A Brief Introduction of the Web++ Framework

Bing Swen (Sun Bin)
Dept. of Computer Science Technology, Peking University, Beijing 100971, CHINA
bswen@cs.pku.edu.cn

Abstract: This paper presents an overview of the Web++ framework, a new mechanism of hypertext resource transmission specifically designed to further improve Web performance. The major components of the framework are briefly described, along with some initial and yet encouraging results from experimental implementation and test.
Keywords: World-Wide Web, HTTP, Performance, Hypertext, Resource transmission

1 Introduction

The simple idea – "batch-fetching" a web page and all of its related objects (such as images, scripts, applets, style sheets, etc.) within a single request and response – is not new, and several previous proposals were made quite a few years ago, e.g., a primitive MGET method by Franks [2], or the GETALL and GETLIST methods by Padmanabhan and Mogul [3], and later the "collection resource" of WebDAV [7], MHTML by Palme and Hopmann [5], and the recent "bundle" proposal by Wills et al [1].

This paper presents an overview of the Web++ framework, yet another mechanism to transfer web resources in a batch manner. The major difference between Web++ and the previous work may be that the former is a little bit more further to try to provide a complete solution to the problem. The major insufficiency of the previous proposals seems to be in the difficulty to handle various partial modifications of embedded objects. It would be exceedingly difficult (if not impossible) to design a uniform and consistent scheme of aggregate resource updating without the help of an elaborate information description of individual resources, and we are not aware of any other work that uses a special transfer encoding together with a transfer control mechanism to speed up HTTP transactions.

Actually, the structural characteristics of "hypertexted Web pages" still provide a great potential for performance improvement. This research is intended to explore such a possibility.

2 Web++ Overview

The framework of Web++ includes three components[8,9]:

- A new URL scheme *sttp* for identifying resources on the Web++, with the general format *sttp: // host : port / path ? parameters*.

- The Structured Hypertext Transfer Protocol (STTP), defining a message set of requests and responses for the transmission control of resources on the Web++.

- The Structured Hypertext Markup Language (STML), for describing the structural information of Web pages, including information of the root page file, number and types of the linked objects, entity attributes of each object, file offsets and sizes of partial update, etc.

Roughly speaking, an STML document is a "hypertext of hypertext", that is, a set of hypermedia objects that are related to the same root hypertext. (The set may or may not be "closed" with respect to the closure of links.) Thus STTP may also be called the protocol for the transmission of a set of hypertexts. Based on the detailed meta-information described in STML, STTP can possibly transfer resources in an optimal way. Before sending a page file to the client, the server first processes the page into a more compact format (*structured hypertext*) with its header containing sufficient meta-information of each element related to the page, so that the client can handle them directly, without any repeated network transmission. On the other hand, the client also presents sufficient meta-information about its desired objects to the server for optimization of the delivery. Such processing of Web page allows the server and client to have a good knowledge of the contents that are transmitted, and helps make a more efficient use of TCP connection.

Using such description, typical STTP Transactions can be performed within one request and one reply. E.g., when a client is to retrieve a Web page that is not locally cached, it tries to get the page by sending a single *selective S-GET* request, expecting a single response from the server with the message body being a full STML document generated for the page. If the page is already cached, then the client generates a partial STML document (*head-part*) listing the meta-information of all the interesting objects related to the page (including the root page itself) obtained since the last visit, and send an *S-COMPARE* request, expecting a single response with an STML document containing all the necessary information of update for modified objects. This would provide the most efficient Web page retrieval model. For a typical Web page with 10 linked objects, there are at least 11 requests and 11 responses (totally 22 messages) needed to transmit between an HTTP client and server (together with mutual acknowledgement for each packet). Though the HTTP/1.1 request pipelining method usually helps reduce the latency, this model is far from optimization in terms of message number and usage of packets. STTP reduces the network traffic by greatly reducing the number of client requests and keeping most of the packets in full size.

Using a new header field of redirection (*Followed-By*), the *S-POST* process, counterpart of HTTP *POST*, can also be performed within two messages.

