Designing manageable protocols

Introduction

Every computer network has a purpose: business networks to support business, education networks to support education, government networks to support government and backbones to carry IP packets. With the possible exception of backbones, which may be expected to treat every packet alike, each of these purposes implies that some uses of the network are more important than others and therefore that when network or other resources are in short supply these uses should have precedence.

The job of a network manager is to ensure that their network delivers the best possible service to the organisation, in support of its intended purpose. This means supporting those uses of the network that deliver the intended purpose, if necessary reducing or even excluding other uses. To manage a network effectively therefore requires:

- a policy that clearly sets out what the network is to be used for, defining local and remote services that must be available from different network locations; the policy should set priorities between these services and may also prohibit some services in some or all parts of the network;
- a requirement that users behave in accordance with the policy;
- technical tools to enforce, or at least detect breaches of, the policy.

Supported by policy and tools, the network manager should be able to implement the purpose of the network as far as possible with the available resources. This should be possible even if local or remote users attempt to misuse the network or the systems connected to it.

This paper will consider how protocol designers have in the past, and can in future, help or hinder the work of the network manager. It will examine the fundamental requirements of network management and the limited tools that are available to achieve them. It will look at some existing protocols and their manageability and suggest lessons that can be learned for the development of future protocols if they are to be welcome on production networks.

Requirements

In implementing policy to create a stable and reliable network, a network manager has three requirements of a protocol. These are that it be identifiable, controllable and non-hazardous. Identification should be the simplest of these: it is the requirement that a network manager should be able to determine what service or protocol a particular packet or network flow belongs to. This is fundamental to the enforcement of any policy that defines priorities for different services, for example that the services used to conduct the business of the organisation are more important than other services used only for recreation. Such policies are normally implemented by traffic shaping, limiting the bandwidth available to some services to give priority to others.

Controllability is required to implement any policy that states that certain services must, or must not, be provided on certain areas of the network. For example a policy might reasonably require that all external access into an administration network be encrypted. Unencrypted protocols should therefore be blocked at the access points into such a network. As in this example, controls often involve a consideration of the direction of a flow of traffic: rules for an internal user accessing a remote service may well be different from those for an external user accessing an internal service.

As well as protecting the network, the network manager is likely to have some responsibility to protect his users. Protocols should therefore be designed so as to minimise the hazard to users, both those who will be using the service and, especially, those who will not. It is reasonable to expect users of a service to understand and accept any risks involved, but those who do not use the service may not even be aware of its existence. They have no way to assess the risk and therefore must not be exposed to it.

Internet Protocols

Unfortunately the basic TCP/IP protocols provide very limited support for these requirements. TCP/IP was designed to be lightweight and easy to route, so IP packets contain only the information needed to deliver them from source to destination. All that is available are the addresses of the communicating computers and a port number at each end. Although port numbers are used by convention to refer to a particular service this is now an unreliable assumption because services tend to appear on ports other than their designated one (e.g. web servers on ports 8000 and 8080 rather than port 80). Identifying the direction of communication is also hard: the first packet of a TCP connection is marked as such, but for other packets and protocols it is impossible to tell whether they are part of a command sent by the computer initiating the connection or part of the reply sent the other way. Nor do packets identify the individuals responsible for them: at most the addresses show the computers involved in the exchange but when proxies or store-and-forward protocols are used these may not be the end systems where the users sit.

Policy implementation

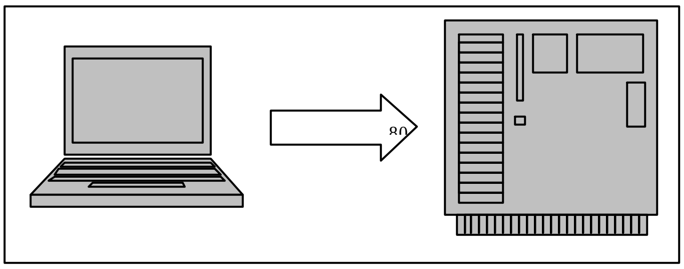
In fact although a network administrator is required to implement policies written in terms of people accessing services, all he can actually see and control are addresses, ports, an approximate location inferred from the router port at which a packet arrives and, in some cases, the direction of the packet flow. None of these map clearly onto the objects that are supposed to be controlled by the policy. The least worst solution is therefore to overload the information in the packets, making assumptions that are often correct in practice but may on occasion be badly wrong. Thus it is assumed that a particular port number always corresponds to a particular service; that a particular IP address corresponds to a single individual who owns the computer and even that the ability to use a port number below 1024 is a sign of authority and trustworthiness.

Because these controls are flawed, and also because the software that implements network services is likely to contain bugs, it is common to implement network policies using a strategy of defence in depth. Wherever possible a control will be enforced by multiple independent devices, for example if a mail server does not need a web interface that service should be blocked at a protecting router as well as turned off on the server itself. For servers, whose function is well defined and which should have the benefit of regular maintenance, it is usual to place more stress on keeping software well configured and up to date, through patches or hotfixes. Conversely for client workstations it is unlikely that policies will be actively maintained on the workstation itself so it is common to protect these by firewalls allowing most services out but only a few, if any, in.

