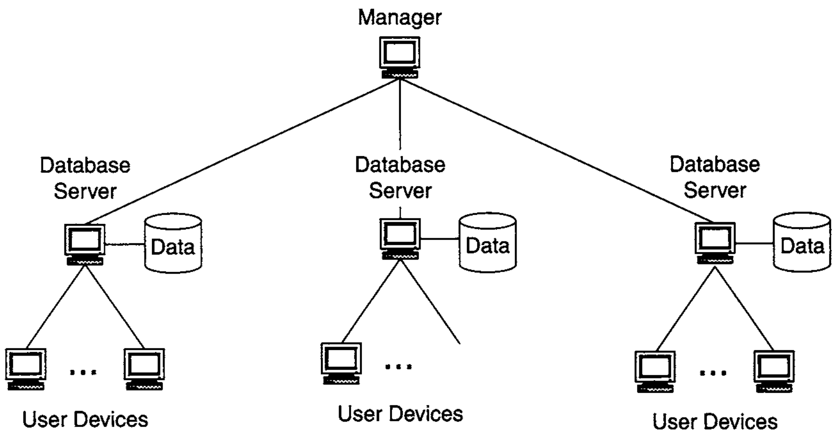


FIGURE 4.52 A Hierarchical Client–Server Environment for an OLTP Application

multiple regional database servers, each acting in a client–server fashion and updating each other's regions via a database manager, as in Figure 4.52.

a. Show the probable data sources and sinks for both environments.
b. How does migrating from the mainframe environment to the hierarchical client–server environment modify the traffic flows in the network?
c. In what ways does the network environment improve the traffic flows?
d. What are some of the potential trade-offs between the two environments—for example, in security, management, and performance?