Here is a brief example of Web page revisiting (For more details, see [8,9]):

S-COMPARE /index.html STTP/1.0
Host: wpp.org.cn
Linked-Object: -text/html -text/xml +image/ local-only*
ETag: 0-85f-724334c4 // ETag of the original STML document

[head]
[root Name= "/index.html" Content-Type="text/html" Offset-Size="502/27371" ETag= "0-54e-383712c4" Linked-Object="-text/html, +/*"]*
[object Name="/logo.jpg" Content-Type="image/" Offset-Size="27960/66808" ETag= "0-23f-626854c4" /]*
[object Name= "/menu.js" Content-Type="text/" Offset-Size="94920/8033" ETag="0-31d-652413c4" /]*
[/root]
[/head]

With the combination of the STML *Offset-Size* attribute and the HTTP *Content-Range* header, partial update of a single object can be efficiently realized in STTP. E.g., in a Web page (or non-root object) there are two parts (marked between specific tokens) corresponding to dynamic contents,

……<%!?# … #?!%> ……<%!?# … #?!%> ……
0 r1 r2 r3 r4 r5

When constructing a response for this page, the server may indicate that the page has two parts that are dynamic using a '+' indicator at the corresponding offset/size values,

[object …… ETag = "0-54e-383712c4" Content-Range="0-r1/, r1-r2/*, r2-r3/*, r3-r4/*, r4-r5/*" Offset-Size="o1/s1,+o2/s2, o3/s3, +o4/s4, o5/s5" … /]*

Then when revisiting the page, the client issues an *S-COMPARE* request with the information

[object …… ETag = "0-54e-383712c4" Range="r1-r2/, r3-r4/*" …]*

The server may then send only the dynamic contents for update. In partial update messages, the server should treat the entity tags of dynamic pages as weak validators [6], which are not affected by dynamic contents.

3 Web Compatibility

STTP is fully compatible with HTTP/1.*x*. STTP retains all HTTP requests and responses while supporting new messages, so that STTP clients and servers can recognize all HTTP messages. This means HTTP is a strict subset of STTP, such that HTTP and STTP clients/servers can coexist and communicate with each other. For example, using the following URLs, *sttp://wpp.org.cn/*, and *http://wpp.org.cn/*, the client should present exactly the same content to the user.

An STTP client may first use the *sttp://* scheme to retrieve resources on a server. If the server returns status code indicating HTTP client error, then it should be regarded as an HTTP server and the client may then try the *http://* scheme. On the other hand, an STTP server can easily differentiate between HTTP and STTP clients from the version field of the request line, in addition to the methods used.

4 Experimental Implementation and Tests

To validate the effect of our mechanism, we made an experimental implementation to compare the elapsed time in transmission of an identical set of Web pages using HTTP/1.1 and STTP/STML. The test set consists of 20 different HTML files, containing 2, 4, 6, …, 40 linked images respectively. The files also include a paragraph of the same text, amounting to 1876 characters. The images are saved using different file names from the same JPEG file, which has 2471 bytes. The page with 40 images is also used to test the caching based retrieval with 0, 2, 4, …, 40 images locally cached.

The network environments tested include two typical connection conditions: a fast intranet and a slow dialup line. The intranet is a 100Mbps Ethernet LAN, with RTT < 1ms and MSS = 1460. The dialup line is a 48Kpbs PPP modem line using a major public commercial dialup service, with RTT $\approx$ 220ms and MSS = 1460. On the intranet, there is one router hop between the server and the client, while on the modem line there are 8. In order to make up for network fluctuations, the tests were made after midnight at several weekends and most runs were repeated more than 10 times.

The performance tests of elapsed time and packet number and the results are listed in appendix. The results show that STTP outperformed HTTP under all circumstances tested. For the first time retrieval, the improvement is around 70% on the LAN and 25% on modem line. For 50% update retrieval, the improvement are 170% and 60% respectively. STTP is superior to HTTP for revalidate tests, usually of an order of magnitude. This is some what significant, since the bulk resources on Web servers remain to be stable [Arlitt&Williamson96], and even on some highly dynamic web sites files tend to change little when they are modified and the variation ratio is often extremely small [4]. The savings in terms of number of packets are of the same magnitude.

STTP also shows the desired scalability, that is, the faster the connection, the better it performed. Connection conditions are constantly improved, from which STTP could benefit more than HTTP.

