Chapter 1

INTERNET TECHNOLOGIES

INTRODUCTION

In the context of industrial information technology, the Internet and World Wide Web increasingly are seen as a solution to the problem of providing "anywhere, anytime" services. In the classical view of an Internet security-enabled IT infrastructure, services are requested and consumed by a user (a human requesting plant production data from his or her desktop); and, data are provided by an origin server (a Web server located in a plant that can authenticate users, implement encryption, serve data, and source multimedia streams). This rather simplistic view works well if the number of users is small, the complexity of services required is modest, and the real-time response requirements are lax. However, it fails to scale when one accounts for the complexities of modern networking: many simultaneous users, potentially operating in multiple languages; many complex data types, including incompatible display formats; many differing schemes for implementing privacy [6] and Internet security through many combinations of authentication and encryption [1].

THE WEB CLIENT/SERVER ARCHITECTURE

Most Internet security software applications have been developed using the classic client – server model – multiple servers hold vast quantities of information of all kinds, and clients of all types reach that information via diverse devices. Originally, this was considered an ideal architecture because all the processing occurred at the network "ends" (clients and servers), thereby allowing the network itself to remain blissfully unconcerned about the type of traffic it was transmitting. This original concept was known as the end-to-end principle of the Internet [1]. As differences in client capabilities arose (hardwired versus wireless connections [5], large monitors versus PDA screens), they were either ignored entirely by applying a "lowest common denominator" approach; or else, they were accommodated by having the server hold data in multiple formats and using special-purpose protocols

to negotiate which format to deliver to which device (creating a specialized cell phone interface) [1].

As Web traffic types became more complex and as clients became more diverse, this pure client – server architecture became less attractive. Caches near the client and at network-edge delivery points were added to make operations faster; gateways (such as for wireless devices) were added to help diverse groups of clients connect; Content delivery networks (CDNs) were invented to organize the vast array of Internet content for the client [1]. All of these services operated "in the network." These services grew up independently, making extensions and new services both vendor specific and hard to manage. Developing network services became tedious and expensive. With today's evolution of diverse client devices, content providers were forced to offer data in every format necessary for every connecting device: PDAs, PCs, cell phones, laptops, e-book readers, etc. [1].

These difficulties were not limited to differing client device and connection capabilities; they also arose from differing client preferences. Client preference with regard to, say, the language of presentation leads to maintaining multiple versions of a common information store; for example, a multinational company might keep duplicate databases in English, French, German, Japanese, etc. Resolving client language preferences at the server increases Internet traffic, while maintaining multiple versions of a Website introduces the obvious difficulties of version control. Although in-the-network services can be quite diverse, they share several key similarities:

- Service detection and operation occurs at in-the-network facilities (proxies) that are in the path between client and server.
- The in-the-path service operation is frequently done with devices both inside and outside the path between client and server (called the outside-path device a "callout server").
- Similarly to a cache, these services transform and organize information but rarely "own" the content (as is done by a server) [1].

However, as noted earlier, each of these services was uniquely designed to suit a particular system and this has resulted in significant maintenance overhead for these software services. To address this problem, an industry group meeting at the Internet Engineering Task Force (IETF) proposed Open Pluggable Edge Services [1] (OPES) as an alternative, unifying architecture for these in-the-network services. OPES defines an architecture for building these stateless services that is suitable for content delivery services, client preference implementation and device adaptation. A client's request is intercepted by an in-the-net device, which is called NetEdge, and is amended (personalized) by client information retained in the edge server, and then passed to the relevant origin server(s). Servers essentially own their content; NetEdge may cache content while making presentation style changes, but NetEdge never owns the data. In the model, an origin server never reformulates its content. All content transformations occur in the NetEdge using well-defined protocols at the request of one end (server) or the other (client). Examples of value-added NetEdge services include language translation, media adaptation, rate adaptation, image transcoding, local content insertion (weather forecast), edge assembly, authentication, and virus scanning. All services benefit from the NetEdge's knowledge of its clients' preferences and capabilities [1].