# Overview

- **Network security** involves protecting a host (or a group of hosts) connected to a network
- Many of the same problems as with stand-alone computer systems apply and are more difficult:
  - User authentication and authorization – determine the identity and privileges of users accessing the system
  - Access control – limiting what actions are permitted

# Additional Challenges of Network Security

- Networking increases message vulnerability to:
  - Interception
  - Modification
  - Destruction
  - Delay
  - Reordering
  - Repetition
- Networking implies cooperation, sharing, and trust
- Networking:
  - Exposes a system to a larger pool of potential attackers
  - Decreases the likelihood of intruders getting caught

# Authentication and Authorization

- Issues:
  - For the Server:
    - Is the Client really who they say they are?
    - Is the request from the Client fresh?
    - Will an eavesdropper be able to read my response?
  - For the Client:
    - How do I know I'm really talking to the Server?
    - Will an eavesdropper be able to read my request?

# Kerberos - Overview

- Trusted third-party authentication service for computer networks
  - Developed at MIT
  - Client-server architecture
- Capabilities:
  - Client program requesting a service can prove the identity of the user on whose behalf it is operating
  - Clients can also (optionally) ask a server program to authenticate itself
  - Kerberos can protect the privacy and integrity of messages between clients and servers

# How Kerberos Works

- With each user, shares a secret DES key
- Phase 1: user obtains credentials (from Kerberos) to be used to request access to other services
- Phase 2: user requests authentication (from Kerberos) for a specific service
- Phase 3: user presents credentials to a server

# Kerberos Credentials

- Tickets
  - Generated by Kerberos
  - Valid until expiration
  - Used to securely pass the identity of the person to whom the ticket was issued from Kerberos to a server
  - Contains:
    - Person's identity
    - Information to show that the person using the ticket is the person to whom it was issued
- Authenticators
  - Generated by the user
  - Valid only once
  - Used to show that the person using the ticket is the person to whom it was issued

# Kerberos Credentials (cont)

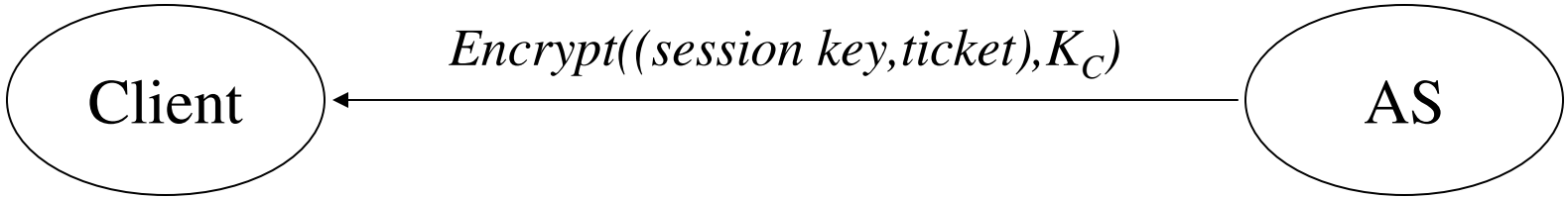
- Ticket = Encrypt((Server,Client,Addr,Timestamp,Lifetime,$K_{S-C}$),$K_S$)
- Authenticator = Encrypt((Client,Addr,Timestamp),$K_{S-C}$)
  - *Server* name of server
  - *Client* name of client
  - *Addr* client's IP address
  - *Timestamp* time ticket was generated
  - *Lifetime* amount of time for which ticket is valid
  - $K_{SC}$ session key to be shared between Server and the Client
  - $K_S$ DES key shared between AS and the Server

# Getting the Initial Ticket

- User enters username
- Request for ticket for *ticket-granting service* (TGS) sent to **authentication server** (AS)

# Getting the Initial Ticket (cont)

- AS checks that Client is a valid user
- Generates a session key, $K_{C\text{-}TGS}$, for the Client and the TGS
- Generates a ticket, *Encrypt((TGS,Client,Addr, Timestamp, Lifetime,*$K_{C\text{-}TGS}$*,,),*$K_{TGS}$*)*, for the Client to use for the TGS
- Sends session key and ticket back to Client (encrypted with Client's key, $K_C$)