The major shortcoming is that STML encoding, decoding and cache synchronization bring additional load for both the server and client. Using a few specific caching methods, a significant part of the load can be optimized away [9]. The cost is low on both the server and the client sides comparing to the improvement. And such load tends to be a smaller and smaller part as computer hardware technology is rapidly progressing, which is much faster than the improvement of the limits of communication connections. We may regard the Web++ framework as a load balance mechanism between the communication hosts and connections.

References

[1] Craig E. Wills, Mikhail Mikhailov, Hao Shang, "N for the Price of 1: Bundling Web Objects for More Efficient Content Delivery". In Proceedings of WWW10 (10th International World Wide Web Conference), May 1-5, 2001, Hong Kong (http://www10.org).

[2] Franks, John. MGET proposal, October 1994, http://www.ics.uci.edu/pub/ietf/http/hypermail/1994q4/0260.html

[3] Padmanabhan, Venkata N., and Jeffrey C. Mogul. "Improving HTTP Latency", Computer Networks and ISDN Systems, v. 28, pp. 25-35, Dec. 1995. Slightly revised version of paper in Proc. 2nd International WWW Conference '94: Mosaic and the Web, Oct. 1994, which is available at http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/DDay/mogul/HTTPLatency.html.

[4] Padmanabhan, Venkata N., and Lili Qiu, "The Content and Access Dynamics of a Busy Web Site: Findings and Implications", Proceedings of ACM SIGCOMM 2000.

[5] Palme, J., and A. Hopmann, "MIME E-mail Encapsulation of Aggregate Documents, such as HTML (MHTML)," RFC 2557, March 1999.

[6] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee. "Hypertext Transfer Protocol - HTTP/1.1", RFC 2616. June 1999.

[7] Stracke, J., "Encoding a DAV resource in MIME" (draft-stracke-webdav-mime-resource-00.txt), Feb. 1999. Internet Draft, IETF.

[8] Swen, Bing. An Overview of the Web++ Framework. In Proceedings of International Conferences on Info-tech & Info-net (ICII2001), Conference E (Information Network). Beijing, Oct.29 - Nov.1, 2001.

[9] Swen, Bing. Improving Web Performance Using Structural Information of Web Pages, Tech. Rept., ICL, CS Dept., Peking University, Jan. 2001. (Available at http://icl.pku.edu.cn/bswen/web++/w++intro.html)

Appendix: STTP and HTTP Performance Comparison Tests

Table 1 and 2 are the results of three different tests, that is, the packet number and elapsed time for first-time retrieval, 50% update (half of the linked images cached) and reload. Reload or revalidate is revisiting a Web page where the contents are already available in a local cache. In our cases, revalidate of a cached page results in no actual resource transfer.

