Proceedings of the

23-24 July 1986

Internet Engineering

Task Force

Prepared by:
Phillip Gross

THIRD IETF

The MITRE Corporation
1820 Dolley Madison Boulevard
McLean, Virginia 22102

Table of Contents

Internet Engineering Task Force

23-24 July 1986

Prepared by

Phill Gross
MITRE Corp.

Table of Contents

1. Introduction

The DARPA Internet Engineering Task Force met Wednesday and Thursday, 23-24 July 1986, at the University of Michican in Ann Arbor. The meeting was hosted by Hans-Werner Braun.

1.1 Attendees

Name	Organization	Net Address
Hans-Werner Braun	U of Mich	hwb@gw.umich.edu
Mike Brescia	BBNCC	brescia@bbnccv
Mike Corrigan	OSD	corrigan@sri-nic
Marianne Gardner	BBNCC	mgardner@bbncc5
Phill Gross	MITRE	gross@mitre
Robert Hinden	BBNCC	hinden@bbnccv
Mike Karels	UCBerkeley	karels@berkeley.edu
Mark Lottor	SRI	mkl@sri-nic
David Mills	Linkabit	mills@isid.arpa
Paul Mockapetris	ISI	pvm@isi.edu
John Mullen	CMC	ucsbcsl!cmcvax!jrm@berkeley.arpa
Ron Natalie	BRL	Ron@brl
Mike St. Johns	DCA/B612	stjohns@sri-nic
Zaw-Sing Su	SRI	zsu@sri-tsc
Mitch Tasman	BBNCC	mtasman@cct.bbn.com
Dave Van Belleghem	NSF	vanb@nrl-acoustics.arpa
Steve Wolff	NSF	steve@brl
Lixia Zhang	MIT-LCS	lixia@xx.mit.edu

2. Agenda

(as distributed prior to the meeting)

23 July 1986, Wednesday

Morning - Status Reports

0900	Chairman's Remarks - Mike Corrigan, OSD
0910	NSFNet - Hans-Werner Braun, UMich
0950	Break
1000	UNIX 4.3 Networking Enhancements - Mike Karels, UCB
1050	Break
1100	Internet Performance Report - Phill Gross, MITRE
1120	Internet Performance Report - Marianne Gardner, BBN
1140	Ad Hoc Status Reports
1200	Lunch

Afternoon - Old Business

1300	Internet Measurement Criteria - Lixia Zhang, MIT
1330	ICMP Enhancements and Changes
1415	Break
1430	EGP Enhancements and Changes - Mike StJohns, DDN PMO
1600	Break
1610	PDN Cluster Masks - Carl-Herbert Rokitanski, DFVLR
1700	Recess

24 July 1986, Thursday

Morning - New Business

0900	Name Domains - Paul Mockapetris, ISI
1020	Break
1035	Name Domain Transition Planning, Mike Corrigan, OSD
1200	Lunch

Afternoon - New Business (continued)

1330	Internet Capacity Planning, Bob Hinden, BBN
1450	Break
1500	ISO Transition Planning - Phill Gross, MITRE
1550	Break
1600	Assignment of Action Items - Mike Corrigan, OSD
1700	Adjourn

3. Meeting Notes

3.1 July 23, 1986

3.2 July 24, 1986

Agenda
Internet Engineering Task Force
23-24th July 1986
University of Michigan, Ann Arbor, Michigan

23 July 1986, Wednesday

Morning - Status Reports

0900 Chairman's Remarks - Mike Corrigan, OSD
0910 NSFNet - Hans-Werner Braun, UMich
0950 Break
1000 UNIX 4.3 Networking Enhancements - Mike Karels, UCB
1050 Break
1100 Internet Performance Report - Phill Gross, MITRE
1120 Internet Performance Report - Marianne Gardner, BBN
1140 Ad Hoc Status Reports

1200 LUNCH

Afternoon - Old Business

1300 Internet Measurement Criteria - Lixia Zhang, MIT
1330 ICMP Enhancements and Changes
1415 Break
1430 EGP Enhancements and Changes - Mike StJohns, DDN PMO
1600 Break
1610 PDN Cluster Masks - Carl-Herbert Rokitanski, DFVLR
1700 Recess

24 July 1986, Thursday

Morning - New Business

0900 Name Domains - Paul Mockapetris, ISI
1020 Break
1035 Name Domain Transition Planning, Mike Corrigan, OSD

1200 Lunch

Afternoon - New Business (continued)

1330 Internet Capacity Planning, Bob Hinden, BBN
1450 Break
1500 ISO Transition Planning - Phill Gross, MITRE
1550 Break
1600 Assignment of Action Items - Mike Corrigan, OSD
1700 Adjourn

Date: Tue, 22 Jul 86 00:27:05 edt
From: gross@mitre.ARPA (Phill Gross)
Organization: The MITRE Corp., Washington, D.C.
To: ineng-tf@isib
Subject: IETF Agenda Comments

Comments on the Agenda for the 23-24 July IETF

A quick perusal of the notes from the last meeting reveals a plethora of oft-discussed-but-yet-to-be-resolved issues on our plate. Mike C. returned from the IAB with the plate stacked even higher. This has led to a packed (probably overly ambitious) agenda, which we've divided into four major sessions: Status Reports, Old Business, New Business and Action items.

Under Status Reports, we'll find out how we're doing now (MG, PG), how activities are progressing which promise to make it worse (HWB) and whether Unix will ever be a respectable gateway (MK). As time permits, other contractors can also chart their progress.

In the next session, we tackle Old Business items- some older than others. We'll have to postpone Noel's discussion but Lixia has volunteered to help Mike StJ expand the EGP talk to more than fill the void. The MOST important goal in these talks is to resolve these topics once and for all or to establish a clear course of action toward final resolution. When we set action items this time, we should give these areas first priority.

The question of Name Domain's applicability for the DoD world is important enough to give it a full morning in the New Business session. Some of the questions concerning the suitability of Domains for the DoD environment are:

- How survivable is the Domain model? What are the estimates/experience with traffic generated by Name Servers? Have caching guidelines been established.

- What assumptions does the Domain model make about the Internet environment? Specifically, does it mesh well with the Internet, or is it more complicated than necessary? Would simplifications lead to more robustness and survivability, that might be more appropriate for the DoD.

- Are there implementations suitable for conformance testing?

- How similar/compatible are Domains to analogous ISO services? Has anyone considered ISO transition or interoperability?

Paul Mockapetris will allay our fears about these Domain issues and Mike C. will lead a discusion on DDN Domain Transition Planning. Internet Capacity Planning and ISO Transition Planning are two other New Business items that hopefully won't get time squished but, of course, there is always next meeting (where they will be old business).

We managed to get out of the last meeting without setting specific action items. For this meeting, time has been alotted for this but, in retrospect, it's woefully too short. Below is a (surely incomplete) smorgasbord of potential action items. As a first step, perhaps we need to draft a Task Force Planning Paper or an Internet Engineering Program Plan, that encompasses the items below and others. As part of

this, funding levels, funding agencies and potential contractors would need to be recommended.

Action Items:

- Attack EGP! List problems, prioritize, list options for solutions, write RFCs documenting/standardizing the solutions. (See 8-9 Apr IETF meeting notes and recent messages by Mills and Zhang for issues. Other issues include estimating difficulty of enforcing an "authorized" gateway list and investigating separate cores.)

- Produce plan for Internet growth analogous to DDN Capacity White Paper (e.g., chart Buttergate deployment, core gateways, mail bridges, EGP capacity, etc.).

- Consolidate Noel's "Host Interconnection to the Internet" and new ICMP ideas. Document as one or more RFCs. Review ISO ES-IS for analogous DoD functionality.

- Congestion Control
  - Review Nagle and Zhang congestion control schemes. Estimate effort to test in the Buttergate.
  - Document Lixia's thoughts on IP congestion control as an RFC.
  - Determine network and Internet performance characteristics needed for Lixia's scheme.

- Review Name Domain concept for suitability in DDN.

- Prepare ISO Transition Plan.

- Document the SPF routing protocol as an RFC.

Appendix A - Presentation Slides

1) NSFNet - Hans-Werner Braun, UMich

2) UNIX 4.3 Networking Enhancements - Mike Karels, UCB

3) Internet Performance Report - Phill Gross, MITRE

4) Internet Performance Report - Marianne Gardner, BBN

5) Internet Measurement Criteria - Lixia Zhang, MIT

6) EGP Enhancements and Changes - Mike StJohns, DDN

7) Name Domains - Paul Mockapetris, ISI

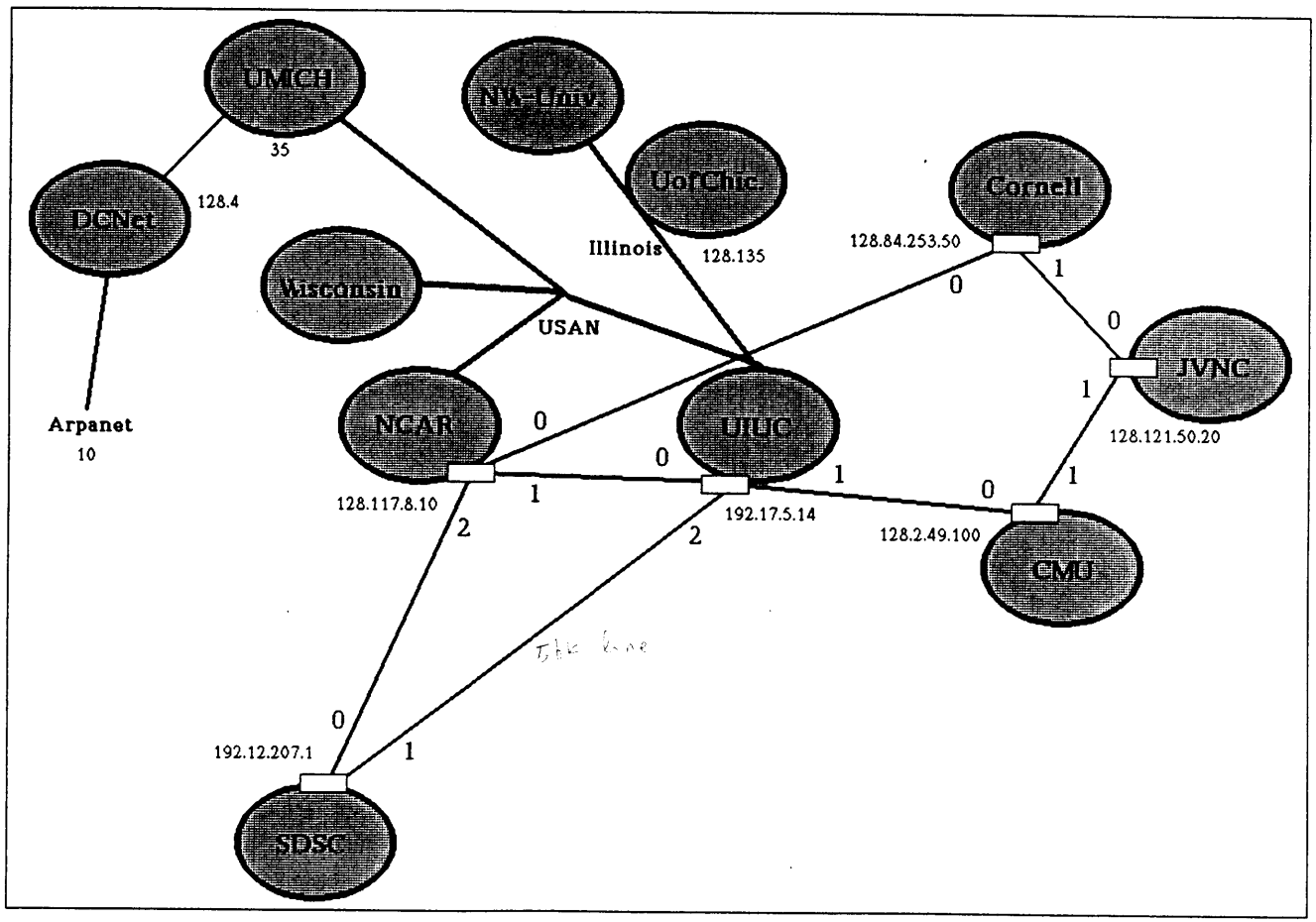
8) Internet Capacity Planning, Bob Hinden, BBN

