

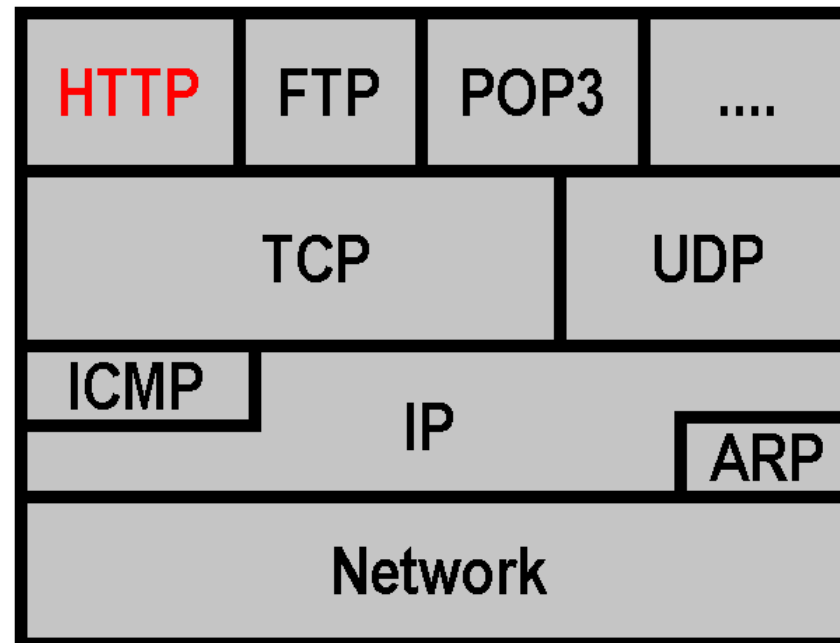
Web Server for Embedded Systems

KLAUS-D. WALTER
SSV EMBEDDED SYSTEMS
HEISTERBERGALLEE 72
D-30453 HANNOVER
WWW.SSV-EMBEDDED.DE



TCP/IP Protocol Basics

* Networking technology is organized in layers. The base is the OSI Reference Model. Each layer only communicates with the layers immediately above or below it.

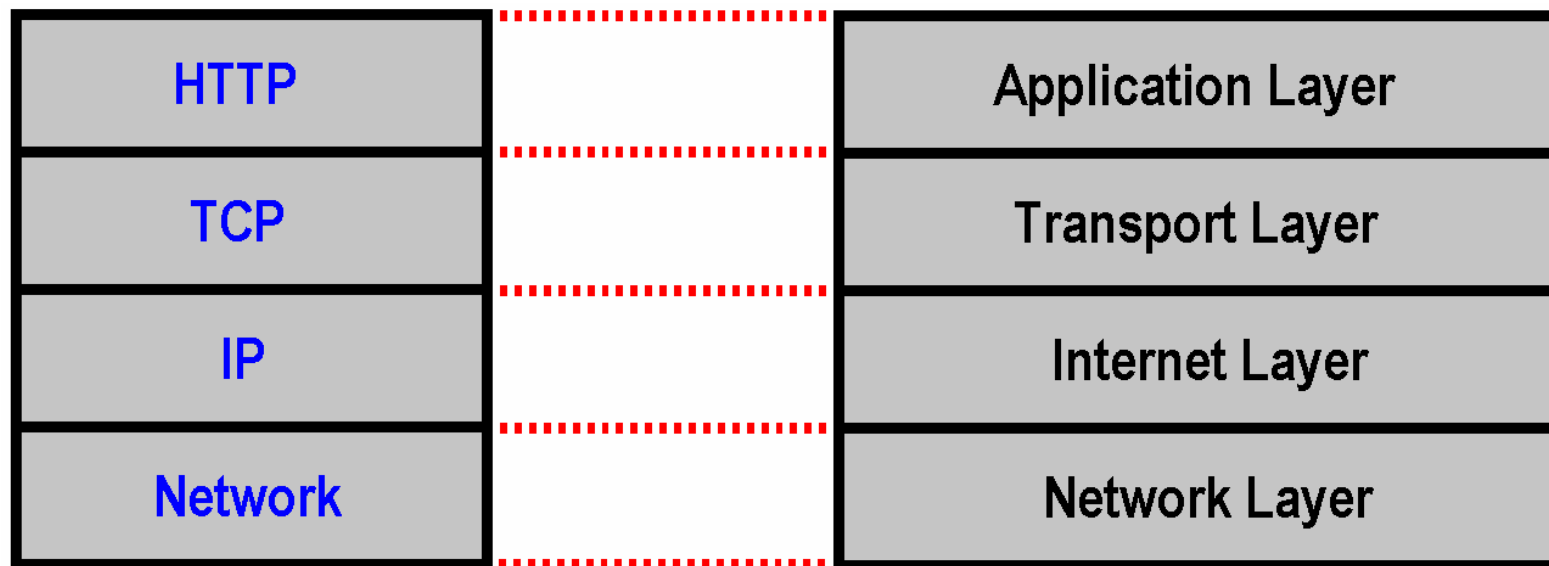


* HTTP define the rules by which Web browsers, Web servers, proxies, and other Web systems establish and maintain communications with each other.



TCP/IP Protocol Basics

* The OSI Reference Model is based on seven layers. HTTP (and a TCP/IP protocol stack) is using only four protocol layers within a computer system.

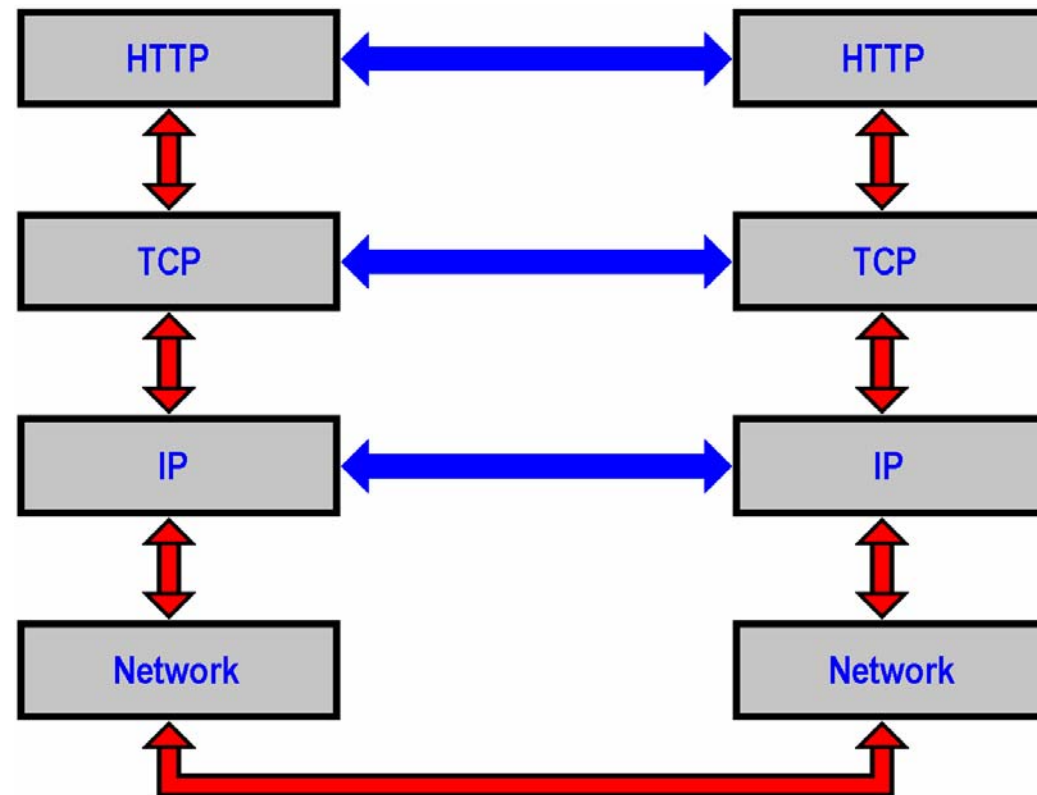


* The lowest layer is the **network layer**. The protocol layer above the network layer is the **Internet Protocol (IP)**. Next is the **Transmission Control Protocol (TCP)**. The final protocol layer is **HTTP**, a TCP/IP application protocol.