Case studies

Existing protocols provide several examples of protocols that are manageable and several that are not. By examining the characteristics of these protocols that make them easy or hard to manage it is possible to identify features that should be included in new protocols to help network managers include them within their network policy implementations.

HTTP

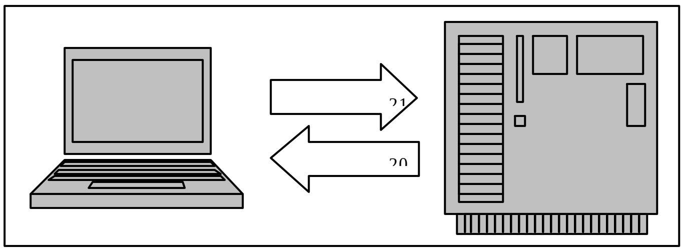


The hypertext transfer protocol is implemented using TCP. This decision has been criticised in the past for performance reasons, but from a manageability perspective it is a great help. When an HTTP client wishes to fetch a page from an HTTP server it makes a single TCP connection to that server, normally on the well-known port 80. Since RFC2068 defined version 1.1 of the protocol a single connection can be used to retrieve multiple files from the same server.

This protocol therefore meets all the requirements for manageability. The well-known destination port makes it easy to identify HTTP traffic (or at least the majority of it), and the clear distinction between the client, which initiates the connection, and the server, which accepts it makes it straightforward to implement policies that clients and servers be on different portions of the network. This allows HTTP to meet the criterion of hazard, as a machine or network segment that wishes only to act as a client, or not to participate in the protocol at all, can do so without the need to accept any inbound connections.

Recent developments in the use of HTTP discussed in RFC3205 illustrate one potential problem. As the use of web interfaces has become increasingly popular, a growing number and variety of services have been made available over the HTTP protocol. From the network management perspective it is not possible to distinguish between these different uses so it is not possible to control them individually. This may become a problem when the same protocol is used to access web pages, send e-mail (through webmail gateways), manage printers or even control network devices such as routers. A policy might well wish to distinguish between these activities, but with them all using the same protocol such a policy will be hard to implement in practice.

FTP

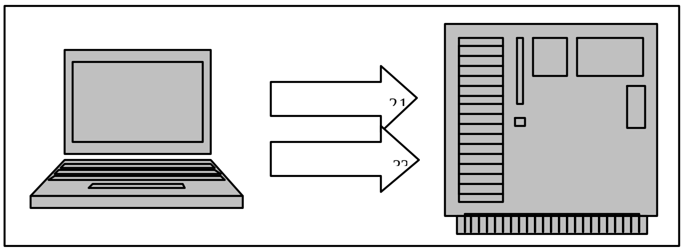


The File Transfer Protocol is also implemented as a client-server protocol over TCP, but is more complex than HTTP. As defined in RFC959, each FTP session uses multiple TCP connections, one to issue commands and others to carry listings and contents of transferred files. The client program initiates a connection to a server on the well-known control port, 21; this connection remains open throughout the session. When the client issues a command that will require either a file listing or contents to be transferred, it includes a port number on which it is listening for a connection. If the server wishes to transfer data it will initiate a TCP connection back to that port number on the client. This connection will always come from the well-known port, 20, but may go to any port number that the client chooses.

It should be clear from this description that FTP poses some manageability problems. The control connection is easy to identify and control, and carries no hazard to other systems. However the data connection is established from server to client, the opposite of the expected direction for a client-server connection. Although it is possible to identify and control FTP data connectio ns based on the knowledge that they will all have a source port of 20, such a rule is likely to match other traffic as well. As described shortly this can result in both false positives where hostile traffic is permitted by the FTP controls, and false nega tives where valid FTP traffic is blocked by rules intended to exclude other protocols.

As FTP requires clients to accept TCP connections, the protocol also involves a hazard to those clients that use it, since it is necessary for a router or firewall to allow through connections directed to them. A firewall that understands the FTP protocol and maintains the state of connections can implement a rule that data connections should only be passed through to the clients that requested them, but in many cases this complex functionality is not available. In those cases the protocol carries a hazard even to systems that are not participating in it, as it will be necessary to allow any inbound connection that appears to be an FTP data connection. In practice this means that any connection request with a source port of 20 must be allowed to pass, whatever the destination port. This weakness is well known to intruders who commonly use source ports of 20 when sending malicious packets to other services.

A further problem of FTP is that the client may choose for the data connection a port that is used by some other service. In this case any policies applied to that other service will affect the FTP data connection and may cause it to fail. For example if a site has a policy of blocking access to the Netbus remote-control service then any FTP client that happens to choose a data connection port of 12345 will inevitably fail.