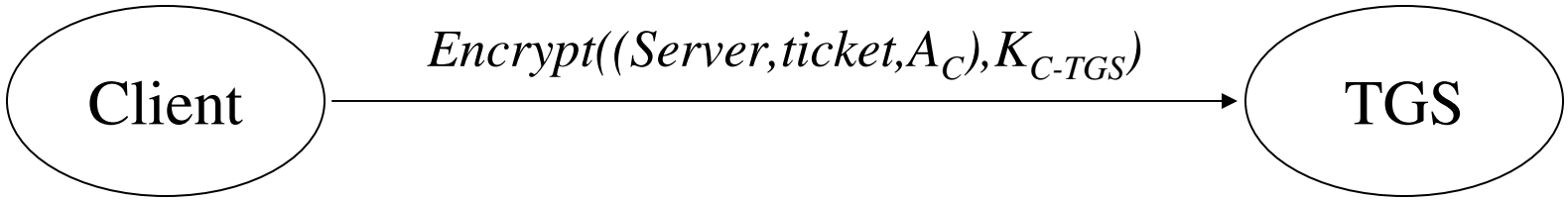


# Getting the Initial Ticket (cont)

- User enters password
- Password is converted to a DES key and used to decrypt the AS's reply
- Client's machine:
  - Stores session key and ticket
  - Erases the user's password and DES key from memory

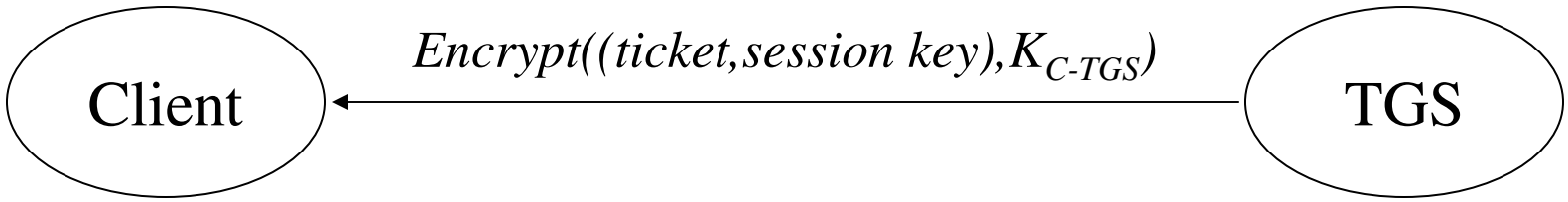
# Getting a Ticket for a Server

- Client contacts TGS and requests a ticket for Server:
  - Name of Server
  - Client's TGS ticket, *Encrypt((TGS,Client,Addr, Timestamp, Lifetime,$K_{C\text{-}TGS}$,,),$K_{TGS}$)*
  - Client's authenticator, *Encrypt((Client,Addr,Timestamp), $K_{C\text{-}TGS}$)*
- Client's request is encrypted under its session key with the TGS, $K_{C\text{-}TGS}$



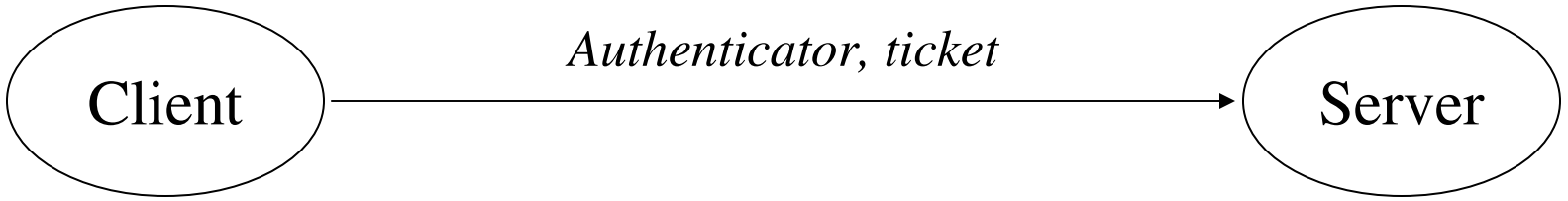
# Getting a Ticket for a Server (cont)

- TGS:
  - Checks the ticket and authenticator
  - Generates a session key, $K_{C\text{-}S}$, for the Client and the Server
  - Generates a ticket, *Encrypt((Server,Client,Addr, Timestamp, Lifetime,*$K_{C\text{-}S}$*,,),*$K_S$*)*, for the Client to use for the Server
  - Sends session key and ticket back to Client (encrypted with session key the Client and TGS share, $K_{C\text{-}TGS}$)



