1995

# HTTP-NG Architectural Overview

*Simon Spero, UNC Sunsite/EIT (ses@unc.edu)*

## About HTTP-NG

HTTP-NG is an enhanced replacement for HTTP/1.0. It is designed to correct the known performance problems in previous versions of HTTP, and to provide extra support for commercial transactions, including enhanced security and support for on-line payment.

## About this document

This document provides an architectural overview of the new protocol. It gives an overall view of how the protocol works, and explains how various operations and features interact with each other and the network. This document is intended for a general audience, and is not a technical specification. This is not an official W3O document.

## About this series

This document is part of a series describing HTTP-NG.

1. Architectural Overview
2. The Session Control Protocol
3. Introduction to ASN.1 and the Packed Encoding Rules
4. HTTP-NG Specification
5. Security Profiles
6. Payment Profiles

### Why do we need a new protocol?

HTTP is the fastest growing protocol on the internet. It is simple to implement, and thousands of people use it every day to browse through gigabytes of on-line hypertext. If HTTP is working so well, why replace it? The driving forces behind this change are the twin needs of performance and commerce.

### The Need For Performance

Part of the need for a new protocol is caused by the very fact of HTTP's success. The original protocol was designed to be a simple way of transferring a file between two machines. This led to several very serious performance problems, adding extra delays to the time taken to fetch pages, and preventing browsers from making efficient use of the network.

MICROSOFT
EXHIBIT 1011

### Commercial Applications

Wide Web, the protocols currently in use do not support several very important features needed for electronic commerce. Currently there is no reliable way to find out the identity of someone trying to access a document. There is also no way to convey information about charging and prices; nor is there a way of dealing with on-line payment.

### Can these changes be made by just tweaking HTTP 1.0?

The major problem in making just minor changes HTTP 1.0 is that the modifications which are needed change the fundamental model on which the protocol is based. Each extension adds more and more complexity to the protocol; as more and more enhancements are made, the problem becomes much worse. Since HTTP 1.0 is a simple protocol, it is much simpler to recreate the existing functionality as part of a new protocol than to attempt to kludge the new functionality into a protocol not designed to handle such changes.

## User Requirements

Any new protocol needs to meet the needs to three different groups. Individual users want to be able to browse the web without being forced to wait for pages to be delivered. Information providers need to be able to support large numbers of users, and to restrict access to authorized users, and to get money from their paying customers. Software developers need a system which is easy to implement, but which can be optimized and enhanced to differentiate between products.

### Simplicity

HTTP-NG must allow simple implementations to be implemented simply without penalising more optimized systems. The protocol should be designed to work well for the commonest cases

### Performance:

HTTP-NG should allow objects to be transferred over wide area networks efficiently.

### Asynchronicity:

HTTP-NG should allow a client or server to initiate a new request without waiting for previous requests to complete. It should be possible to transfer multiple objects in parallel over a single connection.

### Security:

HTTP-NG should support the transfer of encrypted objects. The protocol should not impose a single security policy or mechanism.

### Authentication:

HTTP-NG should support mutual authentication between all parties involved in a transaction. It should be possible to relay authentication information through multiple untrusted intermediaries. It should also be possible to use multiple authentication contexts over a single connection.

### Charging:

HTTP-NG should provide support for on-line payment schemes such as **First Virtual** and **DigiCash**. The protocol should not impose a single payment policy or mechanism.

### Intermediate Servers:

mirrors, and to allow intermediate servers to relay usage information to originating sites.

### Mandatory display:

HTTP-NG should support the mandatory display of information relating to an object, such as licensing information, copyright, authorship.

### Logging Information

HTTP-NG should support the transfer of logging information between intermediate servers and the original source of a resource. Users should be able to specify restrictions on the use of logging information, and to discontinue transactions if such restrictions cannot be satisfied.