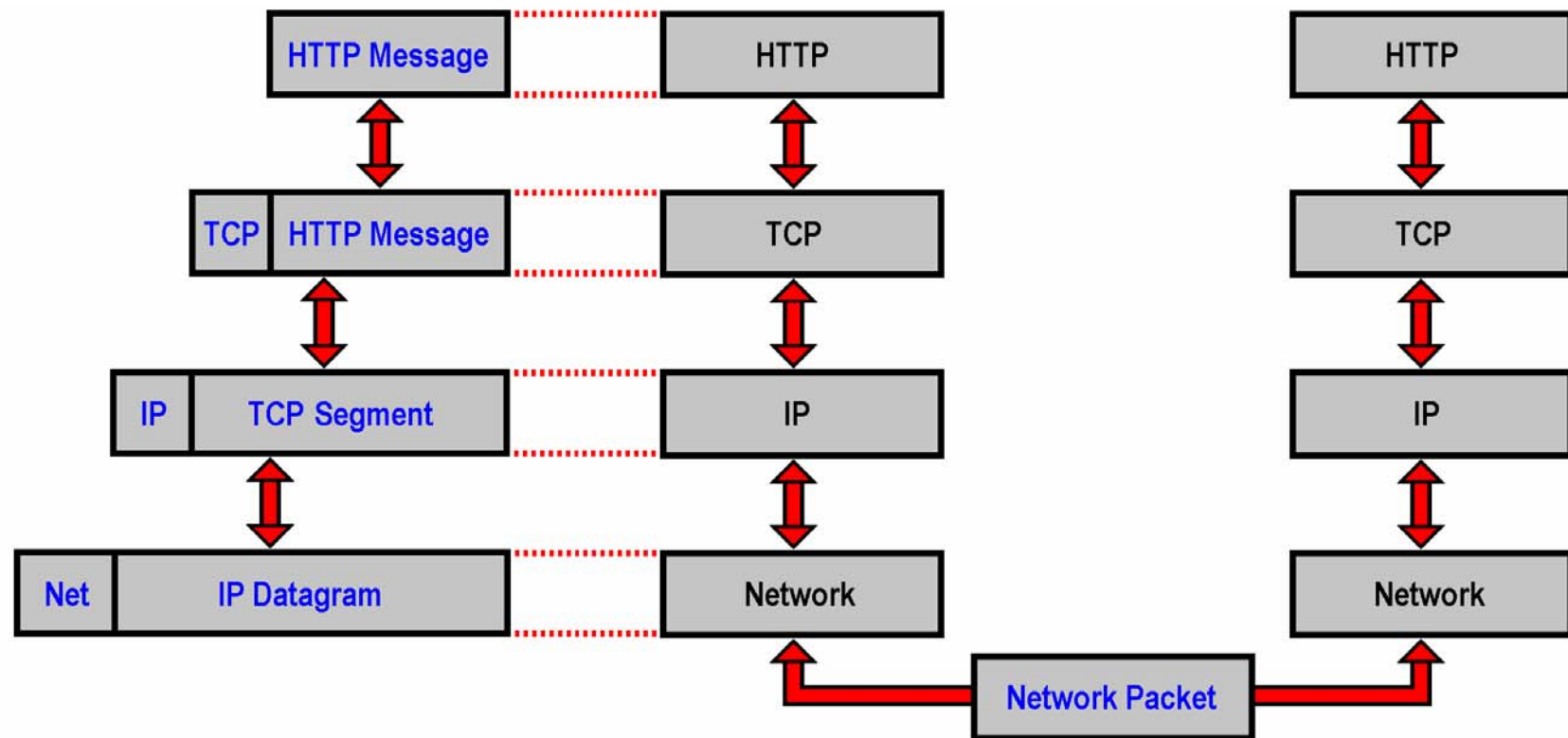
TCP/IP Protocol Basics

- * There is a logical and physical communication between each protocol layer. HTTP reads or writes data from/to TCP. That protocol interacts directly with IP. IP interacts with the protocol controlling the network layer (i.e. Ethernet or PPP).



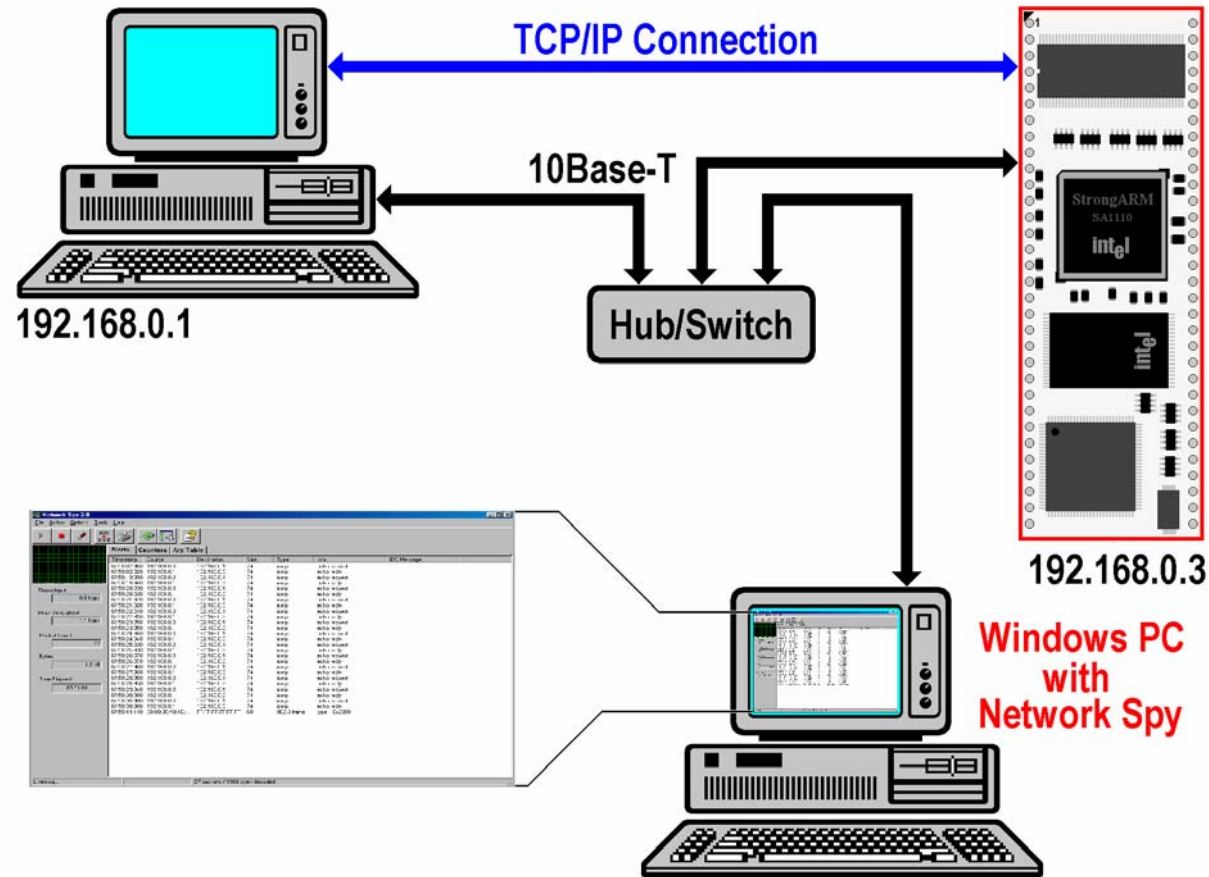
TCP/IP Protocol Basics

* A protocol is using a own header and a own name for the unit of data it sends and receives. Each protocol layer adds and removes it's own specific information.



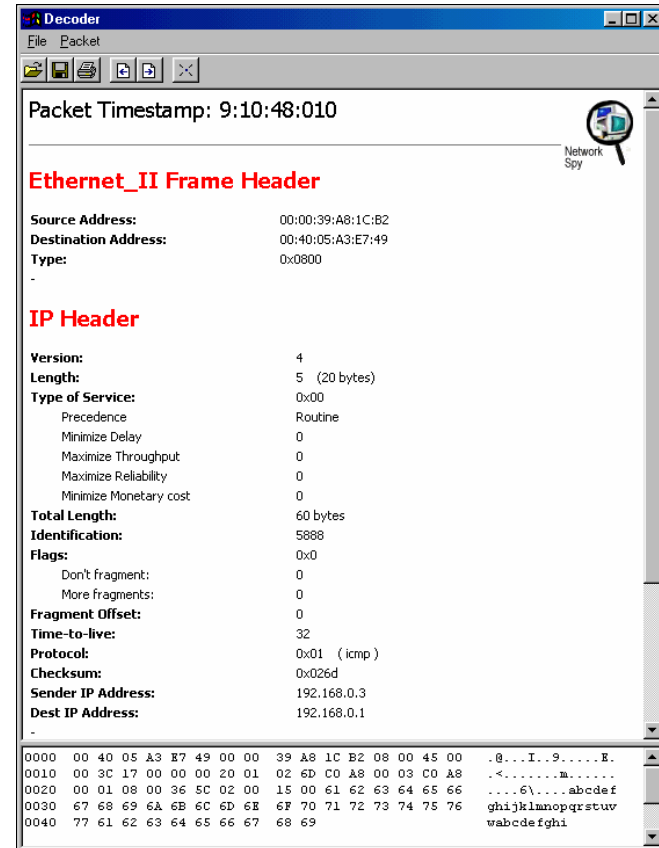
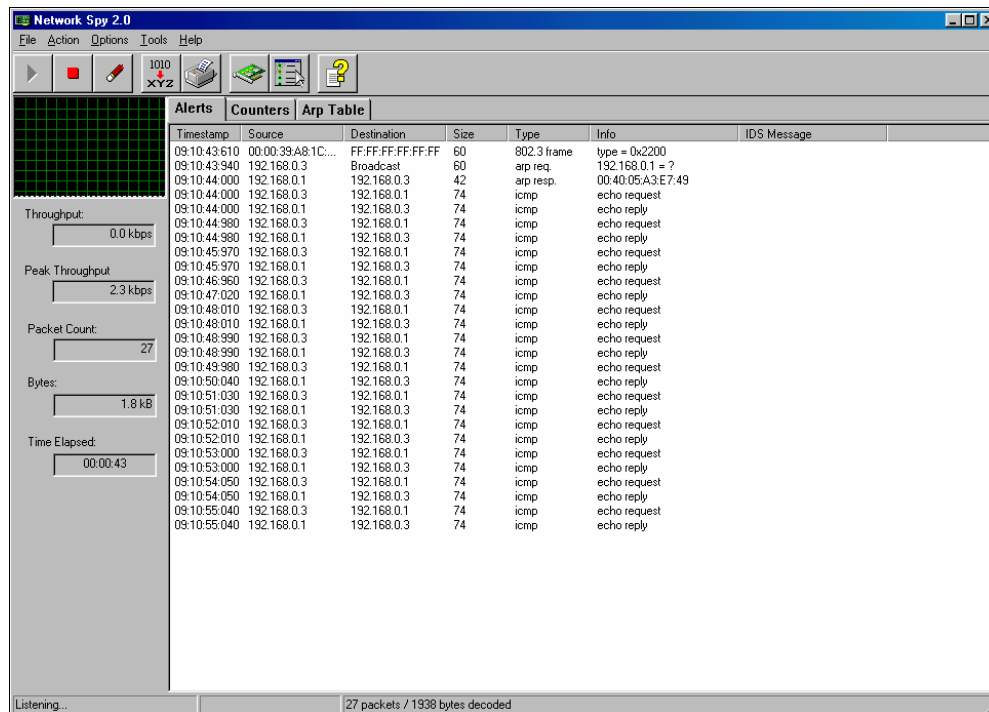
Debugging TCP/IP Connections

* Use a network monitor program (i.e. Ethernet sniffers) . This kind of programs enables you to capture and examine network packets.