9) ISO Transition Planning - Phill Gross, MITRE

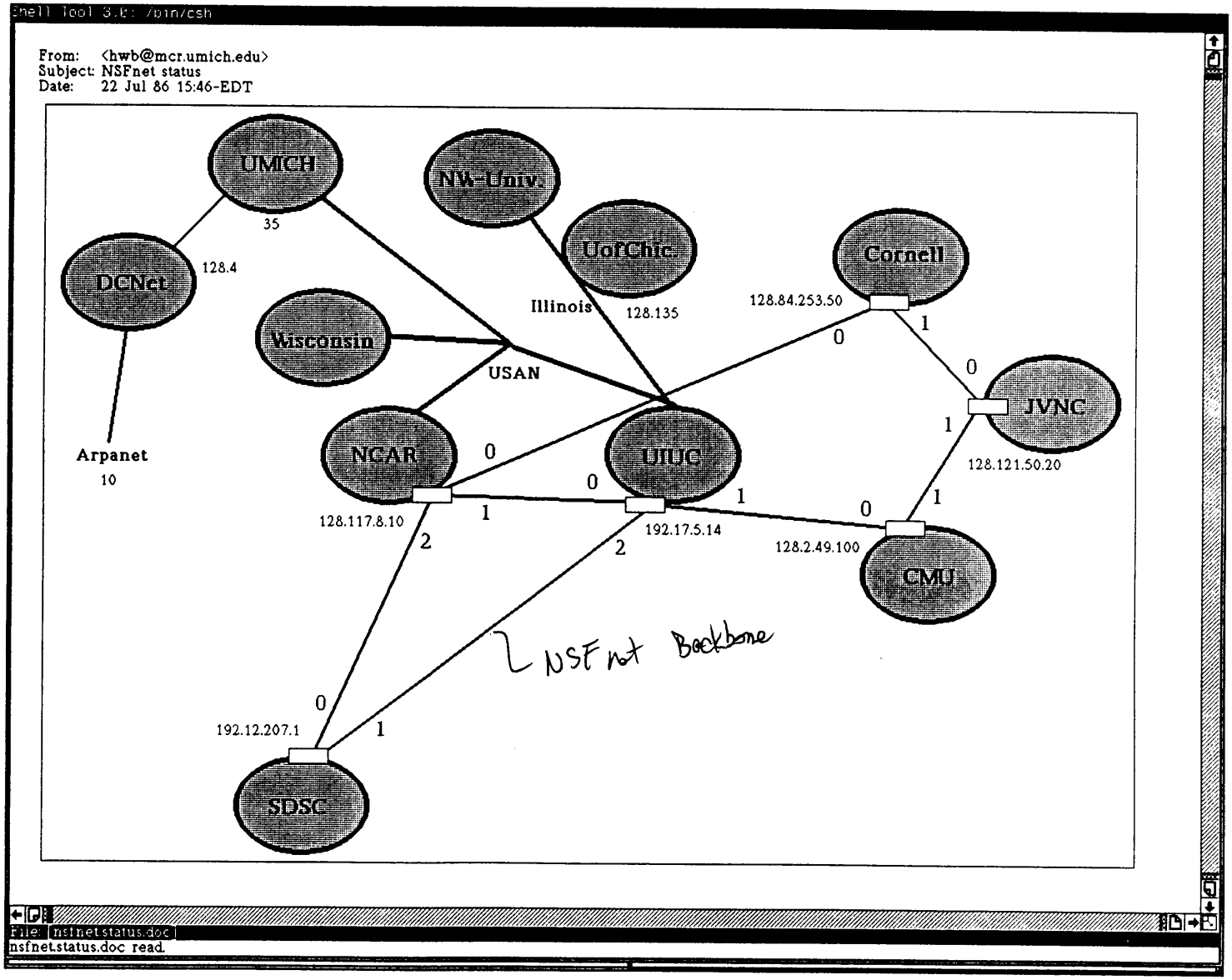
Presentation Slides

1) NSFNet - Hans-Werner Braun, UMich

From: <hwb@mcr.umich.edu>
Subject: NSFnet status
Date: 22 Jul 86 15:46-EDT

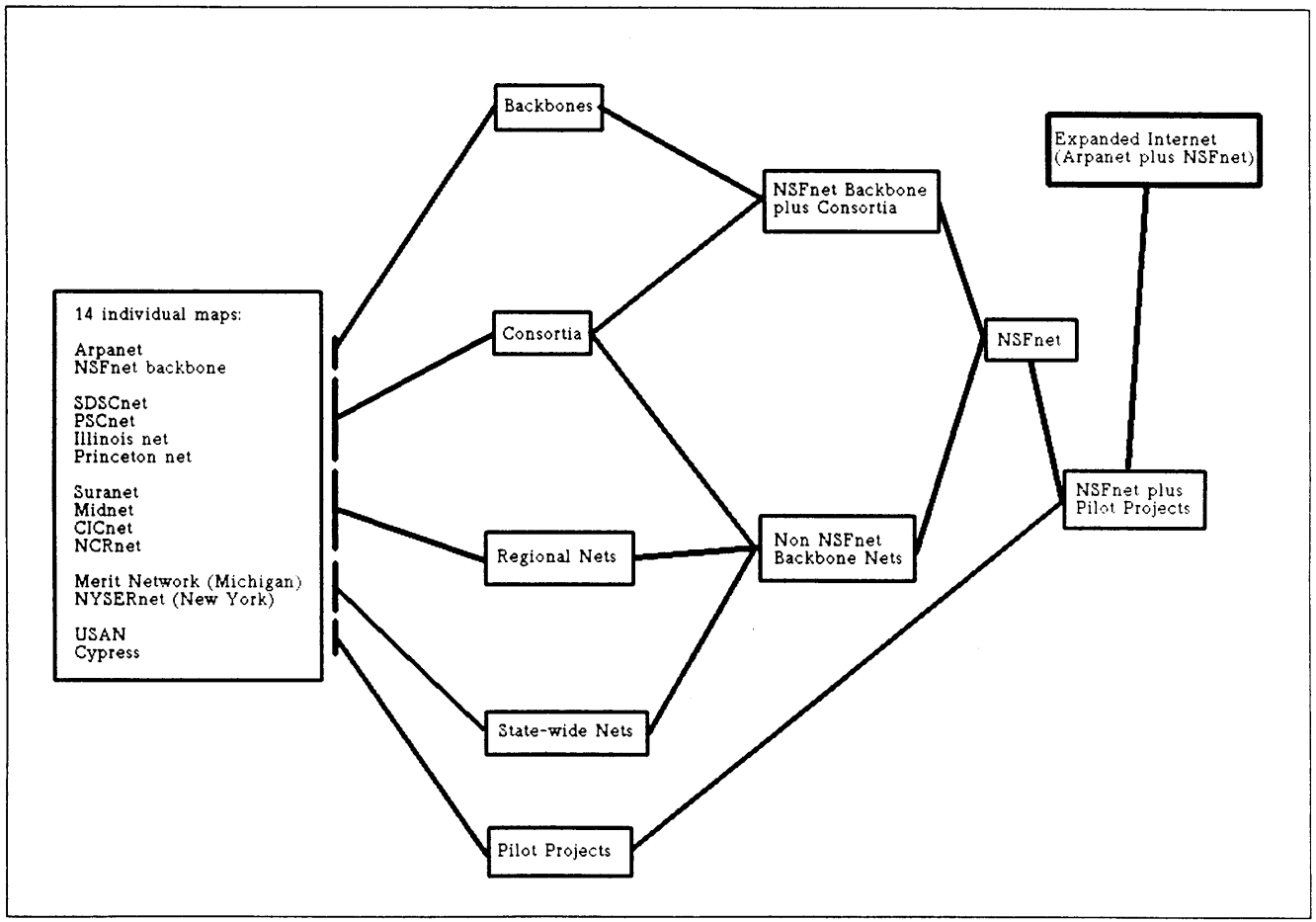


nsfnet.status.doc read.


Shell Tool 3.0: /bin/csh
From: <hwb@mcr.umich.edu>
Subject: NSFnet status
Date: 22 Jul 86 15:46-EDT
UMICH
35
NYs-Univ.
UofChic.
Illinois
128.135
DCNet
128.4
Cornell
128.84.253.50
Wisconsin
USAN
JVNC
128.121.50.20
Arpanet
10
NCAR
128.117.8.10
UIUC
192.17.5.14
128.2.49.100
CMU
SDSC
192.12.207.1
NSFnet Backbone
File: nsfnet.status.doc
nsfnet.status.doc read.