# Requesting a Service

- Client:
  - Builds an authenticator, *Encrypt((Client,Addr,Timestamp),* $K_{C-S}$*)*
  - Sends authenticator and ticket, *Encrypt((Server,Client,Addr, Timestamp, Lifetime,*$K_{C-S}$*,,),*$K_S$*)*, to the Server

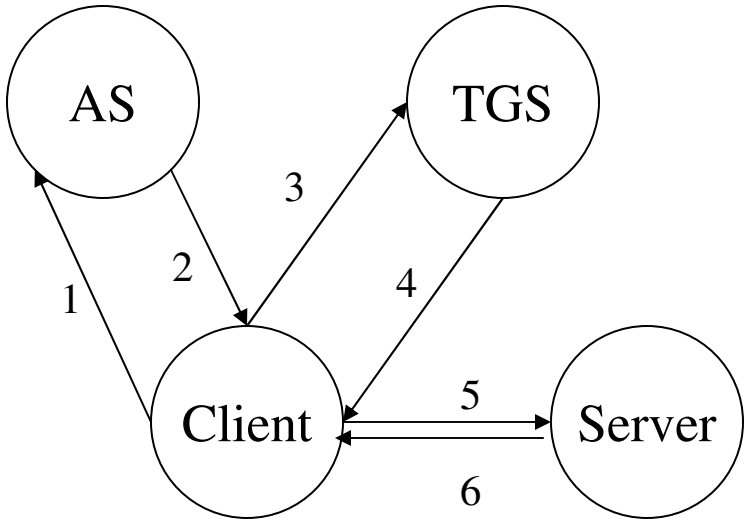


# The Server's Response

- Server:
  - Decrypts and checks the ticket (learns the session key)
  - Decrypts and checks the authenticator
  - Optionally: increments the Timestamp by one and returns it to the Client encrypted with the session key

# Overview of Kerberos Messages

1. Request for TGS ticket
2. Ticket for TGS
3. Request for Server Ticket
4. Ticket for Server
5. Request for service
6. Server authentication

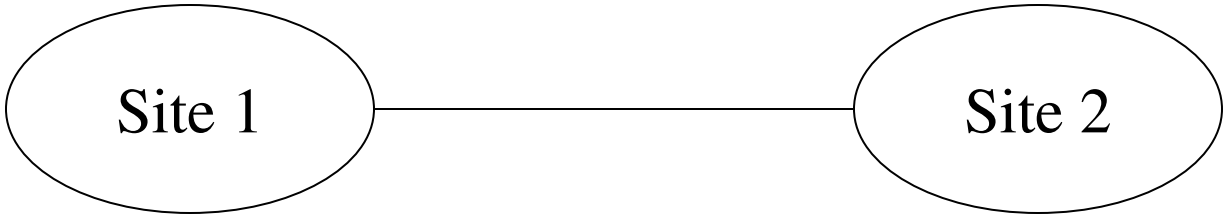


# Kerberos Limitations

- Applications must be “Kerberized”
- Based on:
  - Client/server model
  - Synchronized clocks
- The TGS could be a bottleneck
- Cross-realm operation doesn’t scale well

# Interaction With Other Sites Using Kerberos

- Both Site 1 and Site 2 run Kerberos:




- Can clients at one site use Kerberos to access servers at the other site securely?

# CORBA - Overview

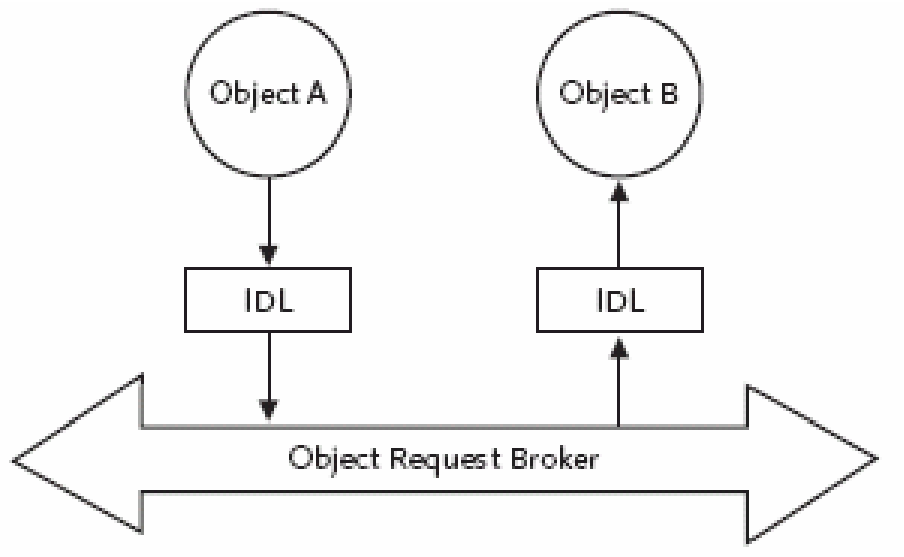
- Developed by the Object Management Group (OMG)
- Standard that allows distributed applications, running in heterogeneous environments, to interoperate
  - Objects are entities that provide services to requestors through well-defined encapsulating interfaces
  - A reference model describes how the objects interoperate by requesting services from one another