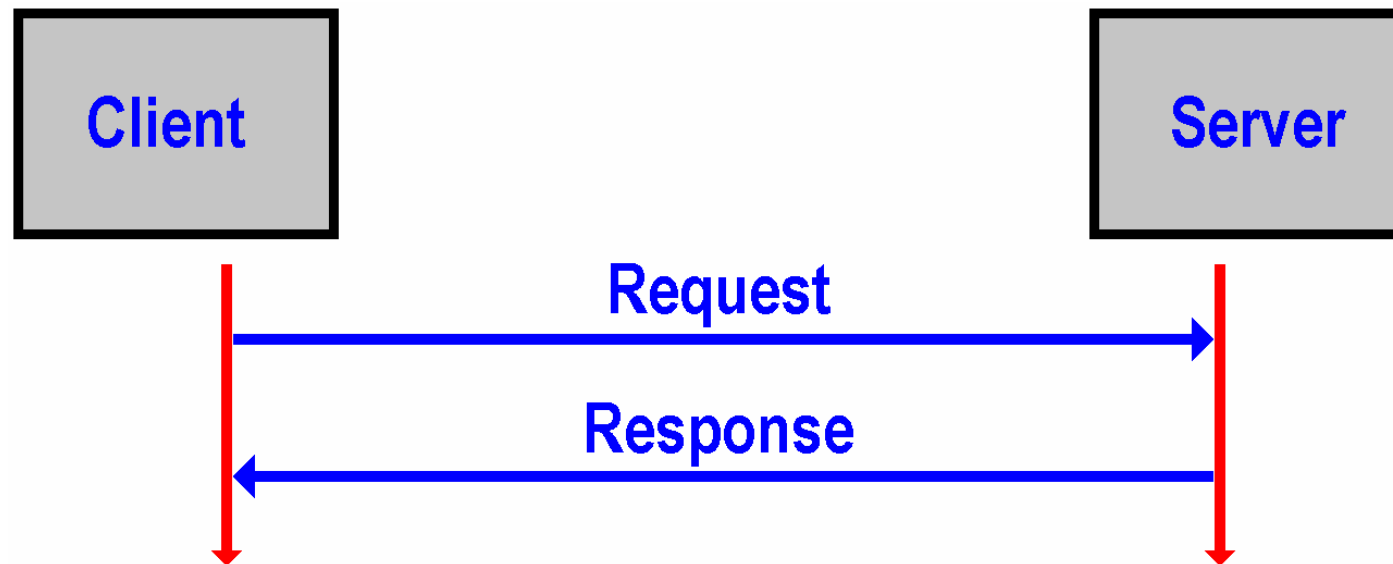
Debugging TCP/IP Connections

* Most network monitor programs comes complete with a suite of TCP/IP protocol parsers and decoders. With the aid of these tools, captured traffic can be analyzed.



HTTP Protocol Basics

- * HTTP follows client/server rules and procedures. The main difference between HTTP clients and servers is the responsibility for initiating communication. Only a client can do that.

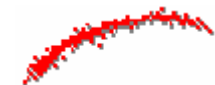
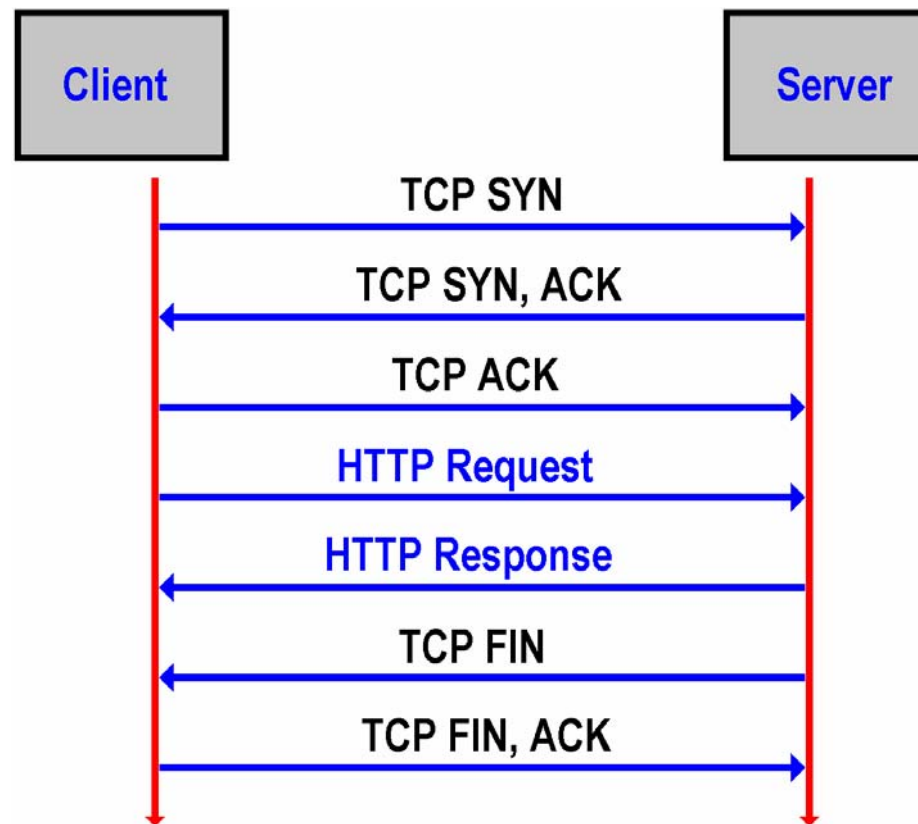


- * The server waits for a client request and reacts with a response. Typical the Web browser, in it's rule of client, send a HTTP request to a server. The server returns a HTTP response.



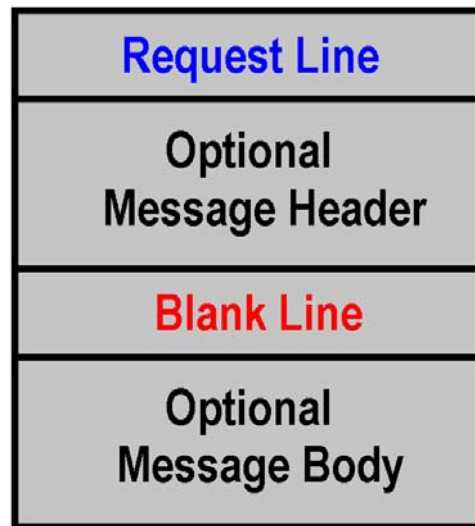
HTTP Protocol Basics

* HTTP requires a TCP connection. The client is responsible for initiating the HTTP communication to the server. After the TCP connection is established the client can send a HTTP request.

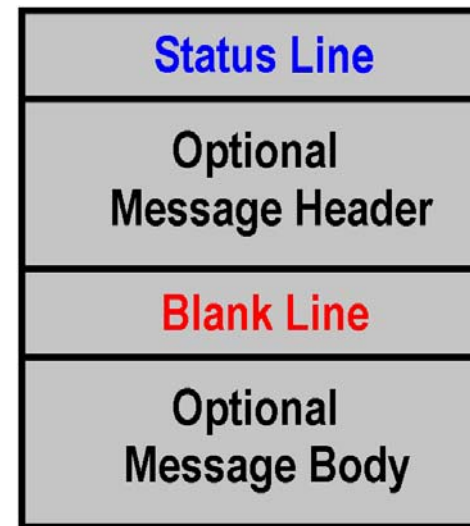


HTTP Protocol Basics

- * The structure of HTTP messages is very simple. Each request begins with a request line. This text line indicates the HTTP method, the resource, and the HTTP version. Optional message header and body follows.



HTTP Request



HTTP Response

- * The response begins with a status line. This text line starts with the HTTP version that the server supports. Then a status code follows within the status line. A optional header and body follows with the response.



HTTP Protocol Basics (HTTP Methods)

- * **CONNECT**: Asks (proxy) server to establish a tunnel.
- * **DELETE**: Asks server to delete the indicated resource.
- * **GET**: Asks server to return requested resource.
- * **HEAD**: Asks server for header information about a special resource.
- * **OPTIONS**: Asks server to indicate the options it supports for a resource.
- * **POST**: Asks server to pass the message body to a indicated resource.
- * **PUT**: Asks server to accept the message body as the indicated resource.
- * **TRACE**: Asks server simply to respond to the request.



HTTP Protocol Basics (HTTP Status Codes)

- * **100 - 199**: Informational. The server received the client request but the final result is not yet available.
- * **200 - 299**: Success. The server was able to act on the client request successfully.
- * **300 - 399**: Redirection. The client should redirect the request to a different server or resource.
- * **400 - 499**: Client Error. The request contained an error that prevented the server from acting on it successfully.
- * **500 - 599**: Server Error. The server failed to act on a request even though the request appears to be valid.