Shell Tool 3.0: /bin/csh

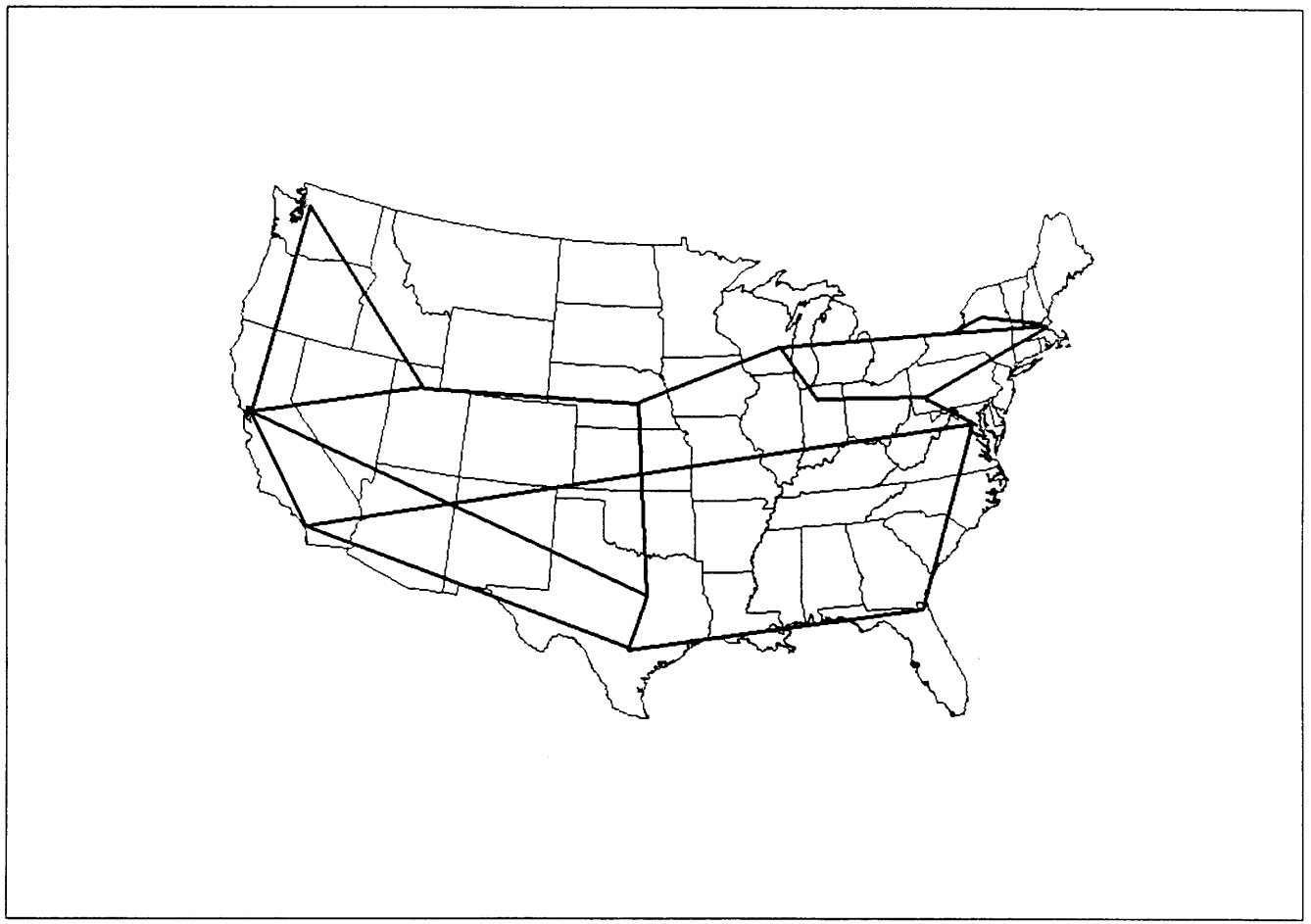
From: <hwb@mcr.umich.edu>
Subject: Expanded Internet maps / presentation flow chart
Date: 21 Jul 86 10:43-EDT



File: [usa.pres.doc]

Shell Tool 3.0: /bin/csh

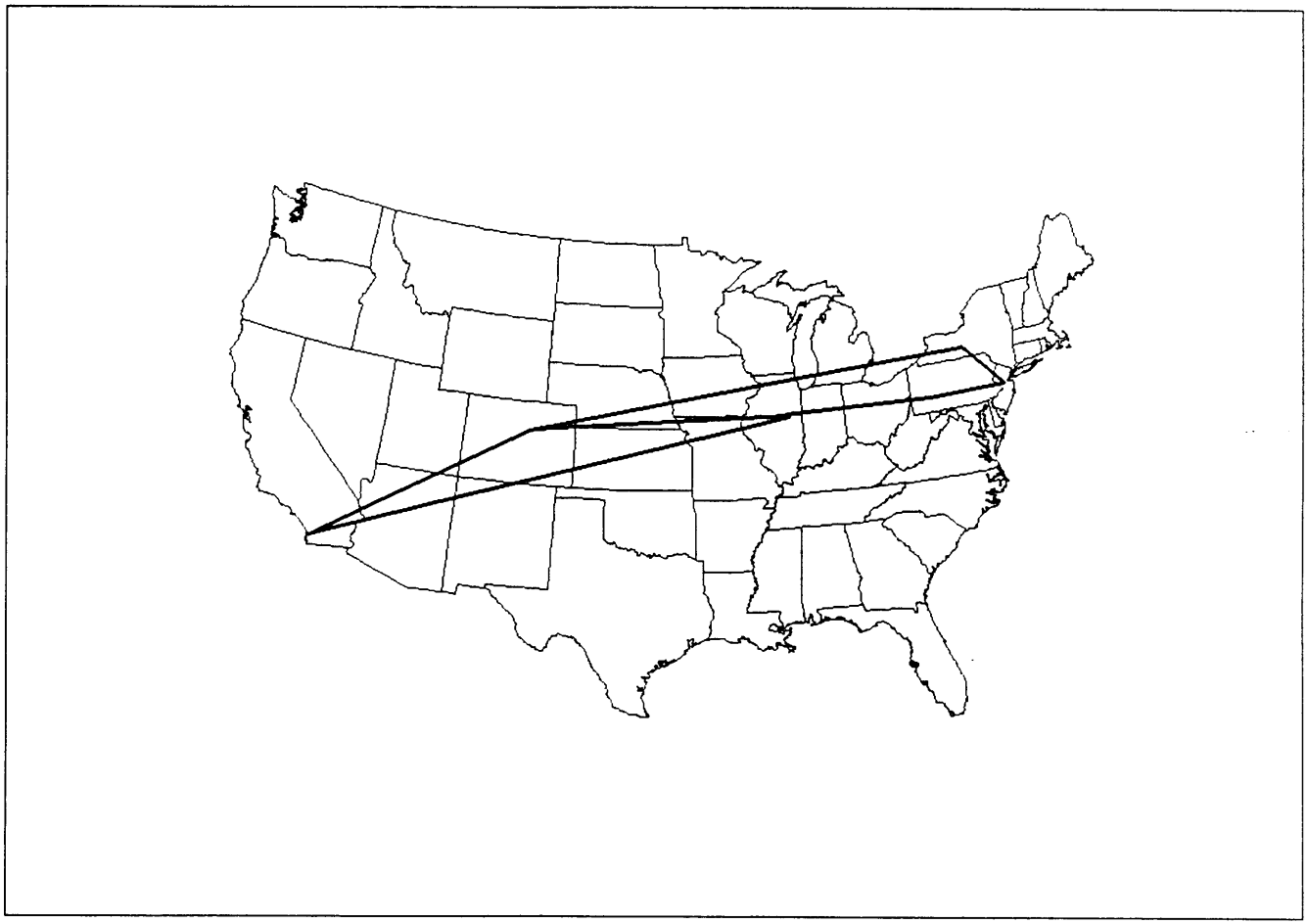
From: <hwb@mcr.umich.edu>
Subject: Arpanet
Date: 20 Jul 86 11:47-EDT



File: [usa arpa.doc]

shell Tool 3.0: /bin/csh

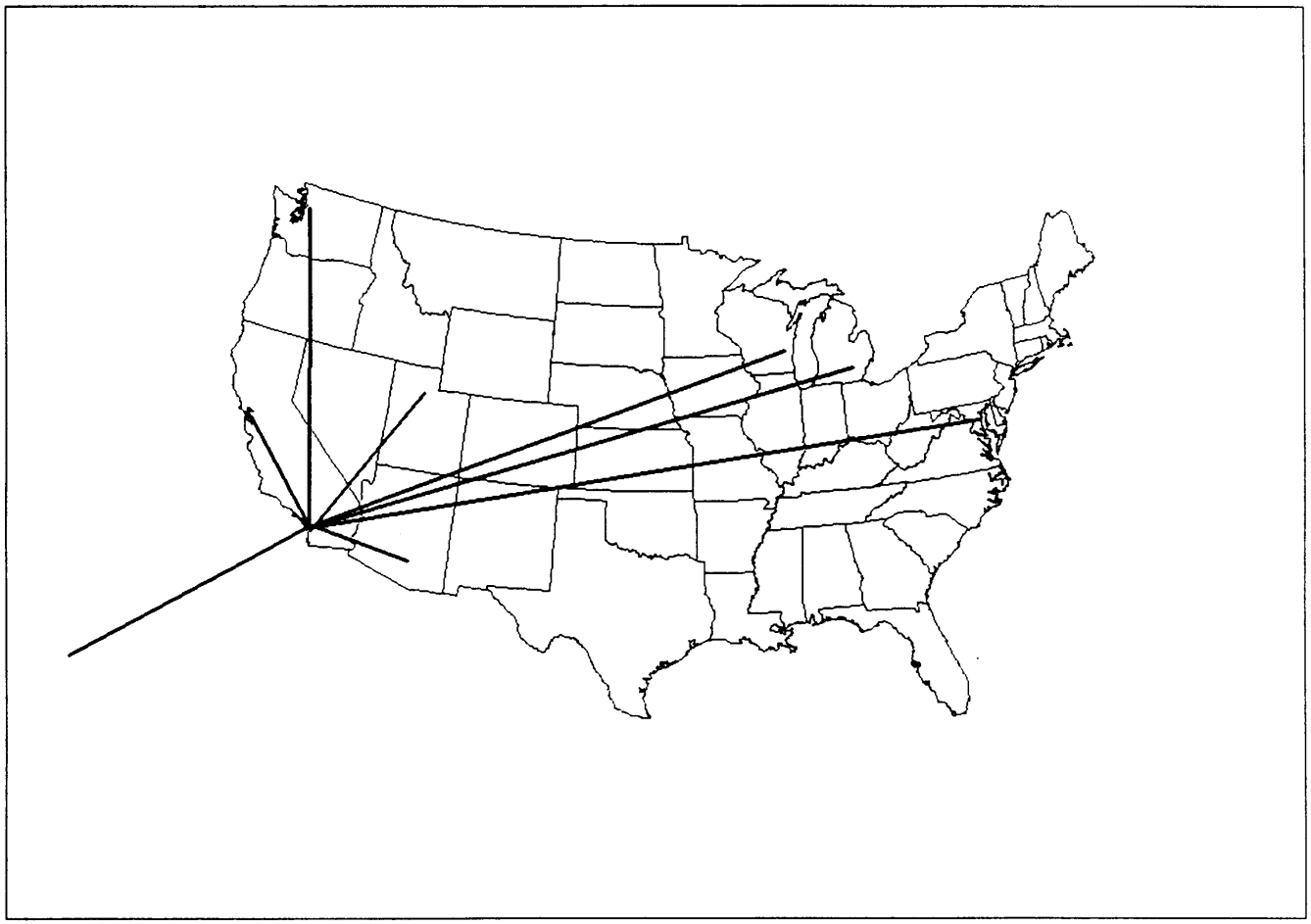
From: <hwb@mcr.umich.edu>
Subject: NSFnet backbone
Date: 20 Jul 86 11:50-EDT