# CORBA – Example

- Two objects, A and B
- Assume that Object B has a method named foo( )
- Object A can request service from B by invoking foo()
- Problem: A and B might be implemented in different languages
- Solution: a translation may be necessary to allow A to understand B's request
  - A universal **Interface Definition Language** (IDL) allows A's request can be converted from A's native form into a request understandable to B

# The Object Request Broker (ORB)

- The Object Request Broker (ORB) mediates the interaction between the objects



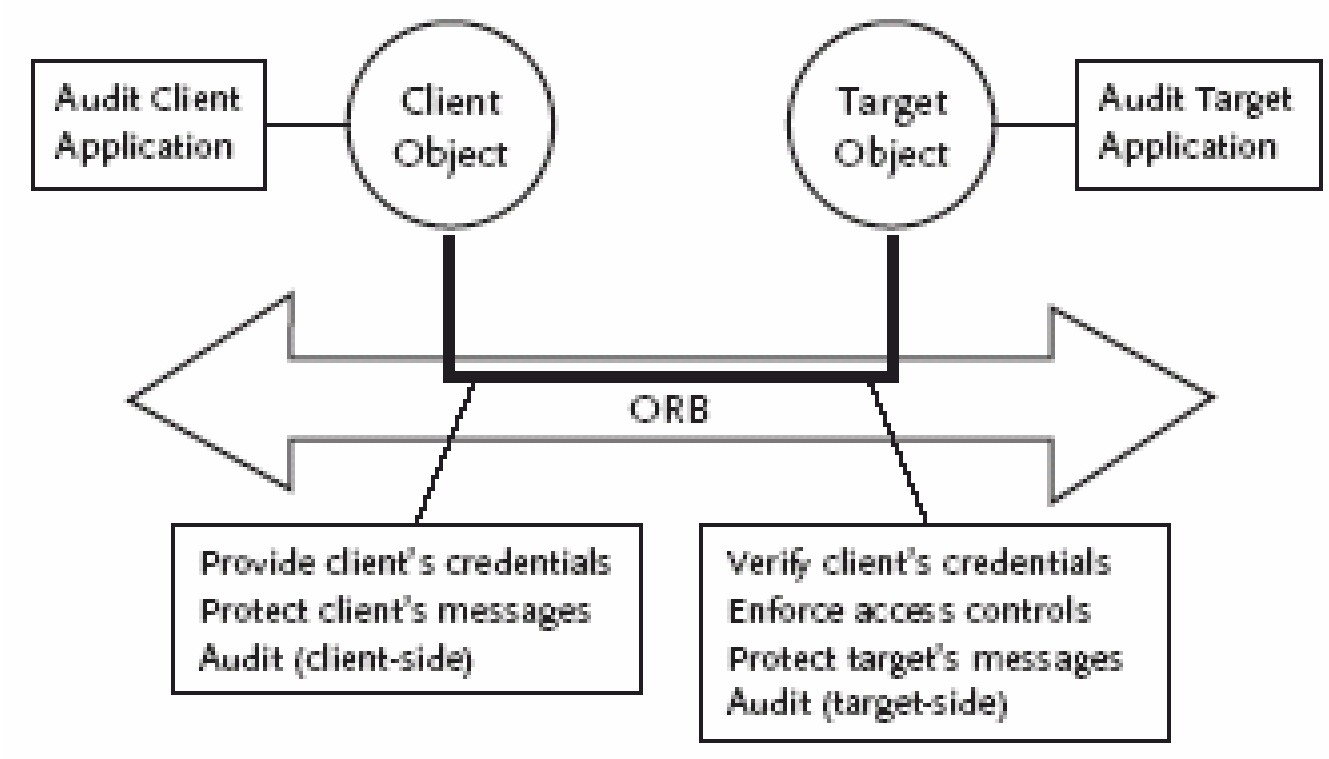
# Functions of the ORB

- Deliver A's request to B and B's reply to A
- Hide "low-level" details from calling objects:
  - Location (local or remote)
  - Implementation details (language and platform)
  - Execution state (currently running or needs to be started)
  - Communication mechanisms (TCP/IP, shared memory, local method invocation)

