Network Working Group
Request for Comments: 392
NIC: 11584

G. Hicks
B. Wessler
Utah

20 September 1972

Measurement of Host Costs for Transmitting Network Data

Background for the UTAH Timing Experiments

Since October 1971 we, at the University of Utah, have had very large compute bound jobs running daily. These jobs would run for many cpu hours to achieve partial results and used resources that may be better obtained elsewhere. We felt that since these processes were being treated as batch jobs, they should be run on a batch machine.

To meet the needs of these "batch" users, in March of this year, we developed a program[1] to use the Remote Job Service System (RJS) at UCLA-CCN. RJS at UCLA is run on an IBM 360/91.

Some examples of these jobs were (and still are!):

(a) Algebraic simplification (using LISP and REDUCE)

(b) Applications of partial differential equation solving

(c) Waveform processing (both audio and video)

The characteristics of the jobs run on the 91 were small data decks being submitted to RJS and massive print files being retrieved. With one exception: The waveform processing group needed, from time to time, to store large data files at UCLA for later processing. When this group did their processing, they retrieved very large punch files that were later displayed or listened to here.

When the program became operational in late march -- and started being used as a matter of course -- users complained that the program page faulted frequently. We restructured the program so that the parts that were often used did not cross page boundaries.

The protocol with RJS at UCLA requires that all programs and data to be transmitted on the data connection be blocked[2]. This means that we simulate records and blocks with special headers. This we found to be another problem because of the computation and core space involved. This computation took an appreciable amount of time and core space we found because of our real core size that we were being charged an excessive amount due to page faulting. The page faulting also reduced our real-time transmission rate to the extent that we

felt a re-write of the transmitting and receiving portions of the program was needed. In order that the program receive the best service from the system, these portions optimized so that they each occupied a little over half of a page. As we now had so few pages in core at any one time, the TENEX scheduler could give the program preference over larger working set jobs. (As an aside, because of our limited core, we have written a small (one and one half pages) editor in order to provide an interactive text editing service.)

The mechanism to access the network under TENEX is file oriented. This means byte-in (BIN) and byte-out (BOUT) must be used to communicate with another host. The basic timing of these two instructions (in the fast mode) is 120 us per byte to get the data onto or off of the network[3]. A distinction was made because the TENEX monitor must do some "bit shuffling" to ready the users bytes to be transmitted or it must put the network messages into some form that is convenient for the user. This is the "slow bin, bout" and occurs once per message. If the users bytes are 36 bits long then it will take an average of 500 us per message. If the bytes have to be unpacked from the message to be usable, then it may take up to a milli-second depending on the size of the message[3].

II. Measurements and Results

We found by timing various portions of the program that the RJS program was using 600 to 700 us per bit byte or between 75 and 85 micro-seconds of chargeable cpu time per bit. (See tables 1 and 2 for actual results). A short discussion of how these figures were obtained is now in order. NOTE! We have not been trying to measure network transmission rates; Rather, how much it costs us to take a program (data) from our disk and send it to another host to be executed (processed).

Column 1 is the clock time (real-time) from when the first byte was brought in from the disk until the last byte had gone onto the network. (Or from the time we received the first byte from the network until the disk file was closed).

Column 2 is computed in the same manner as column 1 except that it is the chargeable runtime for the process.

Column 3 is the actual number of bytes that went onto or came from the network. The letter that follows this column indicates the direction. E.G. s for sending to UCLA, r for receiving from UCLA).

Column 4 was calculated by the following formula:
Bits per second = (real-time)/((number of bytes)*8)

Column 5 was calculated by the formula:
us/bit = (chargeable runtime)*1000/((number of bytes)*8)

Column 6 is the 5 minute load average. (See TENEX documentation for details.)

Using these figures we can conclude that for a million bits of information -- programs to be executed or data -- it would take 75 to 85 cpu seconds to transmit. At a cost of $474.60 per cpu hour on TENEX's[5], this millionbits would cost $9.90 to 11.20 to transfer from the originating host and potentially the same for the foreign host to receive. This is about 33 to 37 times higher than the predicted network transmission costs[4].

It is to be noticed that, in some cases, for programs to be transmitted over the network, the cost incurred by transmitting them was greater than the cost of executing these programs at the foreign host!

III. Analysis

There may be numerous ways to reduce the cost of the network to the host:

(a) Treat the network not as a file device but as an interprocess communications device[6].

(b) Create an 'intelligent' network input/output device. This would, of course, be customized for individual types of operating systems and hardware configurations. For TENEX systems this could be implemented as the ability to do mapping operations from the users virtual memory 'directly' onto the network. In any case, this intelligent network device would be required to handle the various protocols for the host. Some changes may be required in the NCP protocols.

A way to reduce the cost of the RJS program (the one measured in tables 1 and 2) would be to change the RJS-UCLA protocol. One possible change is to allow hosts the option of using 32 bit bytes (because it may be more efficient!) instead of the 8 bit bytes now required by the protocol.

Basically, it is our belief, that, in order to make the network as viable economically as was anticipated by the authors of reference[4], much work is needed on host machines and network protocols rather than on further refinements of the communication devices involved.

References

1. Hicks, Gregory, "Network Remote Job Entry Program--NETRJS", Network Information Center #9632, RFC #325

2. Braden, R.T., "Interim NETRJS Specifications", Network Information Center #7133, RFC #189, July 5,1971

3. Personal correspondence with R. S. Tomlinson of Bolt, Beranek & Newman during the time period of 13-SEPT-71 to 19-SEPT-72.

4. Roberts, L.G., and B.D. Wessler, "Computer Network Development to achieve resource sharing", Spring Joint Computer Conference, May 7,1970 pg 543-549.