File: usa.backbone
usa.backbone read.

shell tool 3.0: /bin/csh

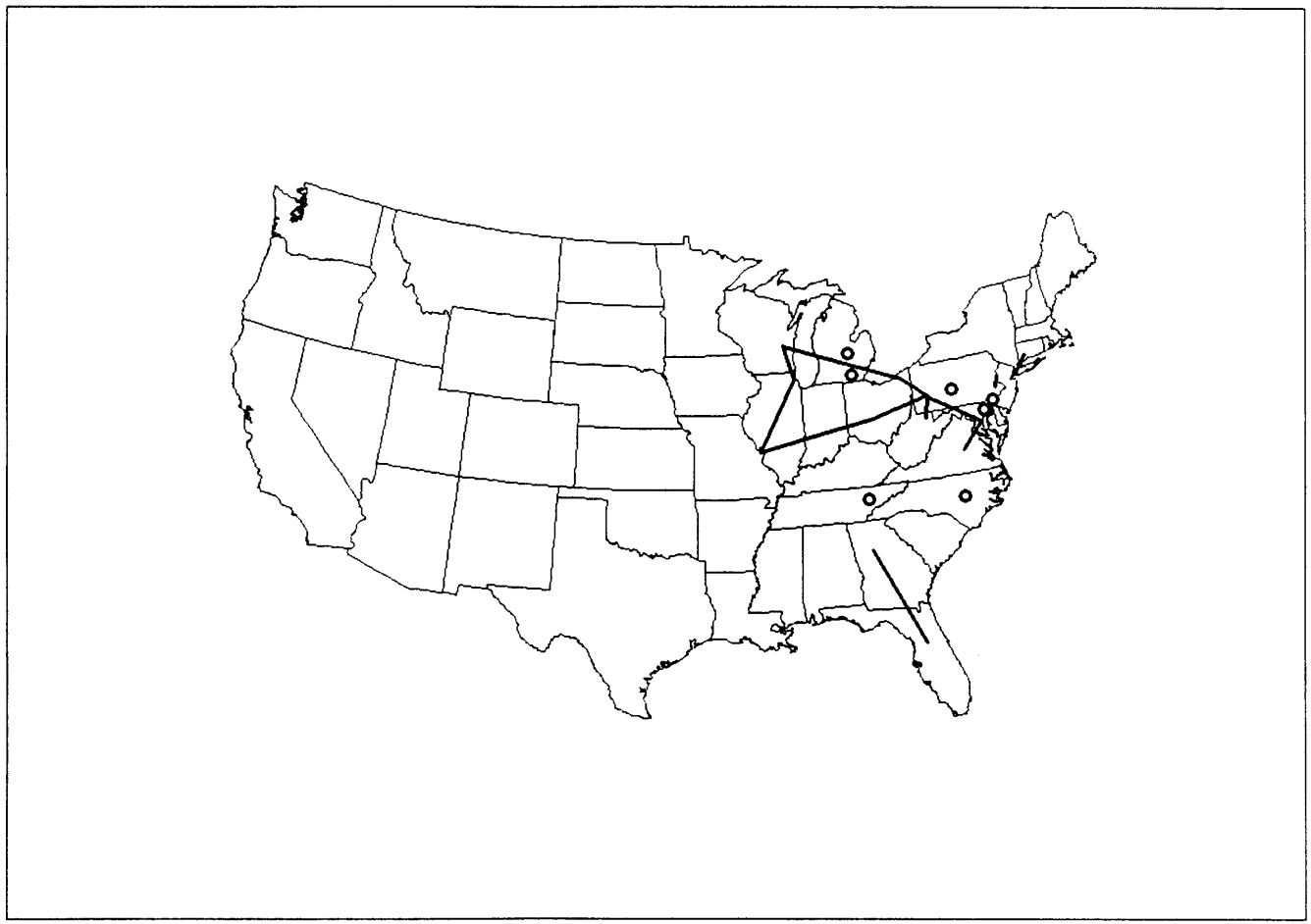
From: <hwb@mcr.umich.edu>
Subject: San Diego Supercomputer Consortium SDSCnet
Date: 20 Jul 86 11:33-EDT



File: usa.sdsc
usa.sdsc read.

Shell Tool 3.0: /bin/csh

From: <hwb@mcr.umich.edu>
Subject: Pittsburgh Supercomputer Center Network
Date: 20 Jul 86 13:25-EDT



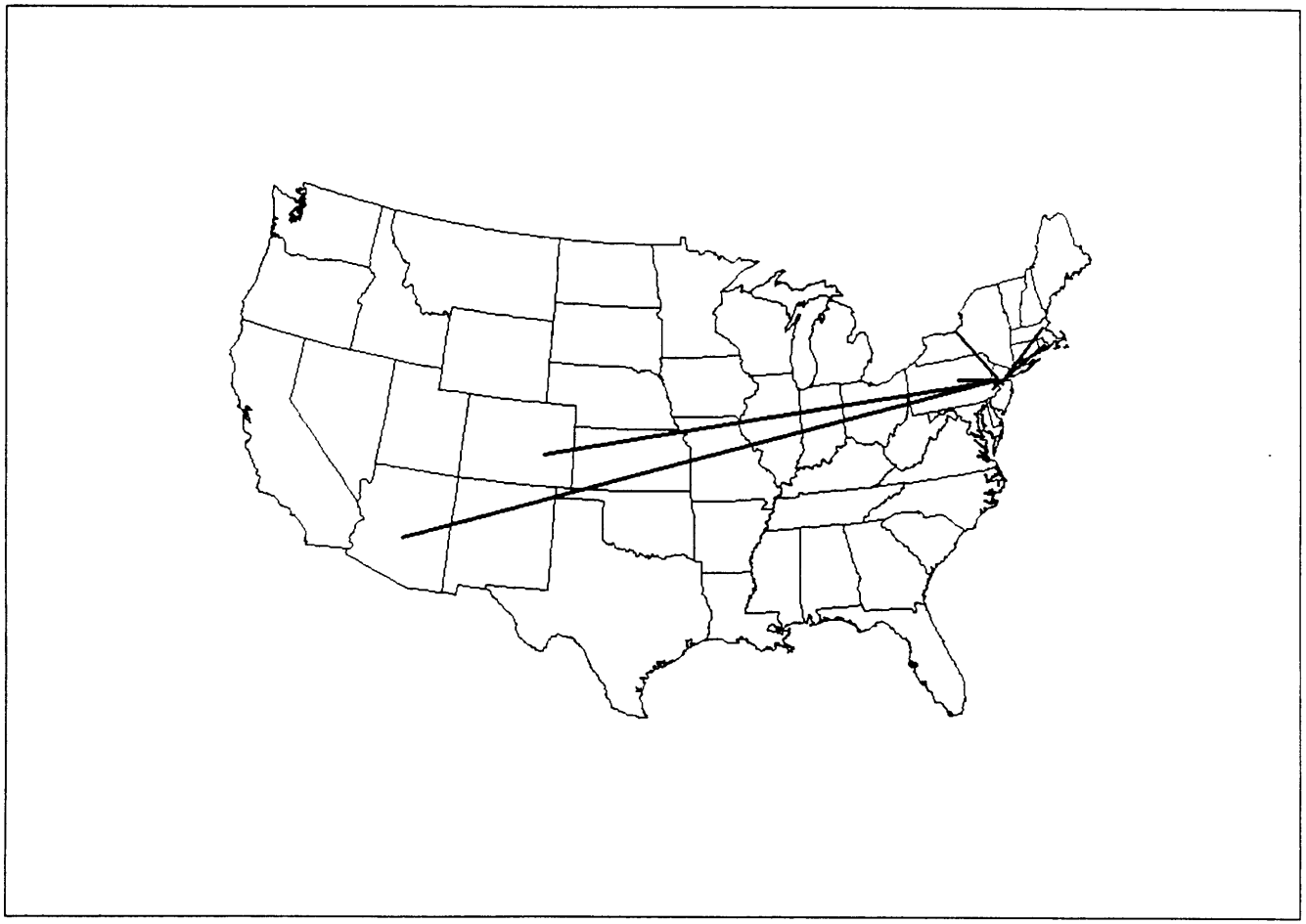
File: [usa.psc.doc]

Shell Tool 3.0: /bin/csh

From: <hwb@mcr.umich.edu>
Subject: Illinois net
Date: 20 Jul 86 11:54-EDT

File: usa.illinois
usa.illinois.doc written.