# CORBA

- The **Common Object Request Broker Architecture** (CORBA) standard:
  - Defined by OMG
  - Allows different ORBs to interoperate
- The CORBA Security specification:
  - Optional
  - If implemented, the ORB provides basic security functionality to all objects:
    - Authentication
    - Communications security
    - Access control
    - Auditing

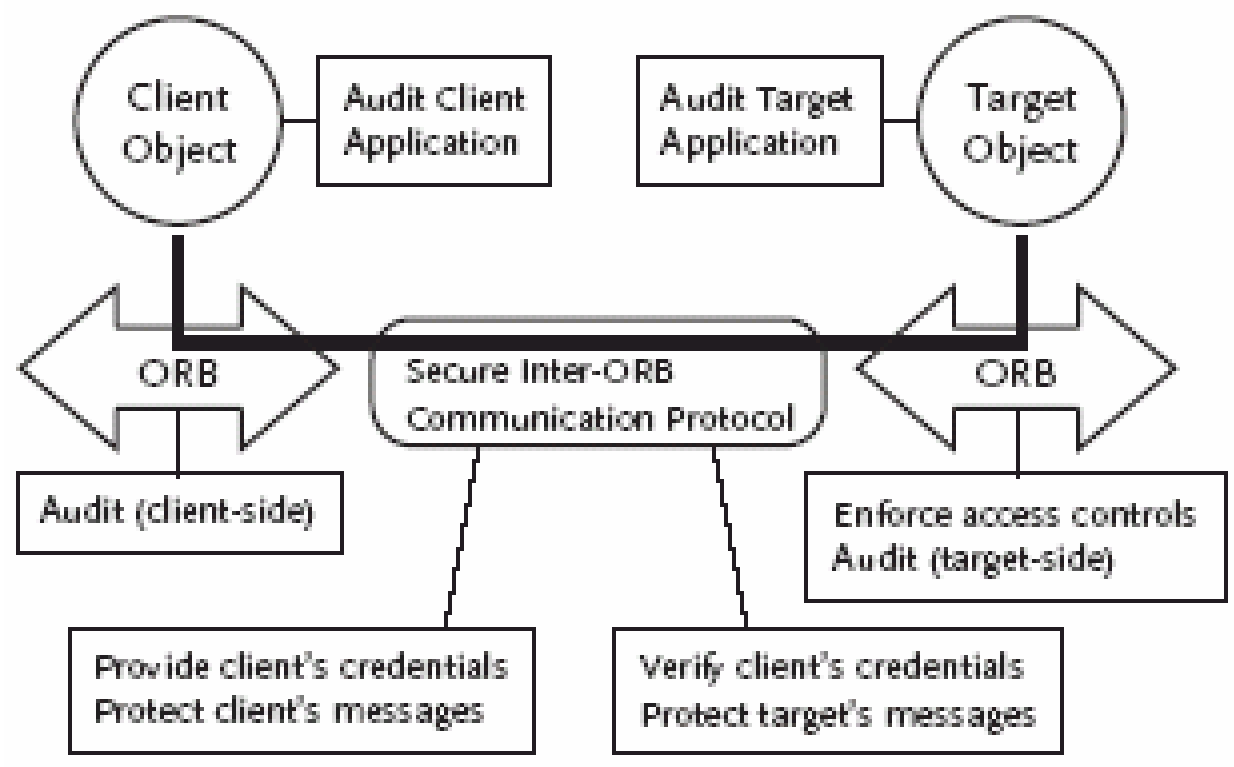
# Services of a Secure ORB



# Secure Interoperability

- Problem: A client and target object may be distributed so that their interaction is not mediated by a single ORB
- Solution: the secure inter-ORB protocol (SECIOP):
  - A standard interoperability protocol defined by CORBA
  - Establishes a secure communication channel between two ORBs
  - Allows authentication and message-protection data to be exchanged securely and in a format that all compliant ORBs understand