HTTP Protocol Basics

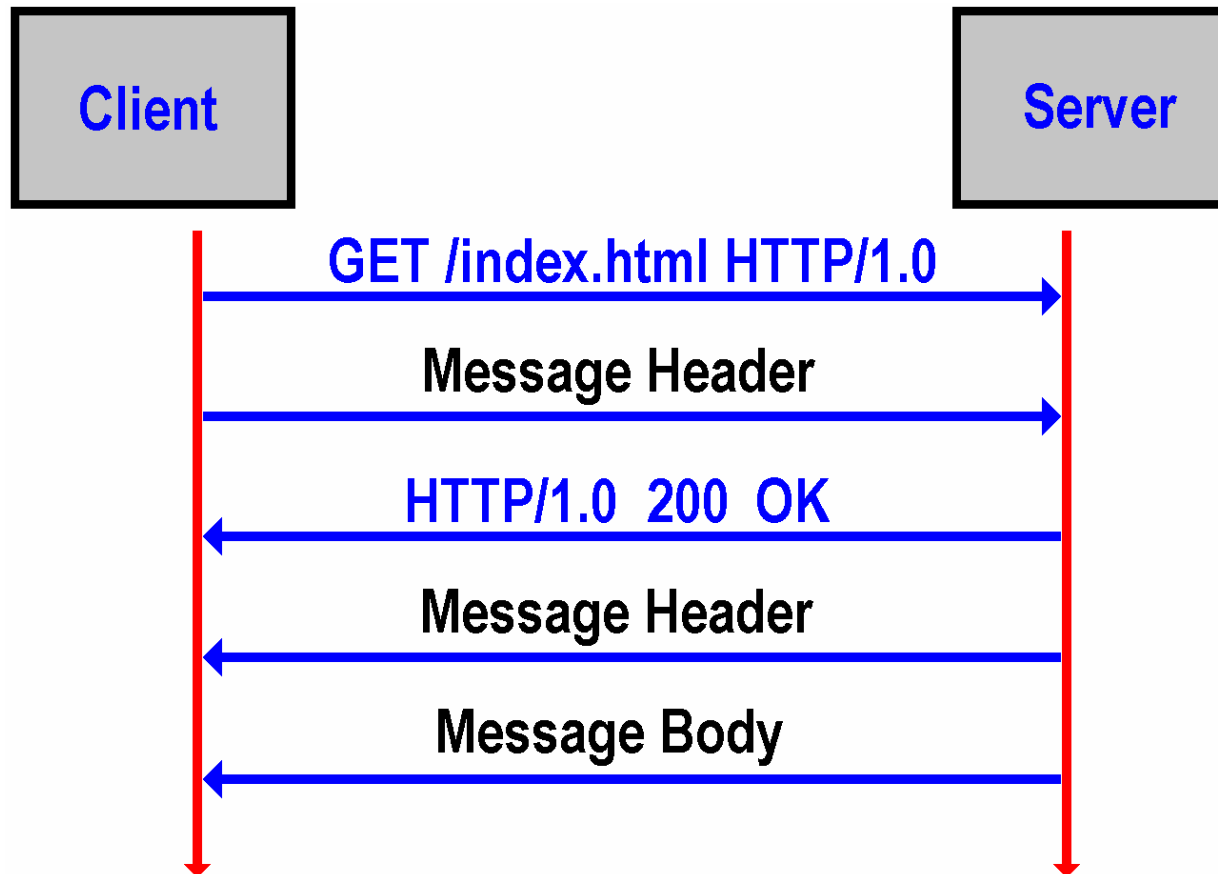
* The message header contains with **Content-type** the MIME (Multipurpose Internet Mail Extension) type of the message body.

<code>text/plain</code>	ASCII text
<code>text/html</code>	HTML formatted text
<code>application/pdf</code>	Document formatted in Adobe PDF
<code>image/gif</code>	Image encoded in GIF format
<code>image/jpeg</code>	Image encoded in JPEG format
<code>audio/basic</code>	Sound file
<code>video/mpeg</code>	Video clip coded in MPEG format



HTTP Protocol Basics

* A typical GET request consist of a request line and a message header. The server returns the status line, a message header and the requested resource.



HTTP Protocol Basics

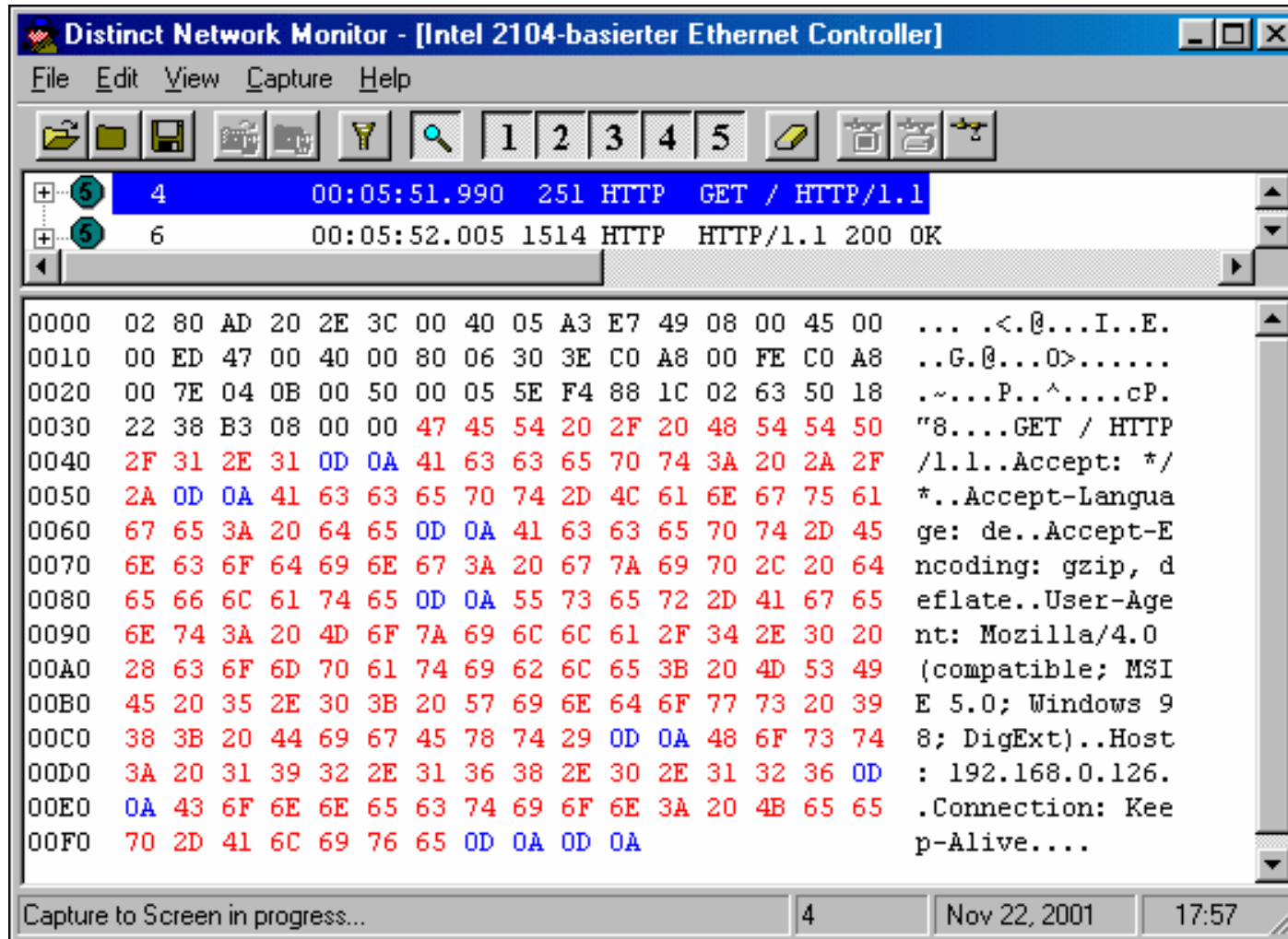
```
GET /test.htm HTTP/1.1
Accept: image/gif, image/jpeg, */*
User selling agent: Mozilla/4.0
Host: 192.168.0.1
```

```
HTTP/1.1 200 OK
Date: Mon, 06 Dec 1999 20:55:12 GMT
Server: Apache/1.3.6 (Linux)
Content-length: 82
Content-type: text/html
```