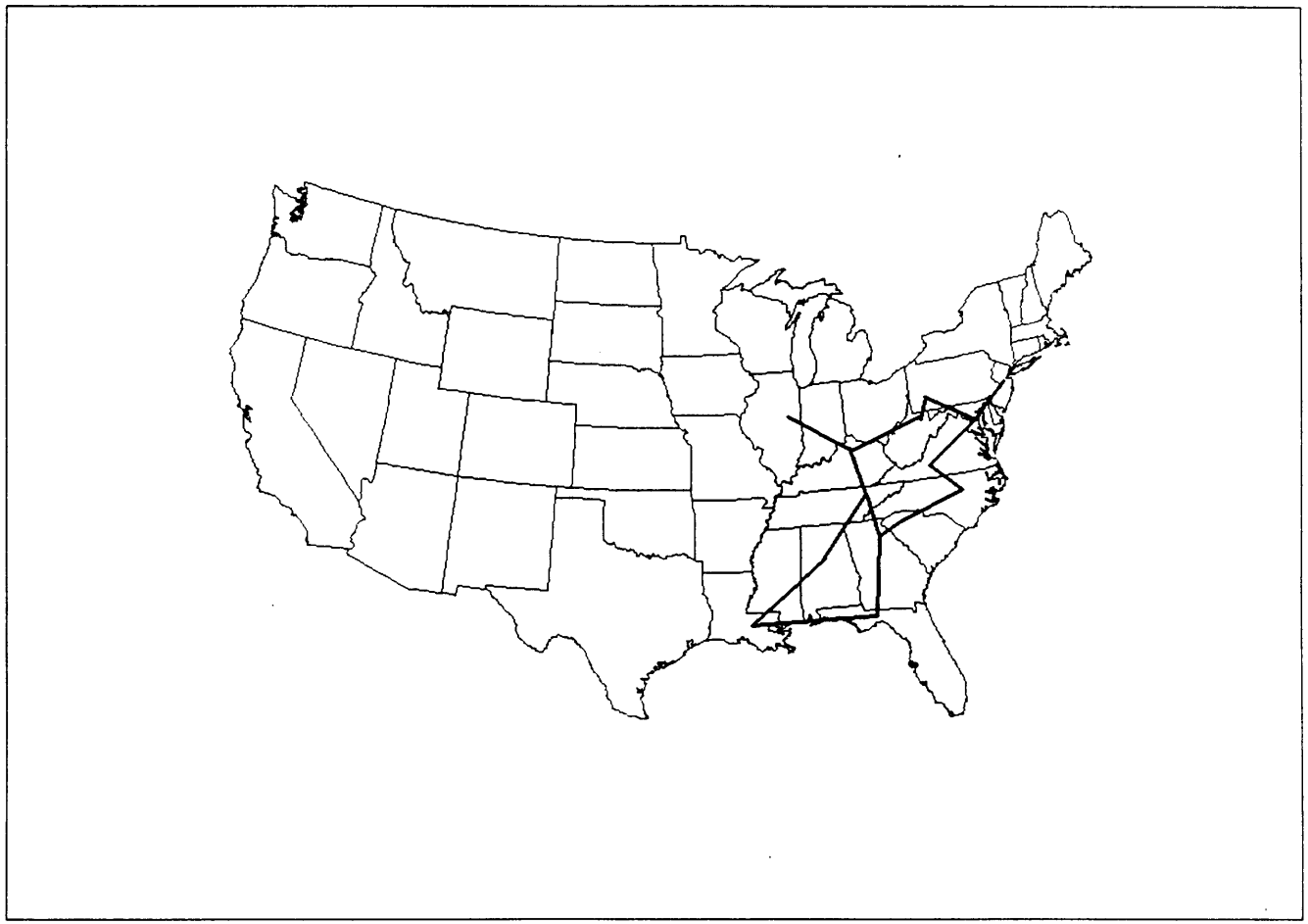
From: <hwb@mcr.umich.edu>
Subject: Princeton Consortium
Date: 20 Jul 86 11:36-EDT



File usa.princeton
usa.princeton read.

Shell Tool 3.0: /bin/csh

From: <hwb@mcr.umich.edu>
Subject: SouthEastern Universities Research Associates Network SURAnet Phase I
Date: 20 Jul 86 12:01-EDT

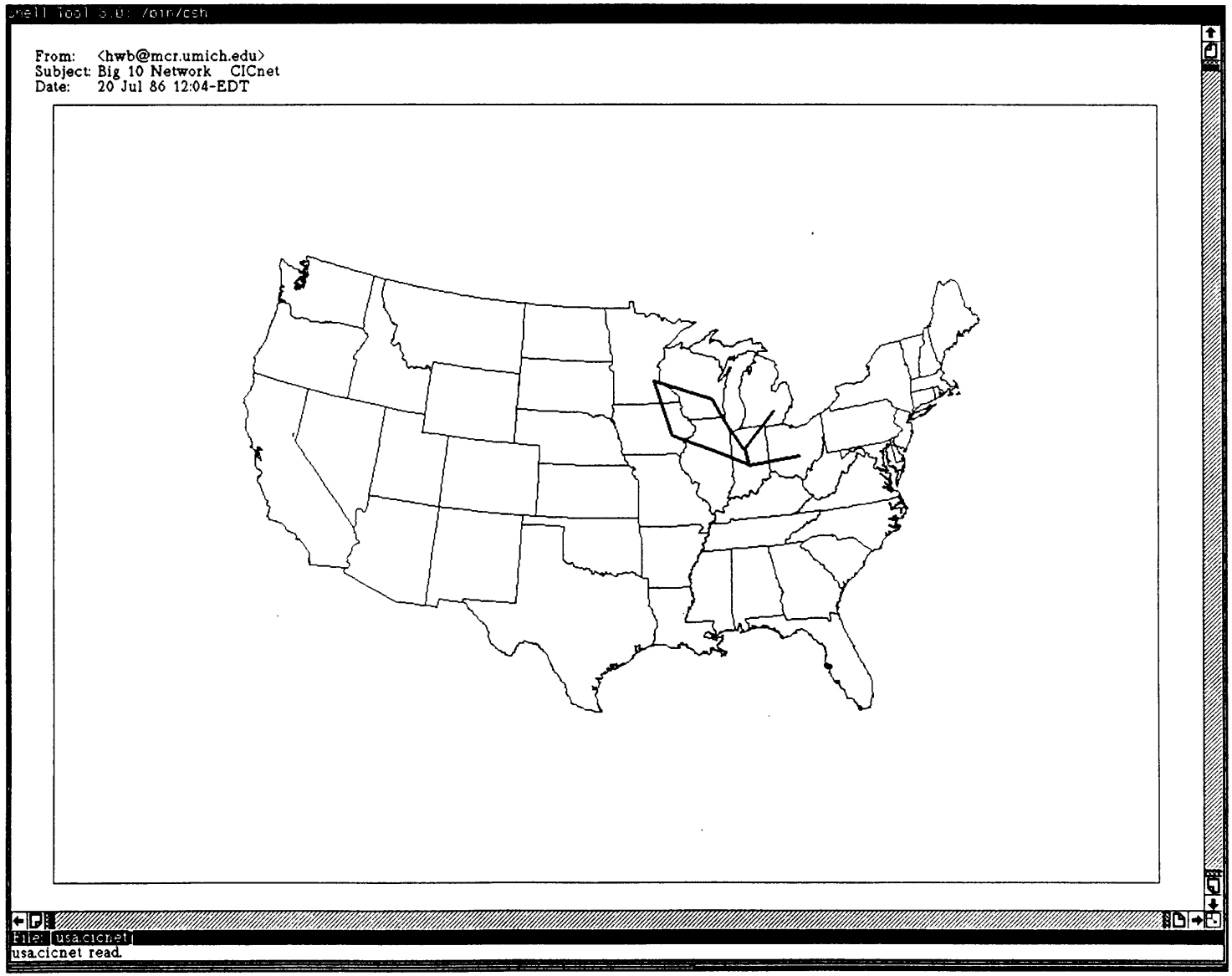


File: usa.sura
usa.sura read.

shell tool 3.0: /bin/csh

From: <hwb@mcr.umich.edu>
Subject: Big 8 Universities Midnet
Date: 20 Jul 86 12:09-EDT

File: usa.midnet
usa.midnet read.

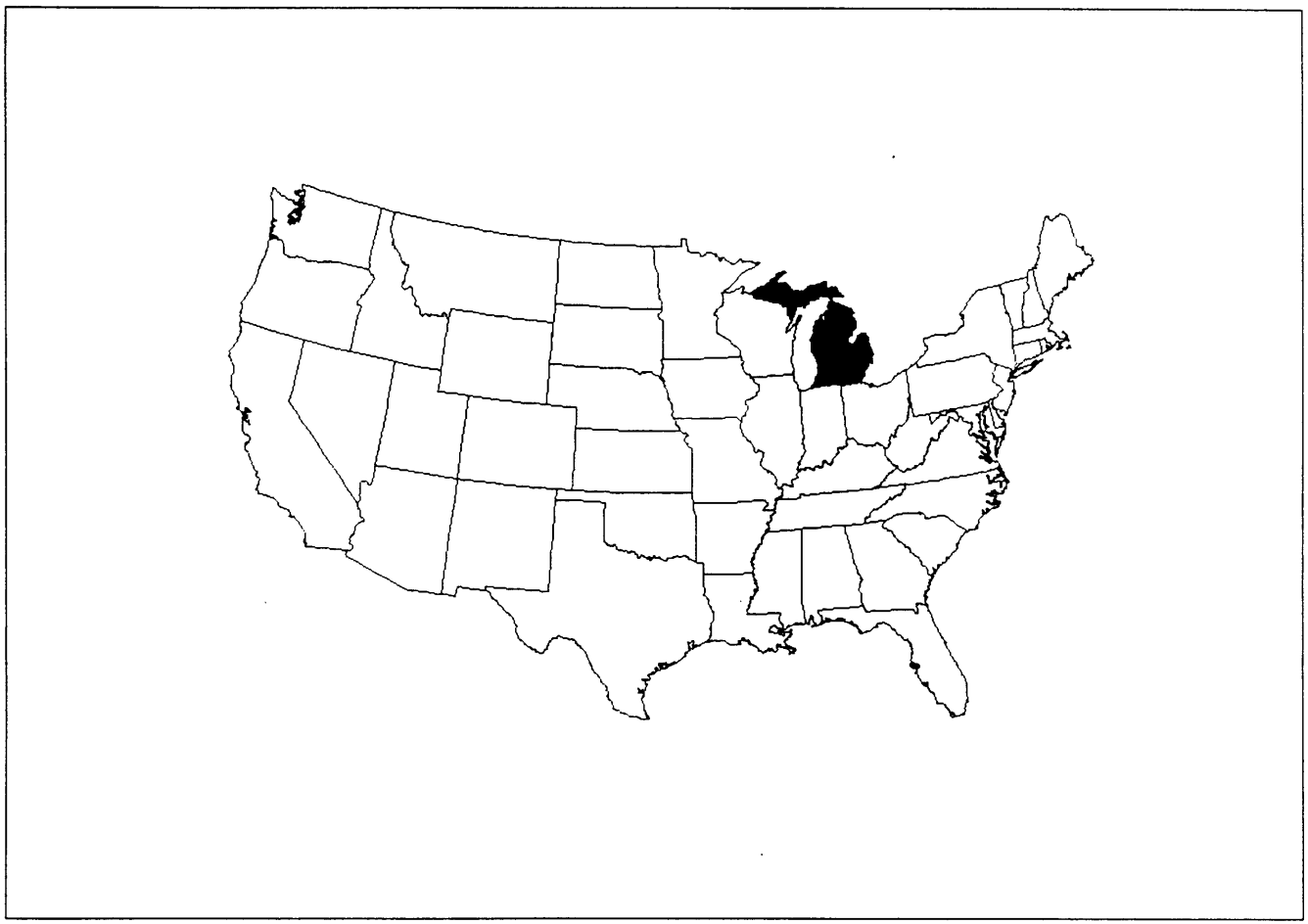

From: <hwb@mcr.umich.edu>
Subject: Big 10 Network CICnet
Date: 20 Jul 86 12:04-EDT
File: usa.cicnet
usa.cicnet read.