# Interaction Between Two Secure ORBs



# User Authentication and Authorization - Summary

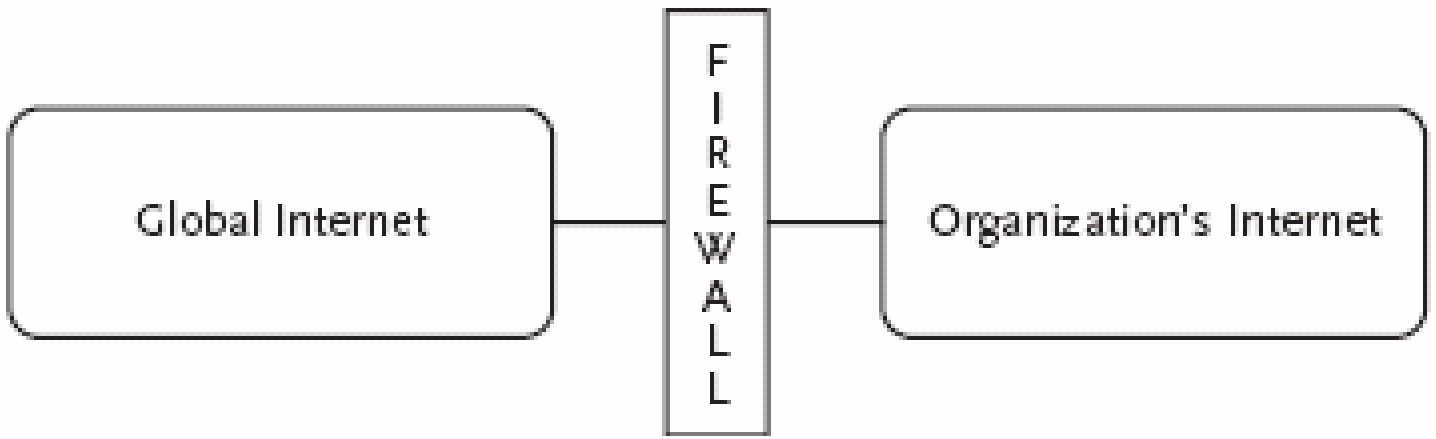
- Very difficult in a network environment:
  - Authentication - determining a user's identity
  - Authorization – determining what actions a user can perform
- Reasons:
  - Vulnerability of network communications
  - May be controlled by several different administrative authorities
- Solutions:
  - Kerberos
  - Secure ORBs

# Access Control for Networks

- Problems:
  - Enforce an access control policy
    - Allow trust relationships among machines
  - Protect local internet from outsiders attempting to:
    - Obtain information, modify information, disrupt communications
- Solution: firewall
  - Forms a barrier that protects one network from dangers on another
- History:
  - Fireproof walls that are often used in buildings to form a barrier across which fire cannot spread
  - Helps to contain a fire and limit the amount of damage it can do

# Firewalls

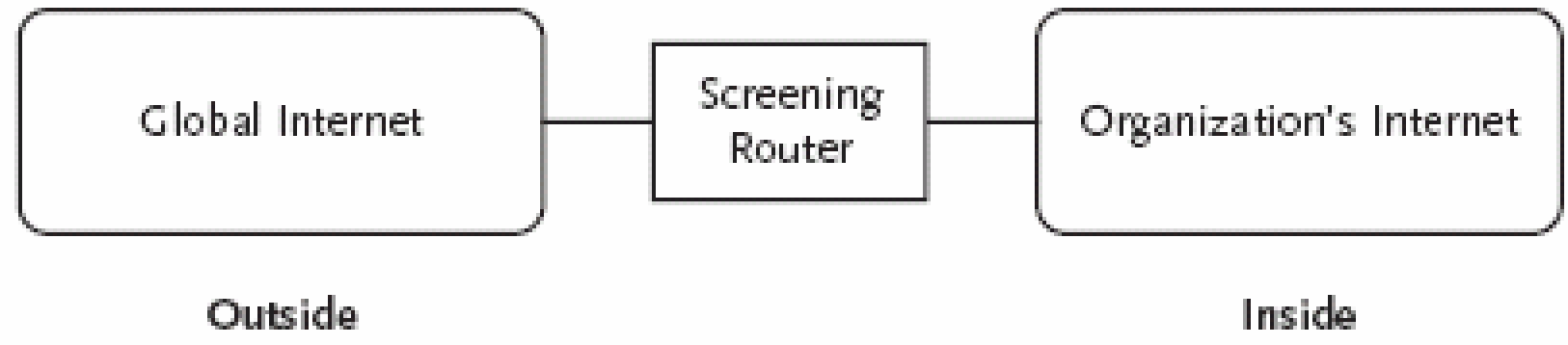
- A firewall can:
  - Partition machines into those inside the organization and those outside the organization
  - Enforce an access control policy about what types of traffic are allowed in and out