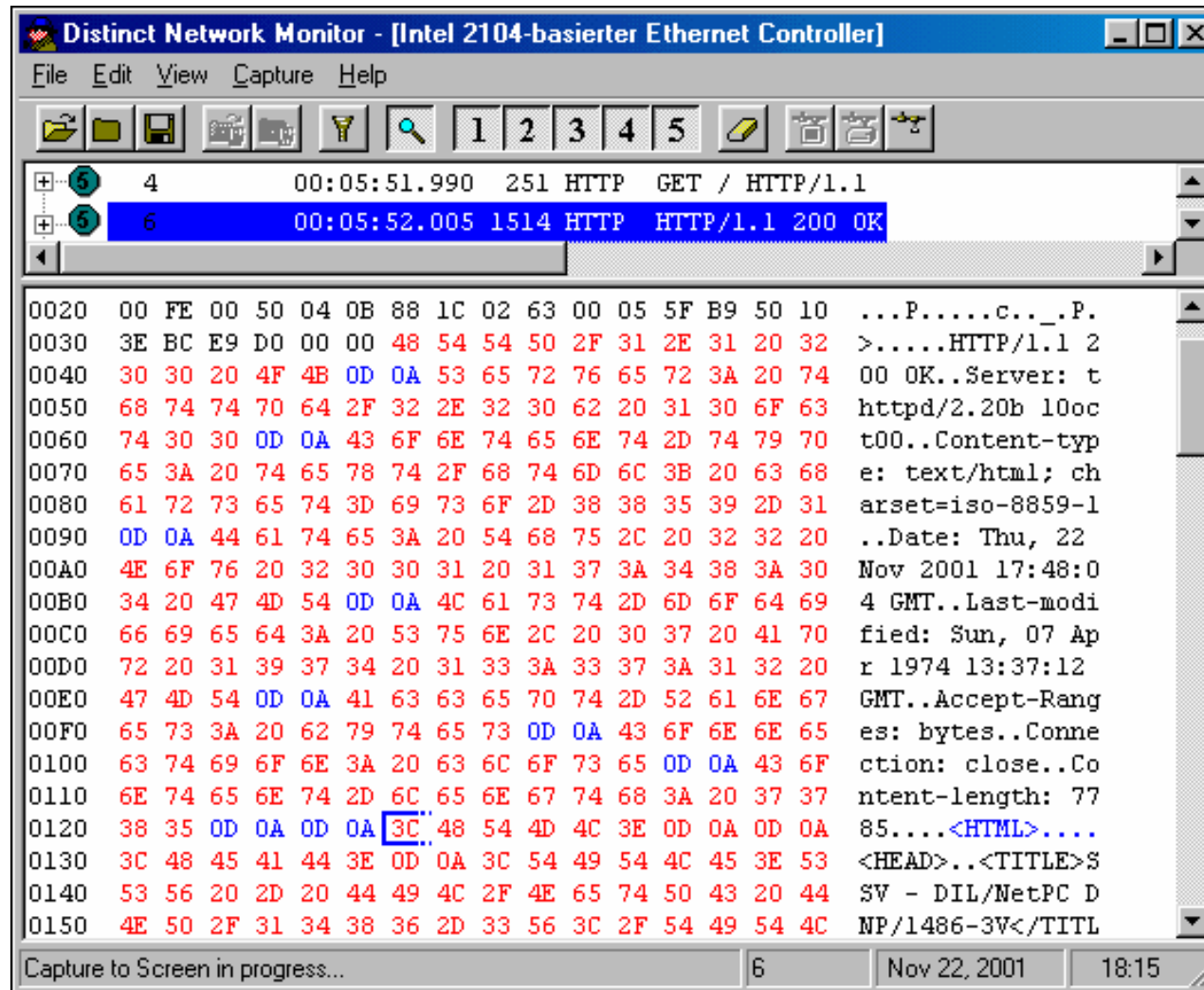
```
<html>
<head>
<title>HTML Test Page</title>
</head>
<body>
HTML Test Page
</body>
</html>
```



HTTP Protocol Basics



HTTP Protocol Basics

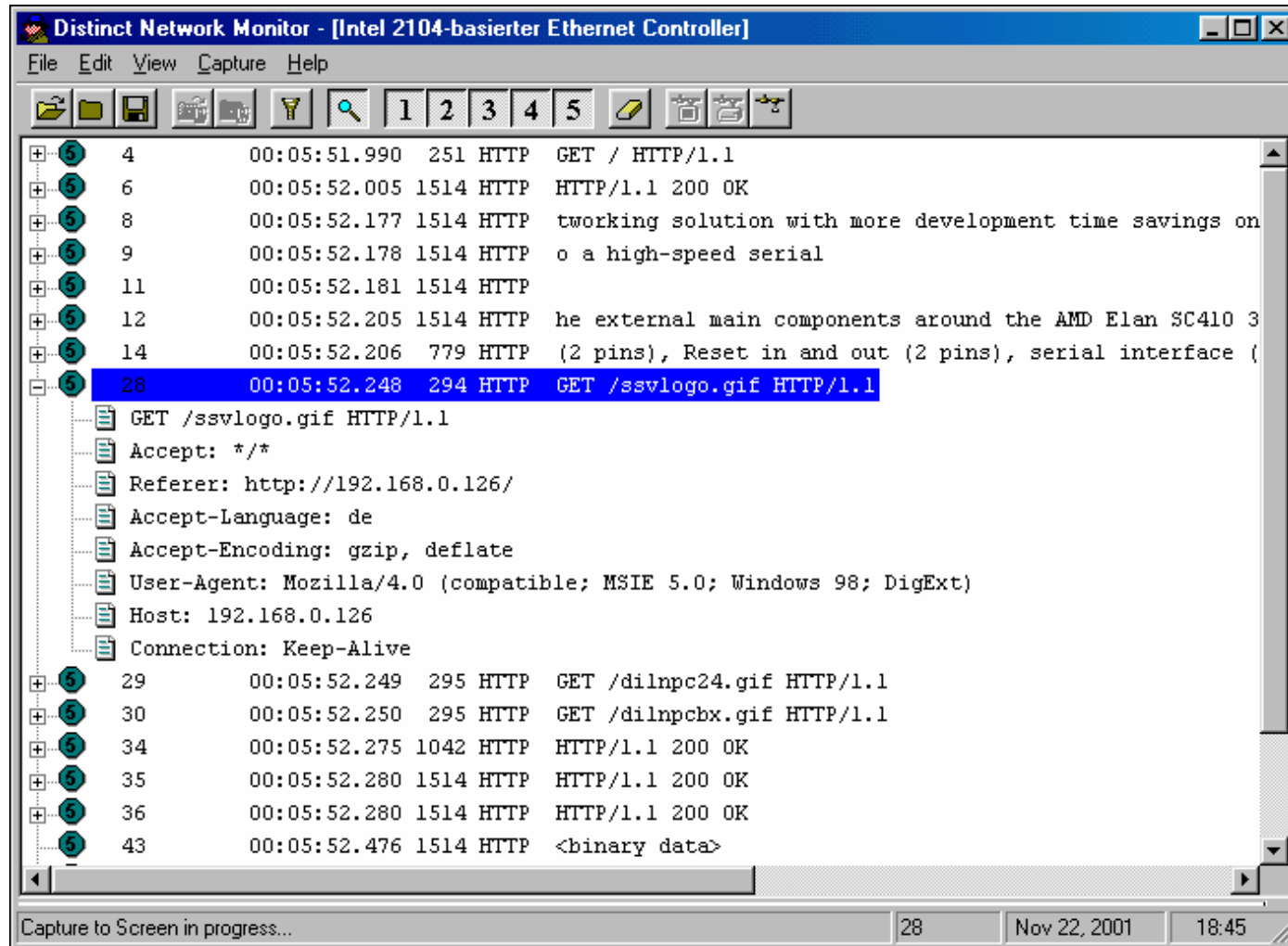


The screenshot shows the Distinct Network Monitor interface. The main window displays a list of captured packets. Packet 6 is selected, showing an HTTP GET request and its response. The response is displayed in hexadecimal and ASCII format.

Packet No.	Time	Length	Protocol	Details
4	00:05:51.990	251	HTTP	GET / HTTP/1.1
6	00:05:52.005	1514	HTTP	HTTP/1.1 200 OK

Hex	ASCII
0020 00 FE 00 50 04 0B 88 1C 02 63 00 05 5F B9 50 10	...P.....c...P.
0030 3E BC E9 D0 00 00 48 54 54 50 2F 31 2E 31 20 32	>.....HTTP/1.1 2
0040 30 30 20 4F 4B 0D 0A 53 65 72 76 65 72 3A 20 74	00 OK..Server: t
0050 68 74 74 70 64 2F 32 2E 32 30 62 20 31 30 6F 63	httpd/2.20b 10oc
0060 74 30 30 0D 0A 43 6F 6E 74 65 6E 74 2D 74 79 70	t00..Content-typ
0070 65 3A 20 74 65 78 74 2F 68 74 6D 6C 3B 20 63 68	e: text/html; ch
0080 61 72 73 65 74 3D 69 73 6F 2D 38 38 35 39 2D 31	arset=iso-8859-1
0090 0D 0A 44 61 74 65 3A 20 54 68 75 2C 20 32 32 20	..Date: Thu, 22
00A0 4E 6F 76 20 32 30 30 31 20 31 37 3A 34 38 3A 30	Nov 2001 17:48:0
00B0 34 20 47 4D 54 0D 0A 4C 61 73 74 2D 6D 6F 64 69	4 GMT..Last-modi
00C0 66 69 65 64 3A 20 53 75 6E 2C 20 30 37 20 41 70	fied: Sun, 07 Ap
00D0 72 20 31 39 37 34 20 31 33 3A 33 37 3A 31 32 20	r 1974 13:37:12
00E0 47 4D 54 0D 0A 41 63 63 65 70 74 2D 52 61 6E 67	GMT..Accept-Rang
00F0 65 73 3A 20 62 79 74 65 73 0D 0A 43 6F 6E 6E 65	es: bytes..Conne
0100 63 74 69 6F 6E 3A 20 63 6C 6F 73 65 0D 0A 43 6F	ction: close..Co
0110 6E 74 65 6E 74 2D 6C 65 6E 67 74 68 3A 20 37 37	ntent-length: 77
0120 38 35 0D 0A 0D 0A 3C 48 54 4D 4C 3E 0D 0A 0D 0A	85....<HTML>....
0130 3C 48 45 41 44 3E 0D 0A 3C 54 49 54 4C 45 3E 53	<HEAD>..<TITLE>S
0140 53 56 20 2D 20 44 49 4C 2F 4E 65 74 50 43 20 44	SV - DIL/NetPC D
0150 4E 50 2F 31 34 38 36 2D 33 56 3C 2F 54 49 54 4C	NP/1486-3V</TITL