From: <hwb@mcr.umich.edu>
Subject: Northern California Research Network NCRnet/BARnet
Date: 20 Jul 86 12:12-EDT

File: [usa.ncrnet]
usa.ncrnet read.

Shell Tool 3.0: /bin/csh

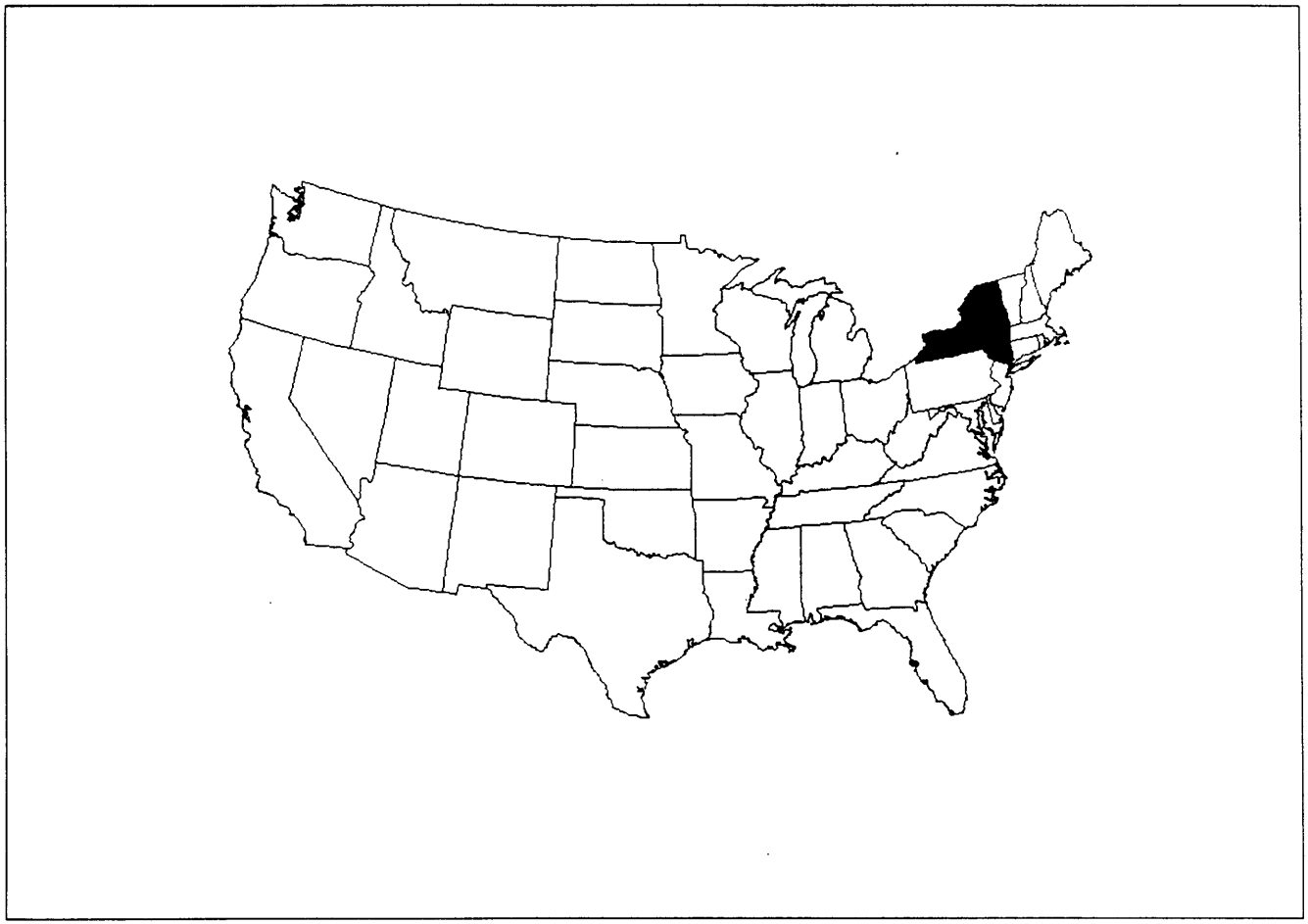
From: <hwb@mcr.umich.edu>
Subject: Merit Computer Network Michigan
Date: 20 Jul 86 12:54-EDT



File usa.merit
usa.merit read.

Shell Tool 3.0: /bin/csh

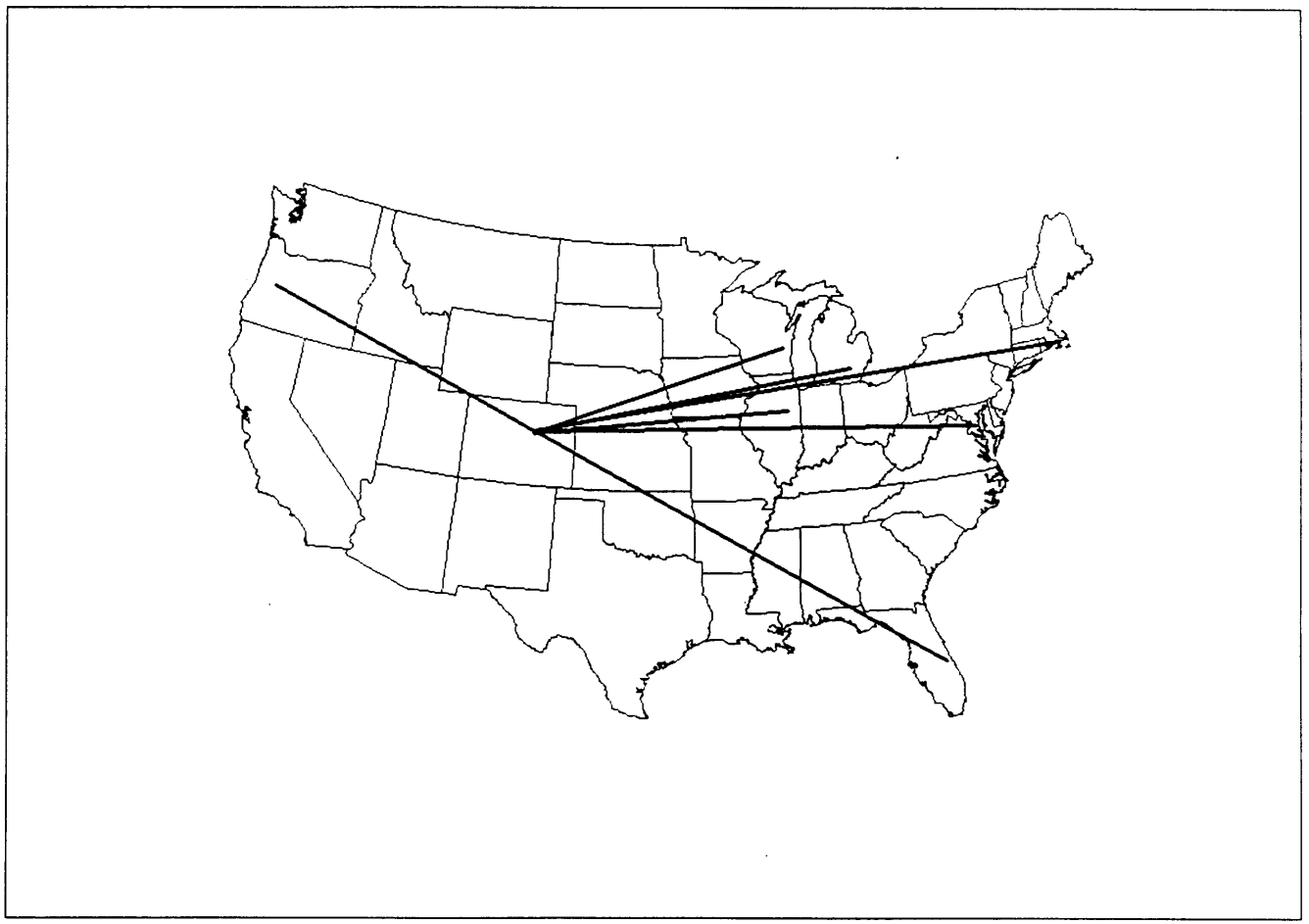
From: <hwb@mcr.umich.edu>
Subject: New York State Network NYSERnet
Date: 20 Jul 86 12:38-EDT



File: usa.nyser
usa.nyser read.