# Implementing a Firewall with a Screening Router

- **Screening routers** perform packet filtering:
  - Examine some fields in the packet header:
    - Source and destination IP address
    - Protocol
    - Source and destination port numbers
  - Allow a packet to pass if it meets the screening criteria
  - Filtering rules are **stateless** to increase speed

# A Screening Router



# Filtering Rules

- Administrator can specify rules regarding which packets should not pass through the firewall
- Can block:
  - Outgoing packets to certain addresses - restrict which outside sites local users can access
  - Incoming packets from certain addresses - restrict access to specific external sites
  - Incoming and outgoing requests to specific services
  - Etc.

# Sample Filter Rules

| Incoming/ Outgoing | Source IP Address | Destination IP Address | Protocol | Source Port | Destination Port |
|---|---|---|---|---|---|
| Incoming | * | * | TCP | * | 79 |
| Incoming | * | * | UDP | * | 69 |
| Outgoing | * | 128.112.*.* | * | * | * |

- Row 1: Block incoming packets from any source to any destination for the *finger* service (TCP port 79) should be blocked
- Row 2: Block incoming packets bound for the TFTP service (UDP port 69)
- Row 3: Block outgoing packets bound for any machine on network 128.112

# Screening Routers

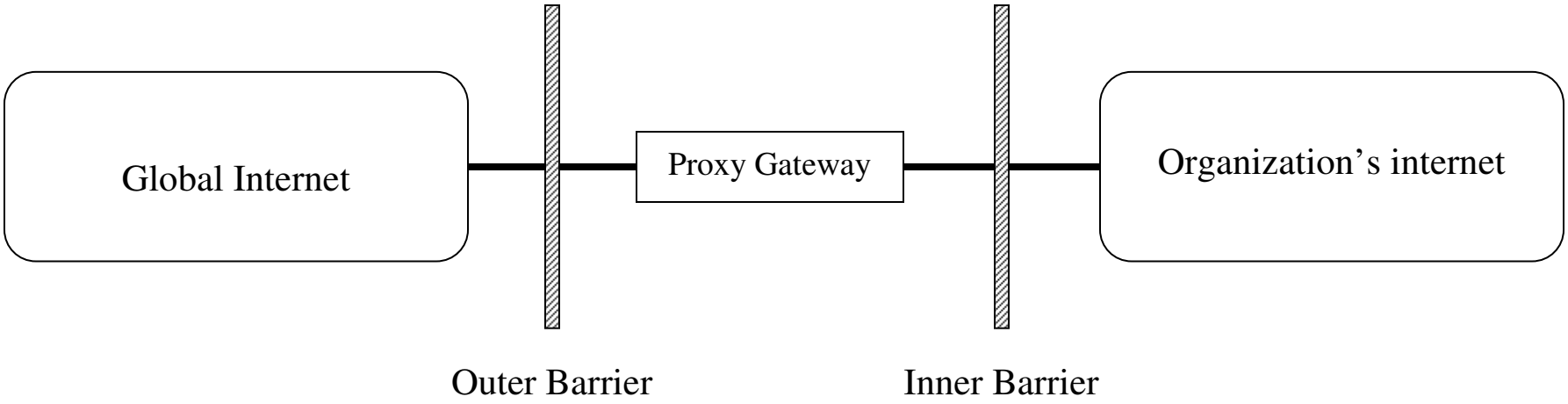
- Advantages:
  - Relatively cheap
  - Help improve security by blocking packets from/to dangerous sites and services
- Disadvantages:
  - Still vulnerable to attacks on enabled services
  - Potential services are large (and growing) requiring frequent maintenance
  - Decisions must be made statelessly

# Implementing a Firewall with a Proxy Gateway

- **Proxy gateway** more powerful than a screening router and can do more/better checking:
  - Examine data (not just header) portion of packets
  - Remember the past behavior of a connection
  - Consider context – is this a response from the outside to a request that originated on the inside?
  - Etc.