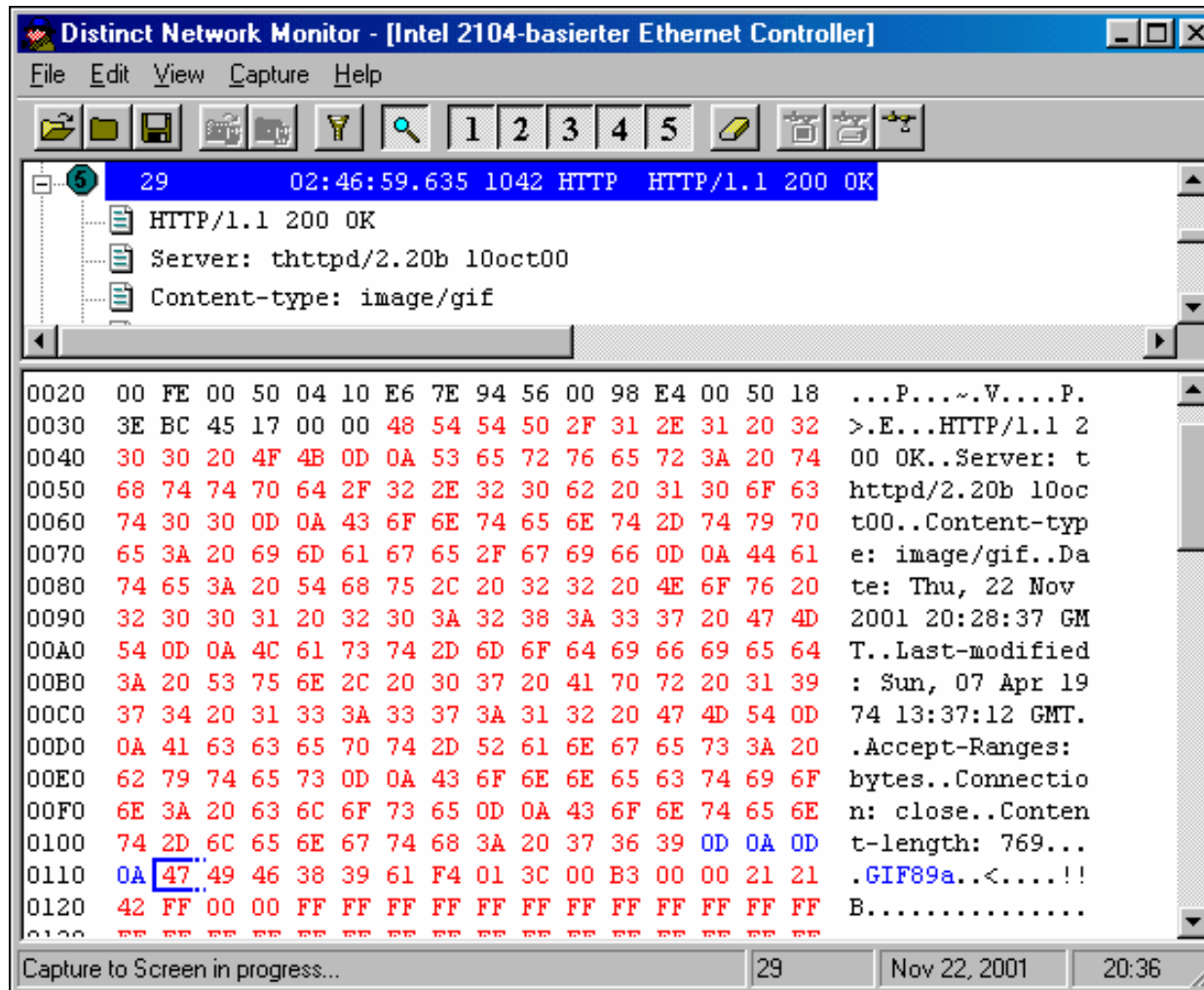
HTTP Protocol Basics



The screenshot shows a network monitor window titled "Distinct Network Monitor - [Intel 2104-basierter Ethernet Controller]". The interface includes a menu bar (File, Edit, View, Capture, Help) and a toolbar with various icons. The main display area shows a list of network packets. Packet 28 is highlighted in blue and shows an HTTP GET request for "/ssvlogo.gif". Below this packet, the details of the request are expanded, showing headers such as "Accept: /*/*", "Referer: http://192.168.0.126/", "Accept-Language: de", "Accept-Encoding: gzip, deflate", "User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)", "Host: 192.168.0.126", and "Connection: Keep-Alive". The status bar at the bottom indicates "Capture to Screen in progress...", packet number 28, date Nov 22, 2001, and time 18:45.

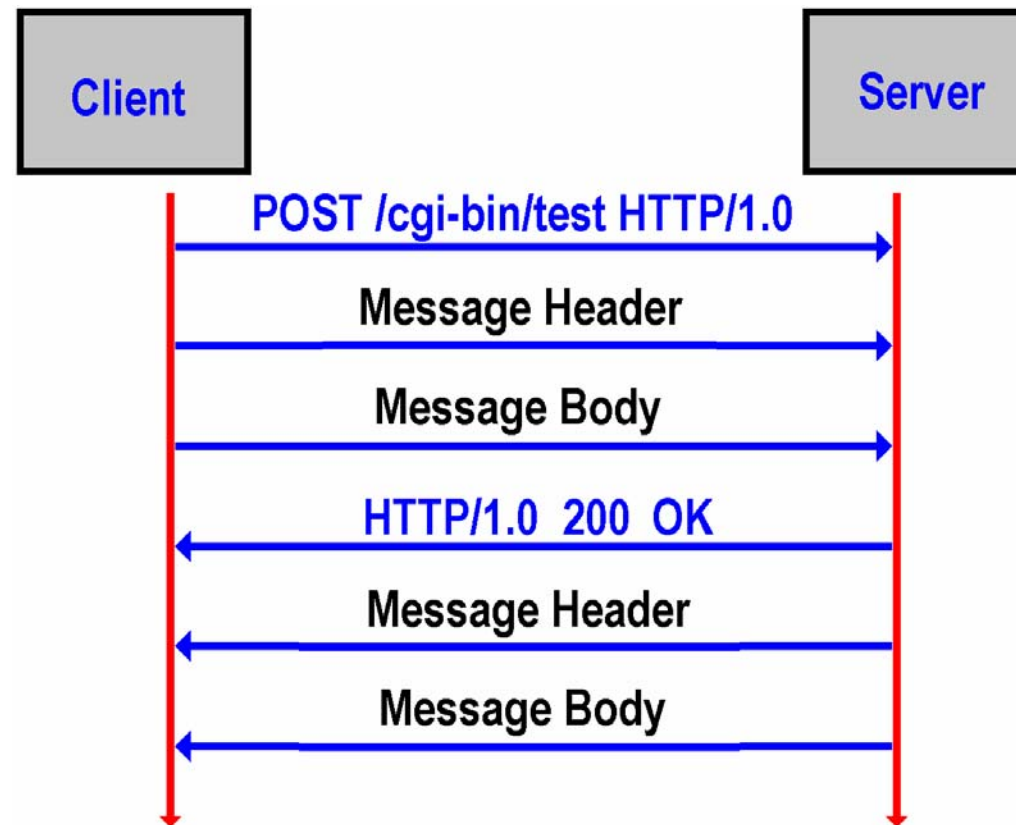
Packet No.	Time	Length	Protocol	Details
4	00:05:51.990	251	HTTP	GET / HTTP/1.1
6	00:05:52.005	1514	HTTP	HTTP/1.1 200 OK
8	00:05:52.177	1514	HTTP	two working solution with more development time savings on
9	00:05:52.178	1514	HTTP	o a high-speed serial
11	00:05:52.181	1514	HTTP	
12	00:05:52.205	1514	HTTP	he external main components around the AMD Elan SC410 3
14	00:05:52.206	779	HTTP	(2 pins), Reset in and out (2 pins), serial interface (
28	00:05:52.248	294	HTTP	GET /ssvlogo.gif HTTP/1.1
29	00:05:52.249	295	HTTP	GET /dilnpc24.gif HTTP/1.1
30	00:05:52.250	295	HTTP	GET /dilnpcbx.gif HTTP/1.1
34	00:05:52.275	1042	HTTP	HTTP/1.1 200 OK
35	00:05:52.280	1514	HTTP	HTTP/1.1 200 OK
36	00:05:52.280	1514	HTTP	HTTP/1.1 200 OK
43	00:05:52.476	1514	HTTP	<binary data>

HTTP Protocol Basics



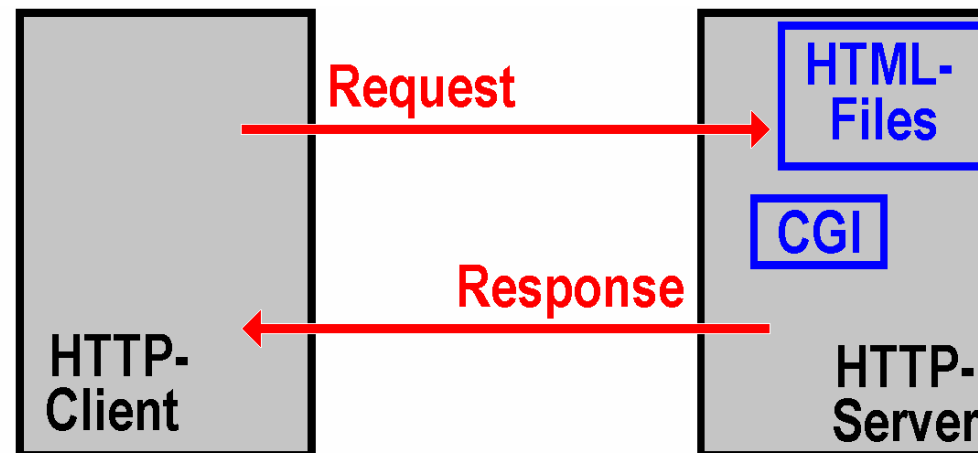
HTTP Protocol Basics

* A POST request consist of a request line, a message header, and a message body. The server returns status line, message header, and the output of the CGI program within the message body.