Table 1 Performance Comparison on a 100Mbps LAN

linked objects	first-time retr. (packets/sec.)						50% update (packets/sec.)						reload (packets/sec.)					
	HTTP		STTP		PR	AR	HTTP		STTP		PR	AR	HTTP		STTP		PR	AR
2	12	0.121	9	0.105	0.33	0.15	11	0.060	7	0.051	0.57	0.18	8	0.040	3	0.035	1.67	0.14
4	20	0.162	14	0.124	0.43	0.31	16	0.087	9	0.070	0.78	0.24	12	0.059	3	0.035	3.00	0.69
6	28	0.204	21	0.162	0.33	0.26	23	0.128	12	0.095	0.92	0.35	16	0.073	3	0.035	4.33	1.09
8	36	0.235	25	0.187	0.44	0.26	28	0.143	16	0.121	0.75	0.18	20	0.089	3	0.035	5.67	1.54
10	45	0.471	30	0.215	0.50	1.19	35	0.196	18	0.146	0.94	0.34	24	0.110	3	0.035	7.00	2.14
12	53	0.541	35	0.260	0.51	1.08	41	0.292	21	0.176	0.95	0.66	28	0.125	3	0.035	8.33	2.57
14	61	0.727	40	0.351	0.53	1.07	47	0.390	23	0.192	1.04	1.03	32	0.133	3	0.035	9.67	2.80
16	69	0.846	45	0.441	0.53	0.92	53	0.535	25	0.208	1.12	1.57	36	0.173	3	0.035	11.00	3.94
18	77	0.929	51	0.494	0.51	0.88	59	0.634	28	0.218	1.11	1.91	40	0.250	3	0.035	12.33	6.14
20	85	1.160	56	0.641	0.52	0.81	65	0.751	30	0.232	1.17	2.24	44	0.305	3	0.035	13.67	7.71
22	93	1.337	60	0.726	0.55	0.84	71	0.863	33	0.245	1.15	2.52	48	0.406	3	0.035	15.00	10.60
24	101	1.472	65	0.818	0.55	0.80	77	0.968	36	0.274	1.14	2.53	52	0.481	3	0.035	16.33	12.74
26	113	1.627	69	0.891	0.64	0.83	83	1.099	39	0.342	1.13	2.21	56	0.561	3	0.035	17.67	15.02
28	117	1.753	76	0.974	0.54	0.80	89	1.207	42	0.389	1.12	2.10	60	0.621	3	0.035	19.00	16.74
30	125	1.933	80	1.087	0.56	0.78	95	1.302	43	0.451	1.21	1.89	64	0.721	3	0.035	20.33	19.60
32	133	2.143	86	1.167	0.55	0.84	101	1.422	46	0.521	1.20	1.73	68	0.761	3	0.035	21.67	20.74
34	143	2.243	90	1.307	0.59	0.72	109	1.556	49	0.550	1.22	1.83	72	0.788	3	0.035	23.00	21.51
36	151	2.414	94	1.392	0.61	0.73	115	1.652	51	0.561	1.25	1.94	76	0.809	3	0.035	24.33	22.11
38	159	2.553	99	1.583	0.61	0.61	121	1.767	54	0.580	1.24	2.05	80	0.831	3	0.035	25.67	22.74
40	167	2.639	104	1.667	0.61	0.58	127	1.873	58	0.661	1.19	1.83	84	0.876	3	0.035	27.00	24.03
Total	1788	25.492	1149	14.592	0.56	0.75	1366	16.925	640	6.083	1.13	1.78	920	8.212	60	0.700	14.33	10.73

Table 2 Performance Comparison on a 48Kbps Modem Line

linked objects	first-time retr. (packets/sec.)						50% update (packets/sec.)						reload (packets/sec.)					
	HTTP		STTP		PR	AR	HTTP		STTP		PR	AR	HTTP		STTP		PR	AR
2	16	1.87	11	1.37	0.45	0.36	12	1.24	8	0.76	0.50	0.63	8	0.55	3	0.22	1.67	1.50
4	26	2.86	17	2.36	0.53	0.21	20	1.96	11	1.43	0.82	0.37	12	0.74	3	0.22	3.00	2.36
6	35	4.28	24	3.24	0.46	0.32	26	2.52	14	1.98	0.86	0.27	16	0.99	3	0.22	4.33	3.50
8	42	5.38	30	4.17	0.40	0.29	33	3.68	18	2.31	0.83	0.59	20	1.21	3	0.22	5.67	4.50
10	50	6.38	36	4.94	0.39	0.29	38	3.95	21	2.69	0.81	0.47	24	1.43	3	0.22	7.00	5.50
12	59	7.36	42	5.83	0.40	0.26	46	4.68	24	3.18	0.92	0.47	28	1.45	3	0.22	8.33	5.59
14	70	8.02	49	6.59	0.43	0.22	53	5.27	27	3.63	0.96	0.45	32	1.87	3	0.22	9.67	7.50
16	76	10.16	59	8.18	0.29	0.24	58	6.53	30	4.12	0.93	0.58	36	2.14	3	0.22	11.00	8.73
18	83	11.42	62	8.67	0.34	0.32	63	7.47	34	4.62	0.85	0.62	40	2.30	3	0.22	12.33	9.45
20	94	12.47	68	10.17	0.38	0.23	70	8.30	37	5.00	0.89	0.66	44	2.53	3	0.22	13.67	10.50
22	102	13.07	74	10.43	0.38	0.25	73	8.84	40	5.44	0.83	0.63	48	2.75	3	0.22	15.00	11.50
24	106	13.73	80	11.32	0.33	0.21	80	9.06	44	5.77	0.82	0.57	52	2.91	3	0.22	16.33	12.23
26	111	15.16	87	12.14	0.28	0.25	87	10.06	47	6.10	0.85	0.65	56	3.18	3	0.22	17.67	13.45
28	122	16.94	93	12.64	0.31	0.34	93	10.36	50	6.26	0.86	0.65	60	3.41	3	0.22	19.00	14.50
30	131	17.72	99	13.70	0.32	0.29	101	10.71	53	6.49	0.91	0.65	64	3.62	3	0.22	20.33	15.45
32	143	19.28	106	14.61	0.35	0.32	105	11.09	56	7.75	0.88	0.43	68	4.06	3	0.22	21.67	17.45
34	152	19.91	112	15.60	0.36	0.28	111	12.91	60	8.24	0.85	0.57	72	4.12	3	0.22	23.00	17.73
36	161	22.16	124	16.48	0.30	0.34	119	13.95	63	8.62	0.89	0.62	76	4.37	3	0.22	24.33	18.86
38	175	22.96	128	18.13	0.37	0.27	126	15.79	66	9.07	0.91	0.74	80	4.47	3	0.22	25.67	19.32
40	183	23.67	131	19.77	0.40	0.20	132	16.48	70	9.39	0.89	0.76	84	4.56	3	0.22	27.00	19.73
Total	1937	254.80	1431	200.34	0.35	0.27	1446	164.85	773	102.85	0.87	0.60	920	52.66	60	6.60	14.33	6.98