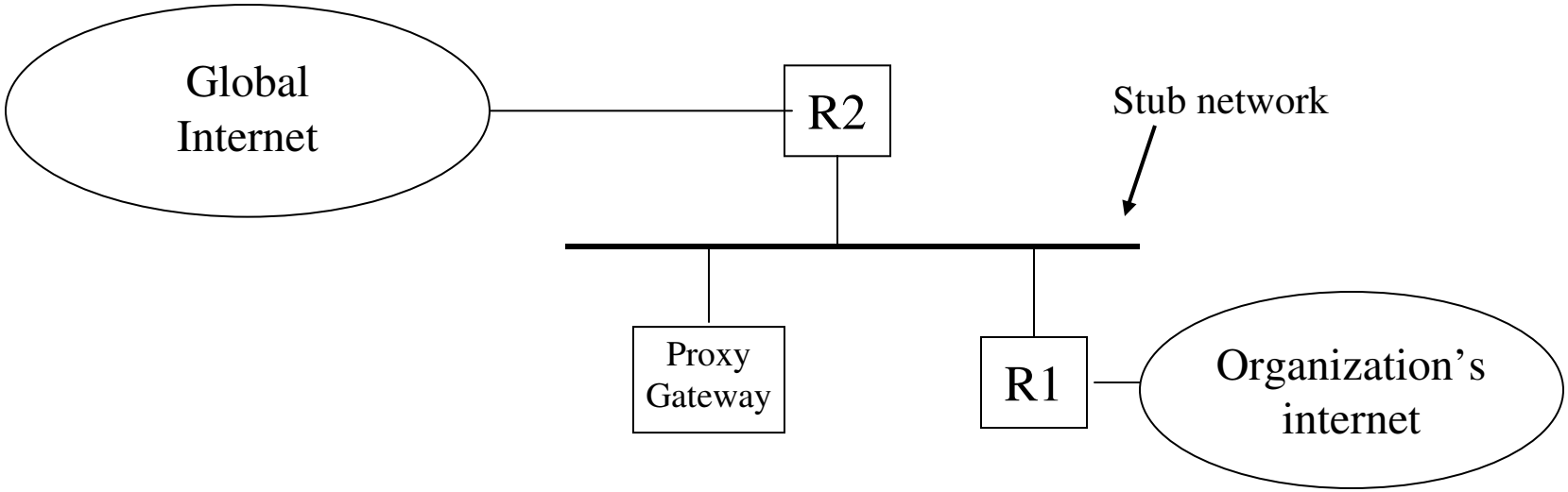
# Proxy Gateways

- Two barriers:
  - Outer barrier: blocks all incoming/outgoing traffic not to/from the proxy gateway
  - Inner barrier: blocks all incoming/outgoing traffic not from/to the proxy gateway



# Proxy Gateways (cont)

- Each barrier is implemented by a screening router:
  - *R2* blocks all traffic not destine for the proxy gateway
  - *R1* blocks all traffic not from the proxy gateway



# Proxy Gateways (cont)

- The proxy gateway typically runs a set of **application gateway programs**
- Act as middlemen between hosts inside and outside the firewall
  - Internal hosts communicate with the application gateway program running on the proxy gateway
  - Application gateway program relays request to the external host
  - The external host's reply is sent to the application gateway program
  - Application gateway program performs some checking and then passes the reply on to the internal host

# Proxy Gateway - Example

- An FTP server behind a proxy gateway firewall
  - An external client issues commands to establish a connection and transfer files
    - Proxy gateway acts as a middleman between the client and server
  - The proxy can check incoming commands:
    - Pass only valid FTP commands on to the server
    - Protects the server from malformed or dangerous input
  - If the external client attempts to upload a file to the server:
    - The proxy could pass the file through virus-scanning software

# Proxy Gateways

- Advantages:
  - Can provide better protection than a screening router
- Disadvantages:
  - Additional cost
  - Proxy gateway could be a:
    - Bottleneck
    - Single point of failure
    - Tempting target for attackers

# Dynamic Firewall Techniques

- Screening routers and proxy gateways enforce static security policies
- Dynamic filters allow administrators to set up **triggers**:
  - Temporarily add, delete, or modify certain rules in response to particular events
- Provides additional flexibility:
  - Permit or deny traffic in special circumstances
- Provides additional security:
  - More stringent rules triggered when suspicious traffic is observed

# Network Access Control - Summary

- Access Control – need to protect local machines/networks from outsiders attempting to:
  - Obtain information
  - Modify information
  - Disrupt communications
- Solution: firewalls (screening routers, proxy gateways, etc.)
  - Forms a barrier that protects one network from dangers on another