Web Server Basics

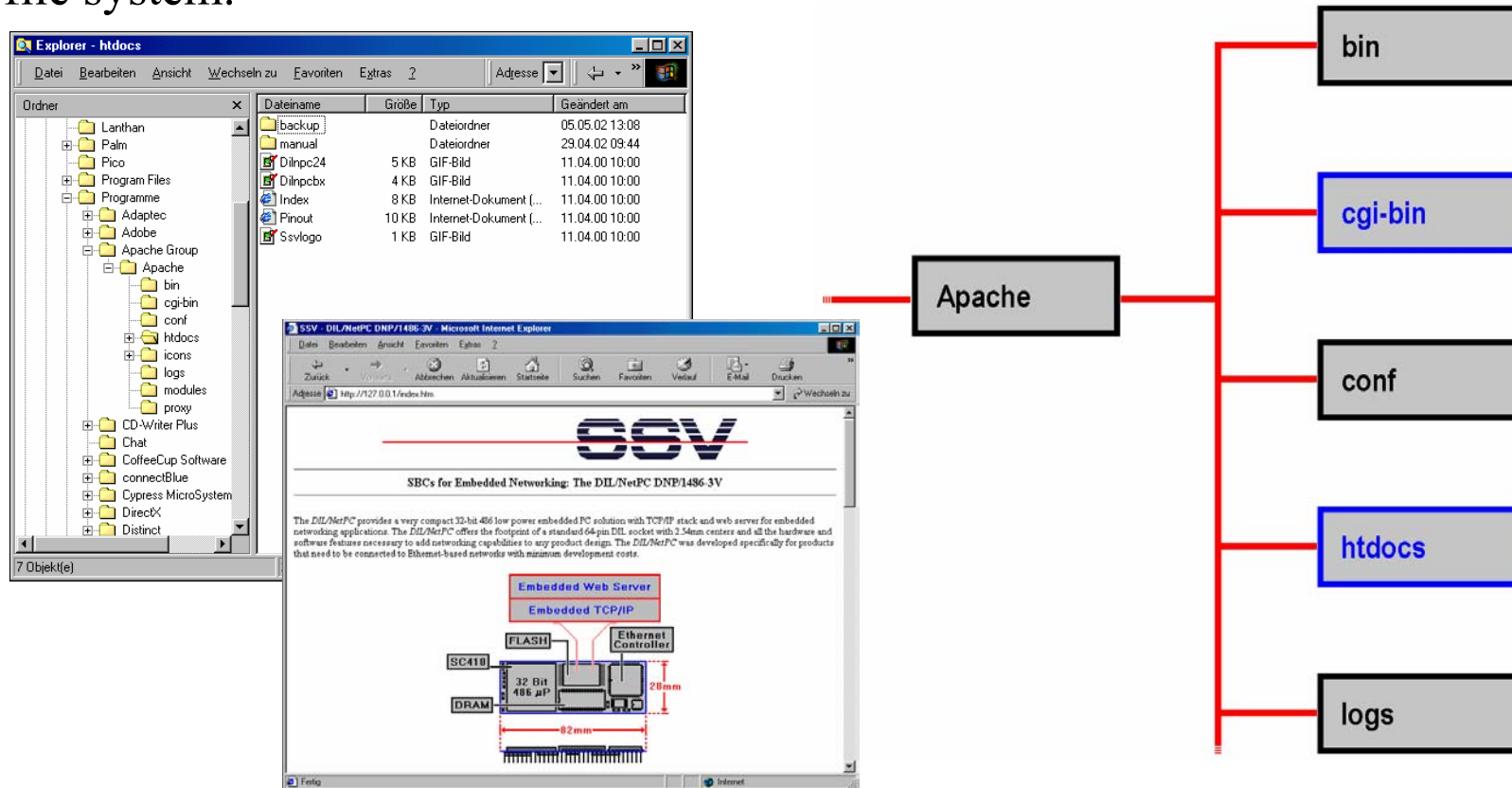
* Simplified a Web server can be imagined like a special kind of a file server. The “files” are the resources (static content: HTML files, GIF and JPEG pictures ...). The Web server deliver these resources with a very special communication protocol (HTTP).



* A Web server can not only deliver static content. There are different technologies (CGI, Server-Side-Includes) for generate dynamic content.

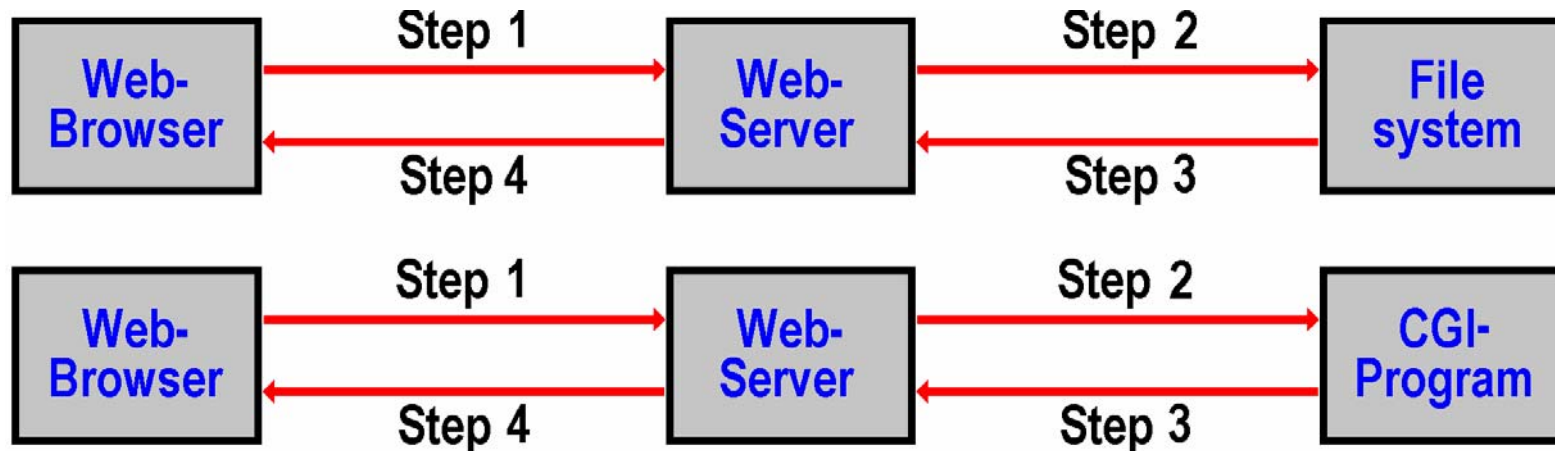
Web Server Basics

* Web server needs storage space for the resources (HTML files, GIF and JPEG pictures ...). Typical these resources are stored within special directories of a file system.



Web Server Basics

* With step 1 the browser sends a GET request for a specific resource to the Web server. Then (step 2) the server try to read this resource. If the resource accessible, the server reads the resource (step 3) and deliver the content to the Web browser (step 4).

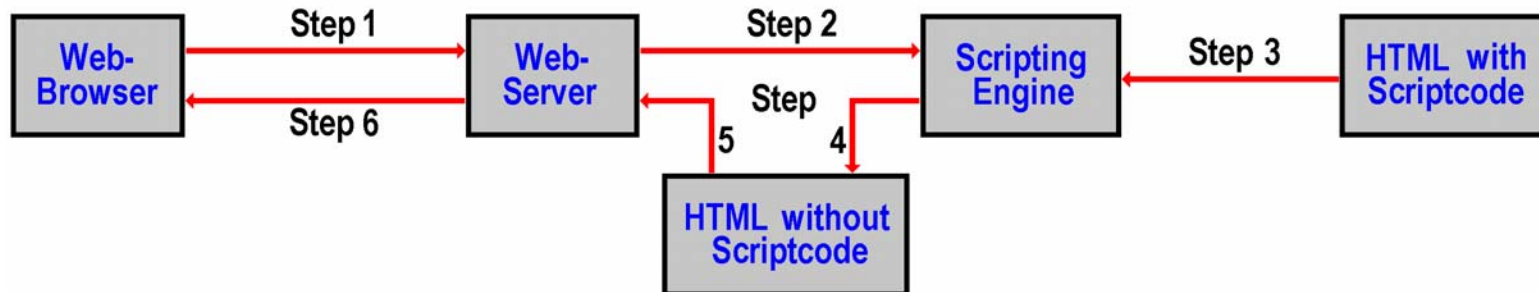


* If the GET request points to **cgi-bin**, then the server runs the indicated program (step 2) and receives some output from this program (step 3). The outputs goes with the HTTP response to the Web browser.



Web Server Basics

* A other way for generating dynamic content is SSI (Server-Side-Include). The basic idea of this technology is to include script language statements to HTML pages. For interpreting the script statements the Web server needs the help of a **Scripting Engine** (i.e. PHP interpreter).



* Step 1 delivers the HTTP request to the Web server. Then the server starts the Scripting Engine (step 2). This program reads the requested HTML page and parses and executes the embedded script code (step 3).