Shell Tool 3.0: /bin/csh

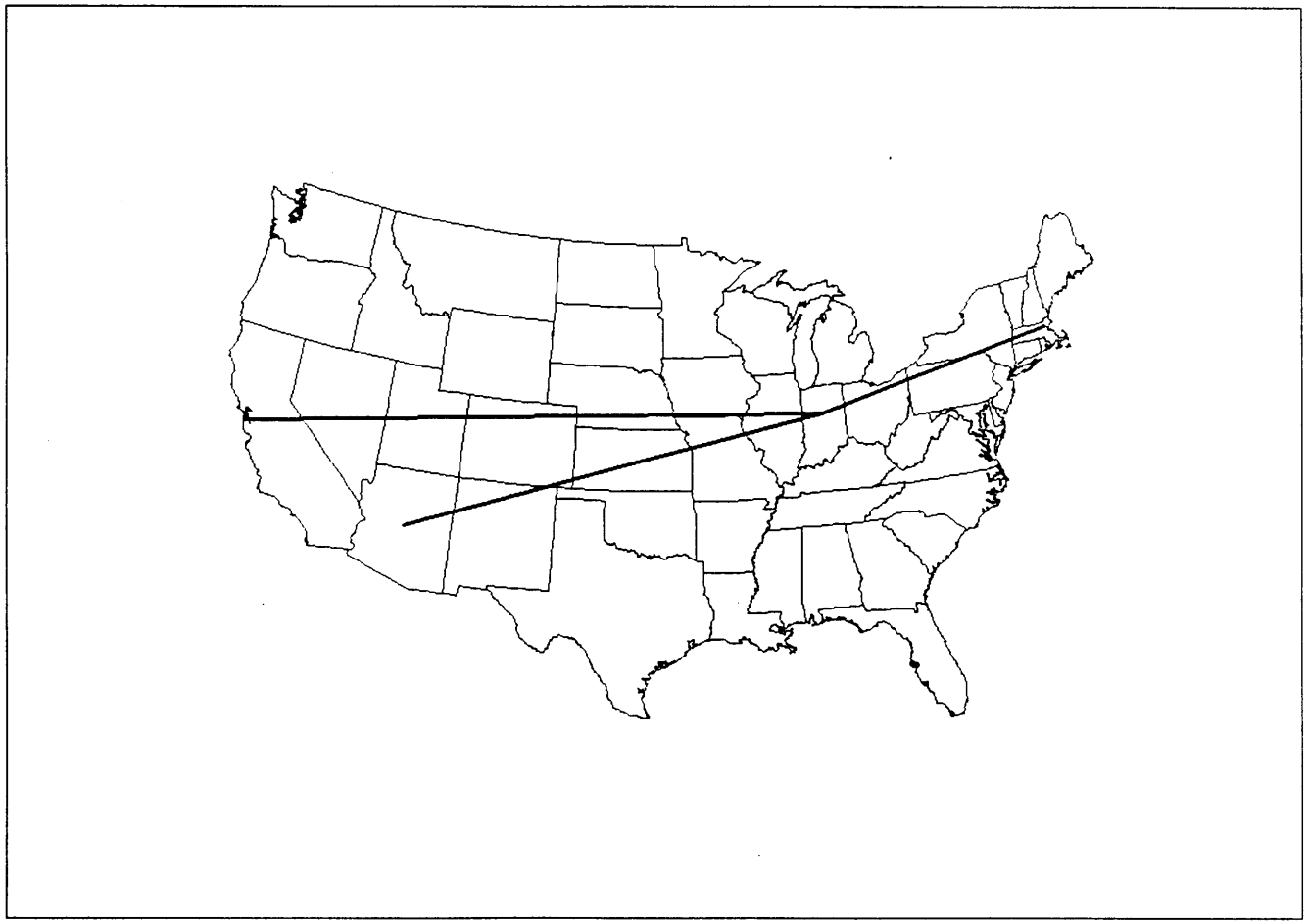
From: <hwb@mcr.umich.edu>
Subject: University SAtellite Network Pilot Project USAN
Date: 20 Jul 86 11:53-EDT



File: [usa.usan]
usa.usan read.

Shell Tool 3.0: /bin/csh

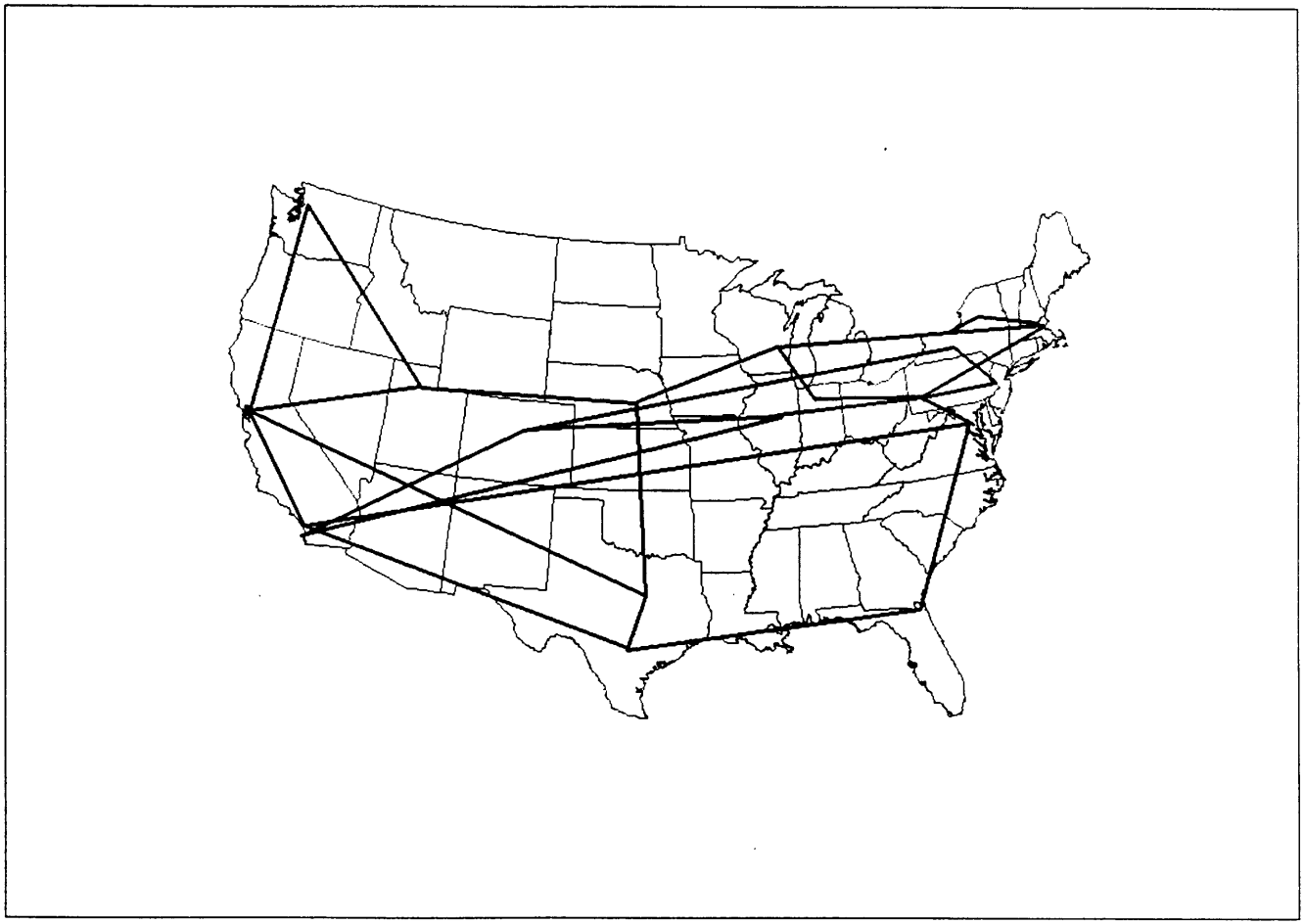
From: <hwb@mcr.umich.edu>
Subject: Cypress
Date: 20 Jul 86 13:01-EDT



File: usacypress1

Shell Tool 3.0: /bin/csh

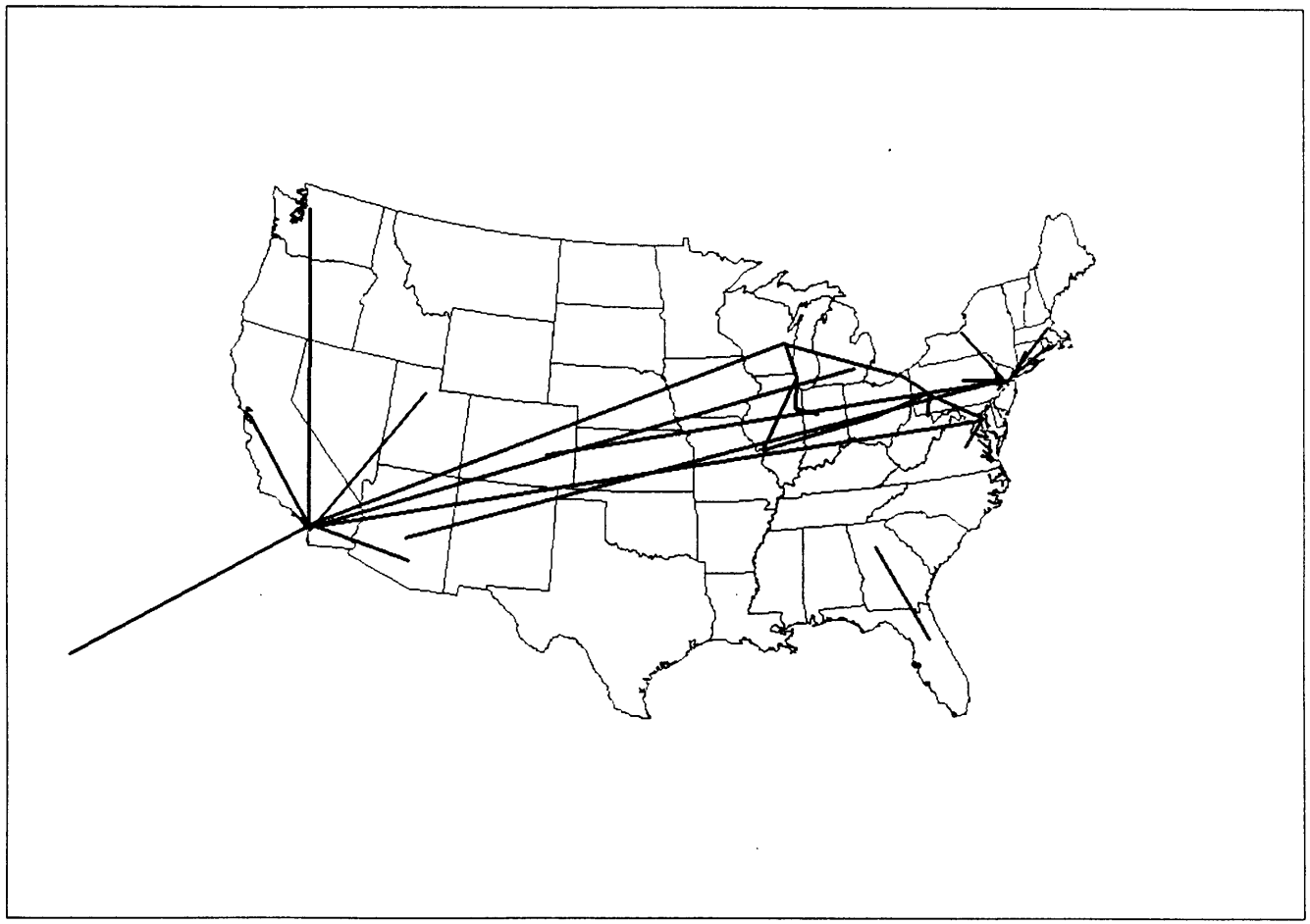
From: <hwb@mcr.umich.edu>
Subject: Expanded Internet backbone
Date: 20 Jul 86 15:23-EDT