Note:

packet saving ratio $PR = (packet\text{-}no_{HTTP} - packet\text{-}no_{STTP}) / packet\text{-}no_{STTP}$

acceleration ratio $AR = (time_{HTTP} - time_{STTP}) / time_{STTP}$

Table 3 and 4 are the comparison of transmission time and packet numbers of a page with 40 linked objects and different numbers of objects being cached (the page is not cached). Again, STTP needs only one request for the revalidate of all the cached images and the retrieval of other files. The packets transmitted were solely used for resources transmission. All response packets (except for the last one) were in the full size.

Table 3 100Mbps LAN

cached objects	update reload (packets/sec.)					
	HTTP		STTP		PR	AR
0	167	2.078	112	1.702	0.49	0.22
2	163	2.043	105	1.508	0.55	0.35
4	159	2.013	102	1.367	0.56	0.47
6	155	1.963	96	1.262	0.61	0.56
8	151	1.913	91	1.251	0.64	0.53
10	147	1.873	86	1.072	0.71	0.75
12	143	1.783	82	1.031	0.74	0.73
14	139	1.722	78	0.992	0.78	0.74
16	135	1.662	69	0.911	0.96	0.82
18	131	1.598	65	0.762	1.02	1.10
20	127	1.528	61	0.711	1.08	1.10
22	123	1.462	54	0.601	1.27	1.43
24	119	1.392	49	0.471	1.43	1.96
26	115	1.342	45	0.436	1.56	2.08
28	111	1.272	39	0.330	1.85	2.85
30	107	1.226	33	0.261	2.24	3.70
32	103	1.167	28	0.231	2.68	4.05
34	99	1.061	22	0.200	3.50	4.31
36	95	1.042	18	0.170	4.28	5.13
38	91	0.982	12	0.150	6.58	5.55
40	87	0.921	7	0.055	11.43	15.75

Table 4 48Kbps Modem Line

cached objects	update reload (packets/sec.)					
	HTTP		STTP		PR	AR
0	201	21.70	140	21.04	0.44	0.03
2	198	21.42	133	19.36	0.49	0.11
4	195	21.26	130	19.14	0.50	0.11
6	187	21.20	123	18.07	0.52	0.17
8	181	20.87	114	17.17	0.58	0.22
10	177	20.57	106	16.43	0.67	0.25
12	173	18.43	100	15.19	0.73	0.21
14	166	17.94	94	14.28	0.77	0.26
16	163	17.26	87	12.93	0.87	0.33
18	161	16.17	80	12.02	1.01	0.35
20	155	14.75	74	10.93	1.09	0.35
22	151	13.82	67	9.83	1.25	0.41
24	147	13.32	61	9.04	1.41	0.47
26	142	12.30	54	7.91	1.63	0.55
28	139	12.09	46	7.17	2.02	0.69
30	136	11.65	40	5.66	2.40	1.06
32	131	10.27	34	4.64	2.85	1.21
34	128	9.73	26	3.68	3.92	1.64
36	122	8.77	20	2.61	5.10	2.36
38	110	7.01	13	1.73	7.46	3.05
40	91	4.21	7	0.60	12.00	6.02