5. Personal correspondence with Bolt, Beranek & Newman

6. Bressler, B., D. Murphy and D. Walden, "A proposed Experiment with a Message Switching Protocol", Network Information Center #9926, RFC #333, May 15,1972.

Utah-10 Accounting for Network Usage

for the period 16-SEP-72 12:48:34, ending 19-SEP-72 13:56:11

Clk Tim	Cpu Tim	# of Bytes		Bits/sec	us/bit	Load
14	11.61	18930	s	10152.175	76.67	3.74
02:56	37.89	59066	r	2670.857	80.20	3.51
02:18	22.71	35377	r	2038.682	80.24	2.98
01:31	34.37	56608	s	4966.431	75.89	3.35
13	11.57	19094	s	10985.401	75.72	4.07
04:03	42.03	63067	r	2069.297	83.30	4.95
03	1.82	2906	s	5932.126	78.37	5.58
45	23.58	35505	r	6237.976	83.00	5.37
09	2.08	3243	s	2804.757	80.21	3.60
03:28	39.25	58632	r	2246.727	83.69	4.86
05	4.60	7470	s	10192.734	76.99	1.12
23	10.83	16525	r	5565.378	81.95	1.17
06	4.32	7142	s	9116.962	75.64	1.44
14	8.56	13223	r	7170.338	80.95	1.29
11	4.42	7142	s	4795.300	77.43	1.89
01:34	13.287	19562	r	1659.819	84.86	2.50
37	10.35	16183	r	3439.807	79.97	3.02
02:43	34.49	56444	s	2764.170	76.38	3.74
38	10.51	16196	r	3400.467	81.13	0.69
45	34.12	56280	s	9820.704	75.75	2.57
03:46	36.09	56280	s	1990.601	80.16	4.06
11	2.75	4085	r	2774.900	84.30	4.86
15	2.88	4085	r	2154.252	88.07	4.86
01:54	11.40	16125	r	1124.203	88.39	5.12
01:14	35.10	56280	s	6057.068	77.96	6.10
01:07	10.67	16125	r	1919.986	82.70	1.89
04:28	36.32	56362	s	1679.377	80.56	5.52
02:12	17.71	27120	r	1634.818	81.62	1.73
06:59	41.88	64333	s	1226.980	81.37	6.66
37	7.63	12082	r	2552.243	78.97	0.64

Utah-10 Accounting for Network Usage

for the period 13-SEP-72 2:23:12, ending 16-SEP-72 11:47:07

Clk Tim	Cpu Tim	# of Bytes		Bits/sec	us/bit	Load
10	2.09	3079	s	2343.227	84.77	3.80
11:09	138.20	204596	s	2444.733	84.43	3.68
06:16	34.78	49994	r	1062.961	86.96	3.95
01:57	16.25	24971	r	1693.451	81.34	2.92
12:07	114.70	183598	s	2019.577	78.09	6.79

01:13	0.92	845	r	91.683	135.80	2.12
05	5.07	7373	s	10842.647	85.99	1.93
03:09	42.10	62414	r	2633.655	84.31	3.86
13:22	115.13	183352	s	1828.467	78.49	0.58
02	0.25	233	s	907.056	134.12	6.05
07:10	44.23	64869	r	1206.001	85.23	5.07
04	0.33	233	s	402.679	179.18	2.24
11:47	114.48	183585	s	2076.187	77.95	2.73
17:45	128.25	185908	r	1395.801	86.23	5.19
09:34	45.97	67158	r	935.067	85.56	0.61
09:23	113.50	183270	s	2600.852	77.41	9.64
12:24	51.65	74916	r	804.656	86.18	9.28
13:30	117.92	183352	s	1809.320	80.39	9.08
19:23	56.42	89640	s	616.586	78.67	6.77
11:49	11.29	16205	r	182.767	87.08	10.17
09:05	34.35	50796	s	744.325	84.53	8.47
21:12	56.17	76423	r	480.512	91.88	7.53
01:00	15.33	23930	r	3156.628	80.08	3.11
03:04	54.60	89731	s	3892.062	76.07	3.81
06	2.62	4106	r	5071.484	79.88	3.77
05:15	54.79	89731	s	2277.559	76.32	3.68
03	2.02	3161	s	7778.530	79.92	2.17
33	9.42	14680	r	3472.810	80.19	2.31
00	0.22	219	s	2646.526	127.28	1.81
19:57	295.16	473489	s	3162.399	77.92	1.85
10	6.62	10025	r	7841.987	82.54	2.75
01	0.23	221	s	1092.032	128.96	2.74
16	6.45	10032	r	1888.591	80.36	2.79
04	2.06	3243	s	6020.887	79.52	2.62
01:28	31.29	48532	r	4382.419	80.60	2.62
07:17	196.34	316072	s	5777.687	77.65	3.86
01:46	30.14	45786	r	3434.229	82.29	3.26
01:30	24.73	38405	r	3399.274	80.50	1.80
02:10	23.46	35633	r	2190.508	82.31	2.61
44	28.80	46897	s	8441.544	76.76	3.26
04:51	192.20	316318	s	8671.027	75.95	3.10
40	11.51	18511	s	3633.437	77.70	2.98
12	7.17	10963	r	6894.427	81.76	3.04
12	11.30	18511	s	11418.614	76.32	3.14
14	7.12	11122	r	6298.740	80.03	3.24
02	0.92	1412	s	5120.580	81.53	3.41
14	7.23	11122	r	6184.042	81.24	3.20

[This RFC was put into machine readable form for entry]
[into the online RFC archives by Helene Morin, Viagenie, 12/99]