### Network requirements - Transports of Delight

HTTP-NG is transport-layer independent, and can arrange for data transfer using different transports. However, since all uses of HTTP-NG in the current internet will be over TCP, HTTP-NG must work well over TCP.

### TCP and the Internet

Many performance problems in HTTP are a result of not taking Transport layer interactions into account. Table 112 lists some important points that need to be taken into account when designing a protocol.

### Connection setup costs

When TCP sets up a connection, it sends connection request to the server, and waits for the connection to be accepted or rejected. This adds a delay of one Round Trip Time.

### Slow start limits transfer rates during start up.

When a connection is first started, TCP initially sends only a small amount of data. The amount of data that can be transmitted before the sender must wait for a reply is increased until a steady state is reached. If congestion occurs and packets are lost, the server slows down the send rate until a new stable point is reached.

Slow-start particularly affects the first request on a new connection. If the request won't fit into a single segment, the client must wait an extra Round Trip before it can finish sending the request.

### Congestion information is not shared between connections

Although slow-start is used to converge upon the correct transmission rate for the path between two computers, this information is not shared between different connections to the same host. Thus is there are several connections running between a pair of hosts, if the path becomes congested, the connections will interfere with each other, leading to poor throughput.

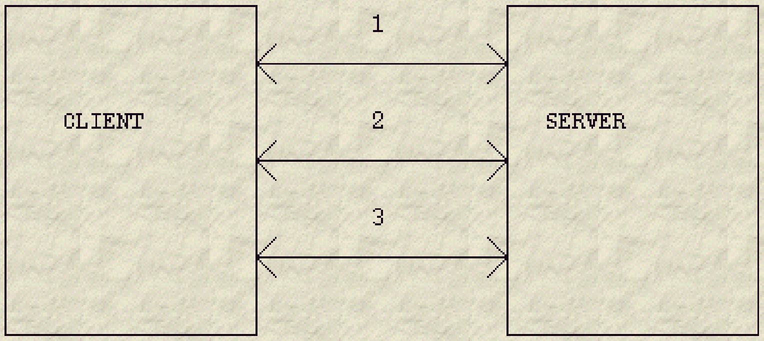
### Special purpose transport layers.

As resource reservation and guaranteed bandwidth become more commonplace, and as ATM moves into more general deployment, it becomes important to be able to make use of special purpose transport layers which are optimised for certain types of media, for example, a multicast video transmission, or a ATM telephone channel. HTTP-NG should be able to refer a user on to another service for actual data trans- fer, whilst still being able to handle the relevant control information.

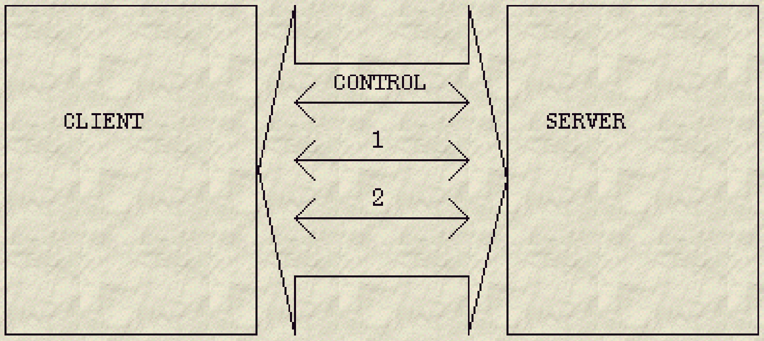
---

## Protocol Model

HTTP 1.0 works by creating a new transport connection for each request. The client sends a request over the connection; the server replies over the same connection, sending information about the response, followed if possible by the requested data.



HTTP-NG uses a different model. Instead of creating a separate connection for each request, HTTP-NG creates a single connection which can be used for many different requests. The connection is divided up into a number of virtual sessions. One of these sessions is used to carry control information - requests from the client, and meta-data from the server. The other channels are used to carry the requested objects.