To avoid these problems RFC1589 proposes a variant on the normal FTP protocol, which allows the c lient to initiate the data connection. This removes the hazard to other systems that share the client network, but at the cost of making the data connection unidentifiable, as it no longer uses a well-known port at either end of the connection. There is also an increased hazard to the server network, as valid data connections may be created to any local port. However in most circumstances this is considered a preferable option as well maintained servers should be more resistant to attack than unmaintained client workstations.

Peer-to-peer file sharing

A range of peer to peer protocols have been developed for systems such as Napster and Kazaa. These systems organize themselves into networks with all participating hosts acting as both clients and servers. As clients, the tools allow users to search for, and download, files on other hosts: as servers they publish an area of disk space originally containing local files, but in more recent versions also including files uploaded from elsewhere.

The early systems used a central directory server to track where files could be found, as well as static well-known ports for communication to the directory and between hosts. This traffic could be identified, though there was a risk of overlap with other services as the ports were not formally assigned. Some degree of control was possible, though as the protocols were not documented there was uncertainty as to the effectiveness of controls. The fact that all systems acted as both clients and servers, and therefore needed to be able to both send and receive connections, put limits on the degree of control that was possible in practice. This also meant the protocols represented a hazard to participants and to other network users, as individuals would rarely accept being limited to designated workstations to use the service. With some sites reporting that 75% of their bandwidth was taken up by these services, which few considered an operational priority, greater control was clearly needed.

Because of the difficulties in managing the protocols, an increasing number of Internet sites have therefore taken the view that peer-to-peer protocols should be banned from their networks. This has led to a form of arms race as protocol designers deliberately attempt to make their protocols impossible to identify or block. Recent systems switch ports and protocols, and even masquerade as other services in order to bypass firewalls and router blocks. As a result considerable ingenuity has been deployed for the wholly negative purpose of preventing managers from providing stable networks for the delivery of services.

New challenges

Developments in the use of network addresses are also presenting challenges for the implementation of policies. In particular there is now much more use of dynamic address allocation through DHCP or Network Address Translation. These make it much harder for a protocol to be controlled at the level of a single workstation, as the address of the workstation may change from one day to the next. The same policies are therefore normally applied across the whole of a dynamic address pool. This means that address pools must be allocated so that computers and users to which different policies apply do not share the same pool. Dynamic addresses are also a problem for services that assume, either explicitly or implicitly that addresses do not change. For example file sharing services often continue to use the same address even after it has been re-allocated to another workstation. The results of this are at least confusing and can have serious implications for privacy.

Although the future deployment of IPv6 will have benefits for privacy of network traffic, it will do little to affect these network management issues. Communications will still use port numbers as an indication of the service they belong to and complex services that involve multiple flows will still be hard to identify and manage.

Features of manageable protocols

From this discussion, certain features can be identified that make a network protocol manageable. These should be considered desirable, even essential, in designing new network communications systems if these are to be welcome on today's multi-use, production networks.

Protocols should give rise to identifiable network flows. This requires that they use well-defined ports and that those ports should not be shared between protocols that are used by different services. Conversely, if different, separate, services are provided then these should use different ports to allow separate management of these services.

Considerate use of ports is needed for a protocol to be controllable, but in addition the protocol should allow the direction of the flow of information to be determined. There should be no confusion between the flow caused by a local client requesting a remote service and a remote client requesting a local service. Connections should be initiated by a client, and remain under the control of that client.

Protocols should allow participating systems to use multiple, independent layers of protection against hazard. They should expect to encounter firewalls, and be prepared to work with proxies. For some particularly complex protocols, an application-layer proxy may be the only way to identify and support the service. Above all, a single user's desire to use a service should not expose the whole network to an increased threat.

As computers and users become ever more mobile and their connectivity more dynamic, protocols that assume addresses will remain constant beyond the life of a single connection will become unusable. If authentication is required then this needs to be done at the application layer, not assumed from an IP address. If a particular user needs to be recognised or contacted on subsequent visits then this should be done using an application layer identity.

Conclusion

Network managers are responsible for managing networks. Networks are a limited, shared, resource, so managers need to ensure that use of them is allocated and prioritized in accordance with the policy of the organization. Protocols that can be identified and controlled, that permit straightforward allocation and prioritisation of resources, make this job much easier. Most mixed-use networks will have a policy of reducing, as far as practical, the risk to systems connected to the network and their users. Protocols that represent a low risk, or allow the exposure to risk to be limited to an informed subset of the network and its users, are more likely to be welcome on a network than those that represent a threat to all users. As the pressure on network managers grows to provide stable, safe, network services, protocols that cannot be controlled, or that present a threat to the service, can only expect to be banned entirely from the network. The lessons of peer-to-peer filesharing, which might have been tolerated on many networks if it had been controllable, must not be ignored.

References

RFC2068 – Fielding, R, Gettys, J, Mogul, J, Frystyk, H, Berners-Lee, T, Hypertext Transfer Protocol – HTTP/1.1
RFC3205 – Moore, K, On the use of HTTP as a Substrate
RFC959 – Postel, J.B and Reynolds, J.K, File Transfer Protocol
RFC1579 – Bellovin,S, Firewall-friendly FTP