* The output of a Scripting Engine is HTML without script code (step 4) direct to the Web server (step 5). With step 6 the Web server deliver the script-free HTML page to the Web browser.

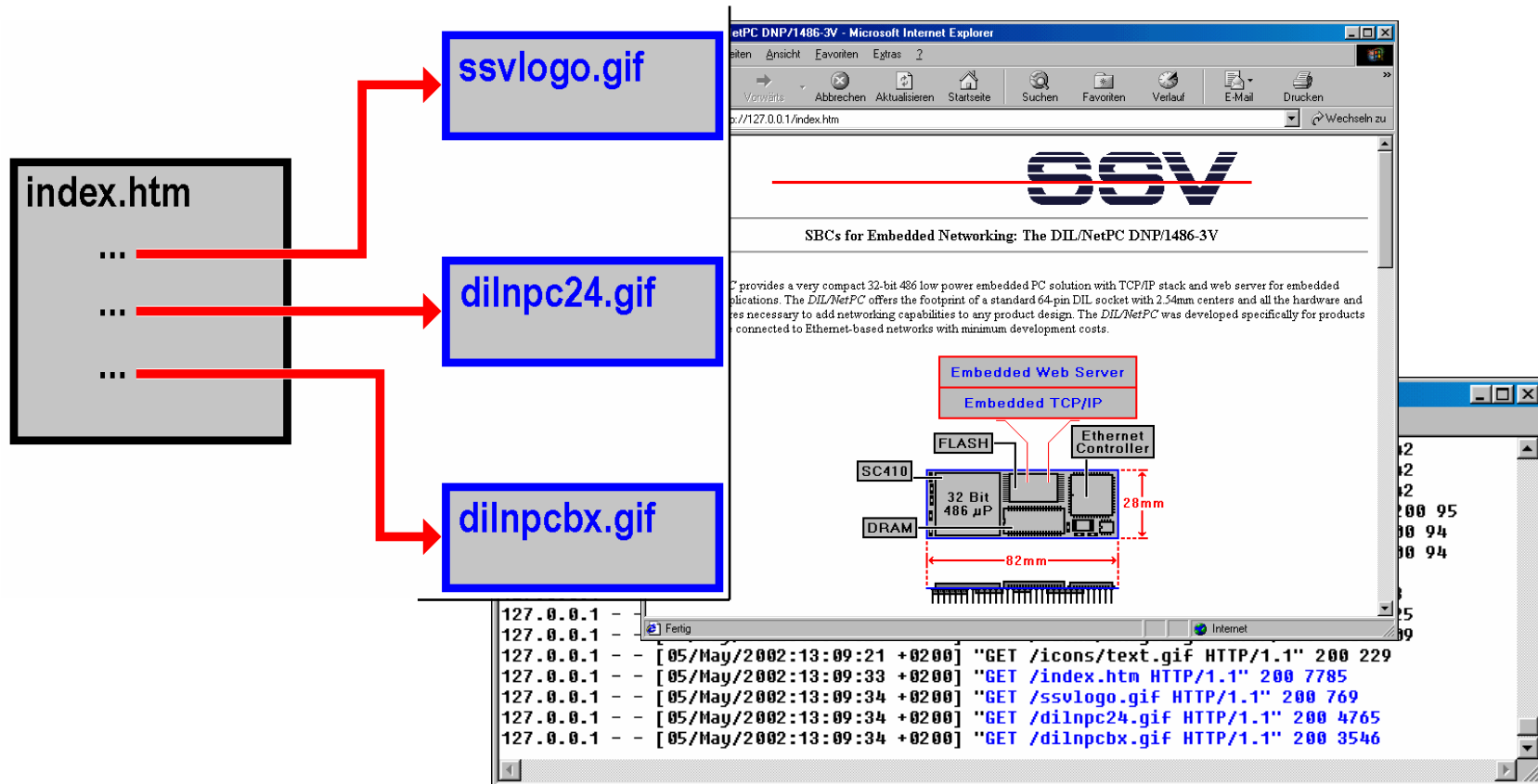


Stay Connected



Web Server Basics

* Web resources are linked together. A browser parses the HTML file direct after the first GET. Then the browser issues a additional GET for each (hyper) link.

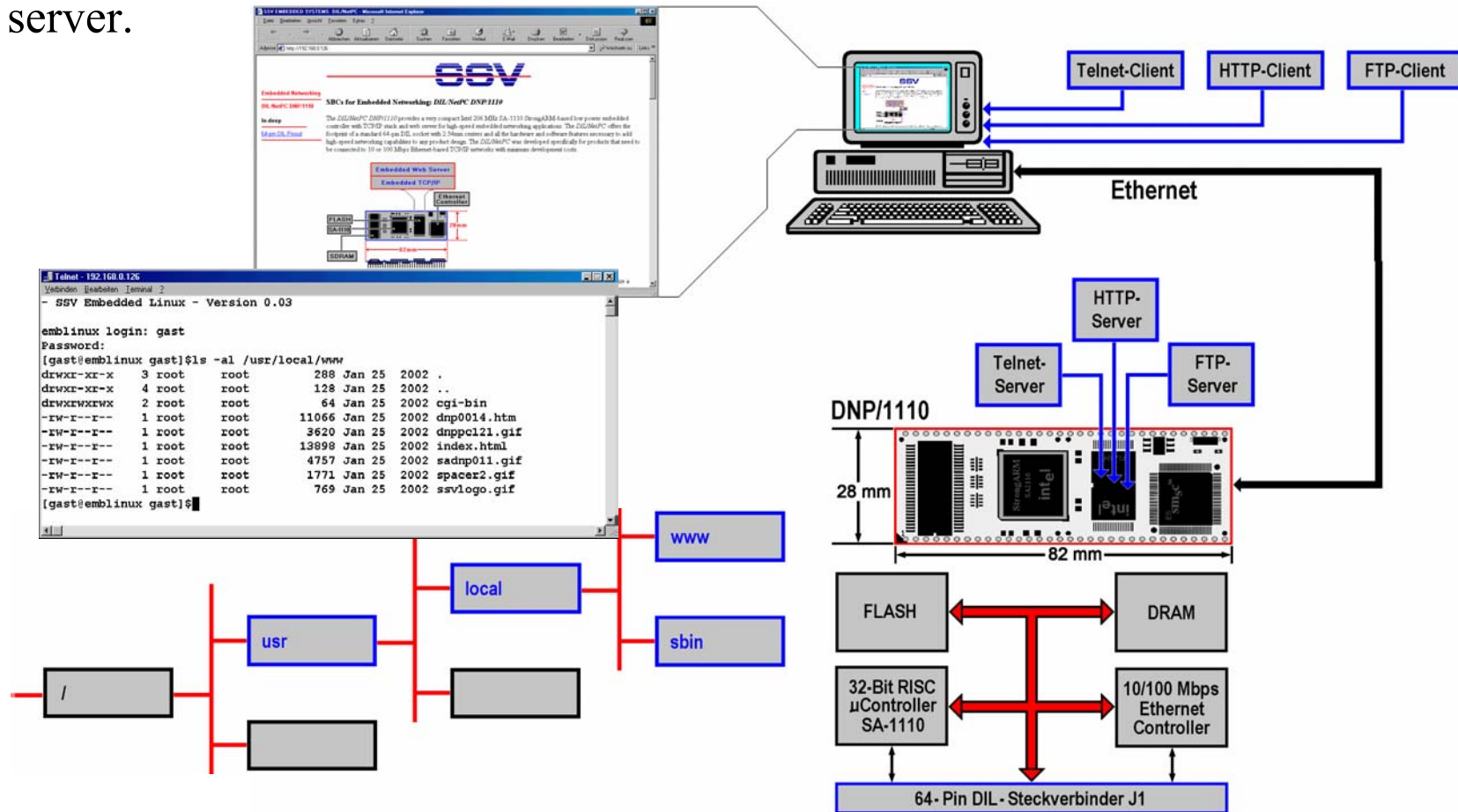


Stay Connected



Solution Demonstration

* DIL/NetPC DNP/1110 with embedded Linux and **thttpd** (Open Source) Web server.



Trademarks

Microsoft, Windows, Win32, Windows 95, and Windows NT are either trademarks or registered trademarks of Microsoft Corporation. Linux is a trademark of Linus Torvalds. DIL/NetPC is a trademark of SSV GmbH, Hannover. Copyright for all pictures and diagrams by Klaus-D. Walter.

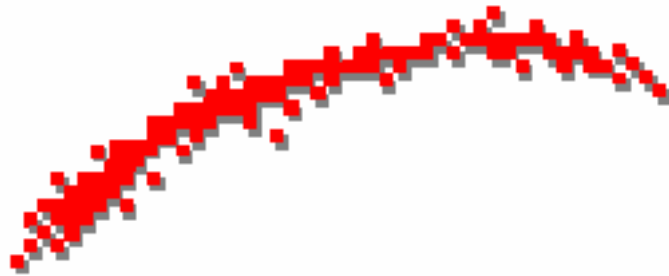
All other brand and product names are trademarks or registered trademarks of their respective holders.

Disclaimer

Information in this document is subject to change without notice. No part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means electronic or mechanical, including photocopying and recording for any purpose without prior written permission from SSV GmbH, Hannover, Germany.



 Stay Connected



**SSV EMBEDDED SYSTEMS
HEISTERBERGALLEE 72
D-30453 HANNOVER
TEL.: +49-(0)511-40000-0
FAX.: +49-(0)511-4000040
WEB: WWW.SSV-EMBEDDED.DE
EMAIL: INFO@IST1.DE**

File: WsforES1.ppt Revision: 1.1 - 11.05.2002