## Requests and Responses

Each request and response is sent as an HTTP-NG message. Some parts of a message are the same in all cases. For example, each message may need to be signed by the sender to make sure that the request is genuine. Other parts of the message are different for each operation. A GET request will specify a list of the objects to fetch the response will contain information describing the retrieved objects.

All of this information must be encoded in some way before it can be sent over the network. HTTP 1.0 uses a text based syntax to encode requests, and a modified form of MIME to handle responses. This textural representation is easy for humans to understand, but quickly becomes extremely complicated when used to handle requests with complicated structure.

To avoid this complexity, HTTP-NG uses a different way of describing and encoding the request message. The scheme used is a simplified form of ASN.1 and PER (Abstract Syntax Notation, and Packed Encoding Rules). This scheme allows efficient, compact parsers to be generated automatically, whilst remaining simple enough to allow hand-crafted parsers to be built easily.

HTTP-NG messages can be sent at any time. In the typical case, the client sends a request message to the server, and gets back one or more responses in return. The client doesn't have to wait for a response to one such as Mosaic Communications NetScapeTM to request objects as soon as they discover they are needed, and allows intermediate servers to handle requests from different clients to the same server in a fair and efficient manner.

Although the usual pattern is for the client to issue requests to the server, there are some cases where this pattern needs to be reversed. One example for this is the need to warn browsers if a requested action would result in a hefty charge. In this case, the server needs to be able to check to see if the client still wishes to perform that action, and to discuss with the client how sir wishes to pay.

## Negotiations - The Art of The Deal

Because HTTP-NG doesn't restrict the types of objects which can be requested, and does not impose a single security or payment policy, there needs to be some way of negotiating which types and mechanisms a client and server can support.

HTTP 1.0 allowed clients to propose a list of types which they were able to accept; unfortunately the mechanism used had several problems. Because the client had to sent a list of all possible types with each request, requests typically contained over 1K of type information. This caused serious performance problem, and required a lot of effort to process correctly. As a result of this complexity, very few servers properly supported this type negotiation.

HTTP-NG tries to simplify negotiation by adding an extra mechanism to support the most common cases without affecting more complicated situations. This new form of negotiation is based on the observation that the vast majority of all WWW traffic involves the exchange of just a few well-known object types. HTTP-NG defines a small list of these well-known types, and allows sets of these types to be encoded in a short bitmap. A text only browser would sent a bitmap indicating support for just HTML and plain text. A graphical brower would claim support for several graphics and sound formats. A server or browser which only supports these types need only support this simple form of negotiation.

To indicate support for other types. HTTP-NG also allows the client and server to add extra items to the bit-sets. The proposer send a message containing the option being proposed (for example, text/ms-word), together with a numeric code which will be used in later messages to refer to the proposed value.

This kind of indirect reference is useful, because it allows the proposed types and values to become more structured without affecting performance. This extra freedom can be used to support parameterized types; for example, if a client is running on a 4-bit display, a server can avoid generating a 24-bit deep image.

The same method is used to indicate supported security schemes, authentication information, and payment mechanisms.

## Security

Because there are so many different security schemes and policies, HTTP-NG provides a general security framework into which the various security components can be fitted. This results in several architectural decisions. One such is the absence of any special support for certificate exchange; this exchange is handled using the Fetch request (which is after all the most fundamental operation in HTTP-NG).

The HTTP-NG message wrapper has fields which can carry arbitrary authentication and security information. This allows each message to be individually authenticated.

HTTP-NG also allows an intermediate server to relay authorisation and verification to and from another server on behalf of one of its clients. This feature allows untrusted proxies to cache encrypted data, and to relay the information needed to decrypt this data to the end user without having to be able to decode the document itself.

## Charging and Payment

# Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

## Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

## Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

## Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

## API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

**LAW FIRMS**
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

**FINANCIAL INSTITUTIONS**
Litigation and bankruptcy checks for companies and debtors.

**E-DISCOVERY AND LEGAL VENDORS**
Sync your system to PACER to automate legal marketing.