File usa.backbone.doc
usa.backbone.doc read.

shell tool 3.0: /bin/csh

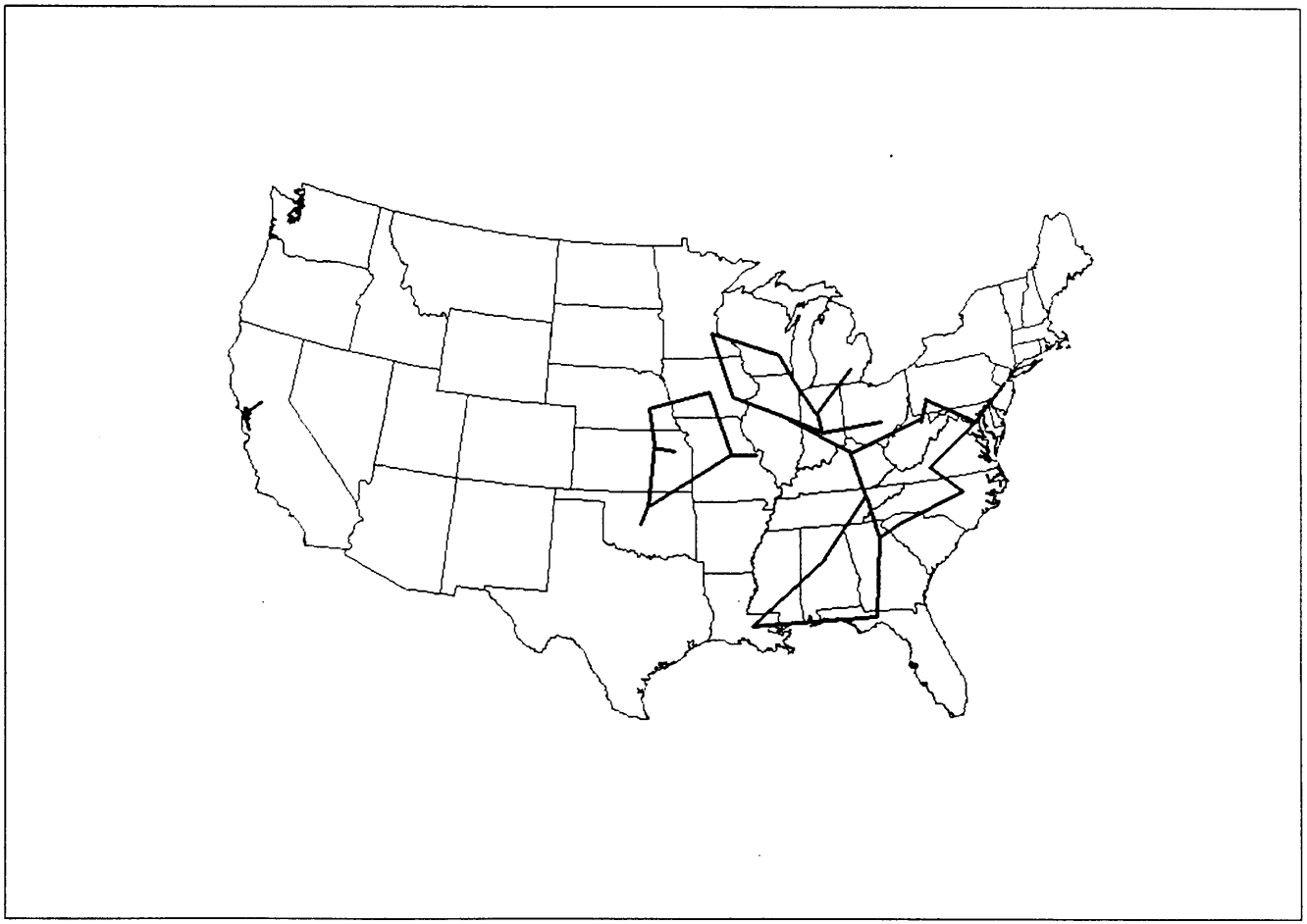
From: <hwb@mcr.umich.edu>
Subject: Supercomputer Consortia Networks
Date: 20 Jul 86 15:37-EDT



File: usa.consortia
usa.consortia read.

Shell Tool 3.0: /bin/csh

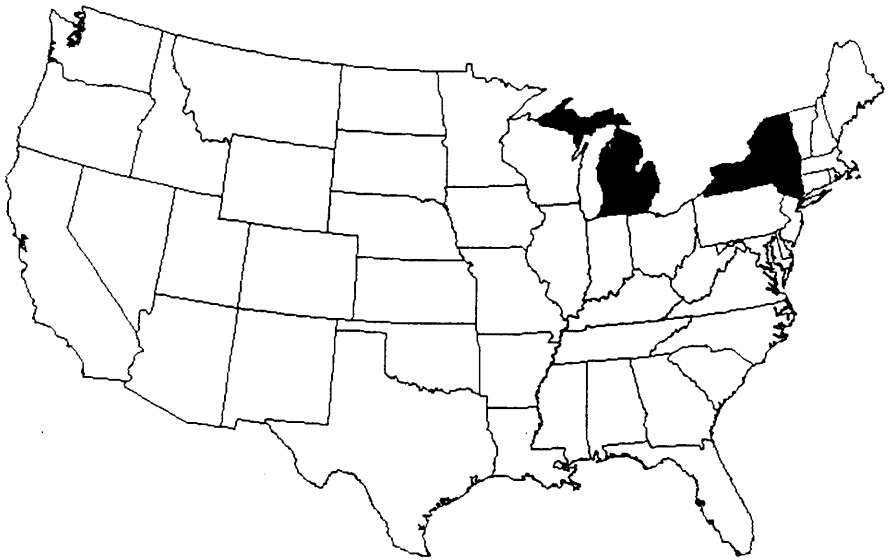
From: <hwb@mcr.umich.edu>
Subject: Regional networks
Date: 20 Jul 86 15:45-EDT



File: usa.regional
usa.regional read.

Shell Tool 3.0: /bin/csh

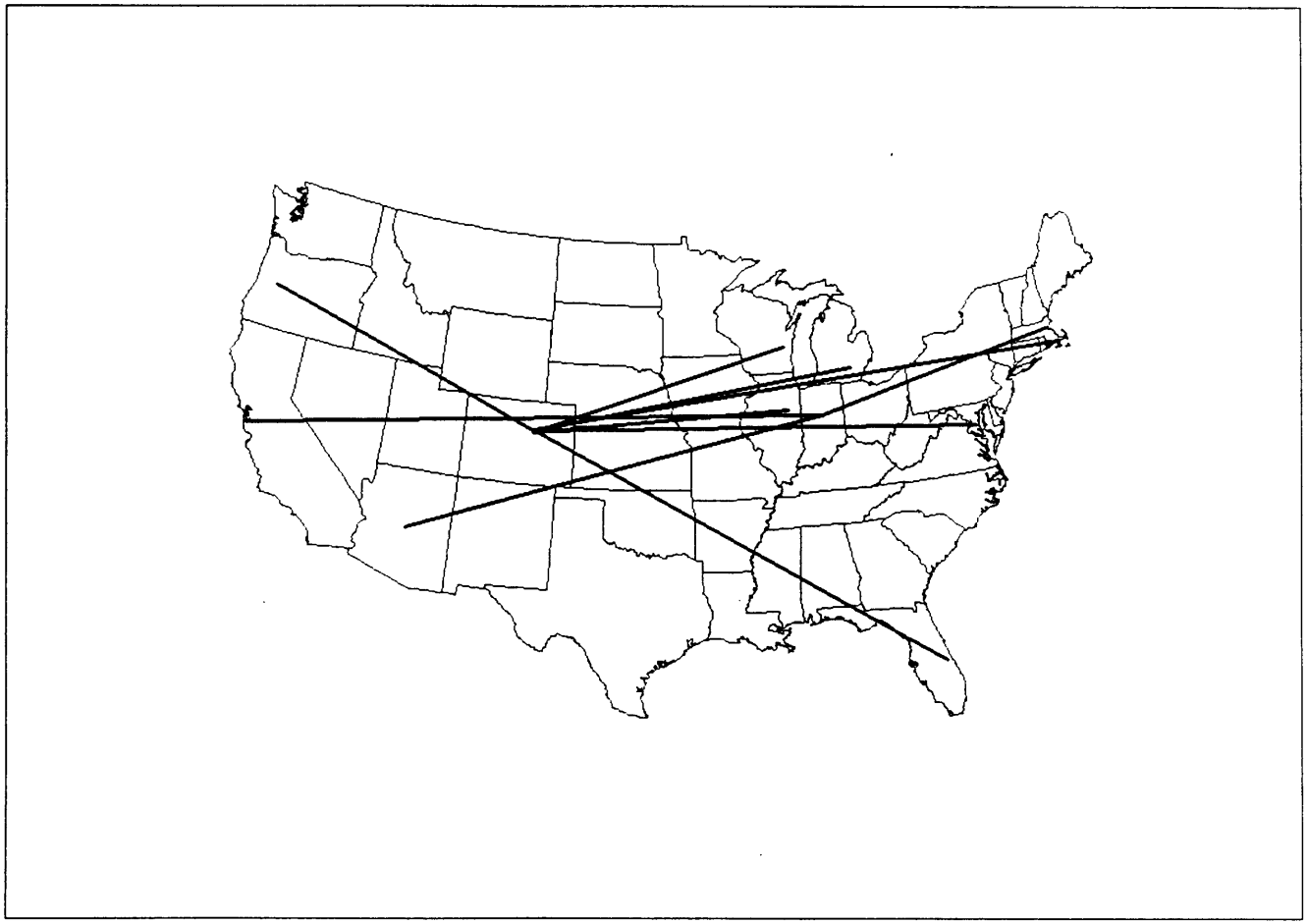
From: <hwb@mcr.umich.edu>
Subject: State networks
Date: 20 Jul 86 15:28-EDT



File: usa.state
usa.state read.

Shell Tool 3.0: /bin/csh

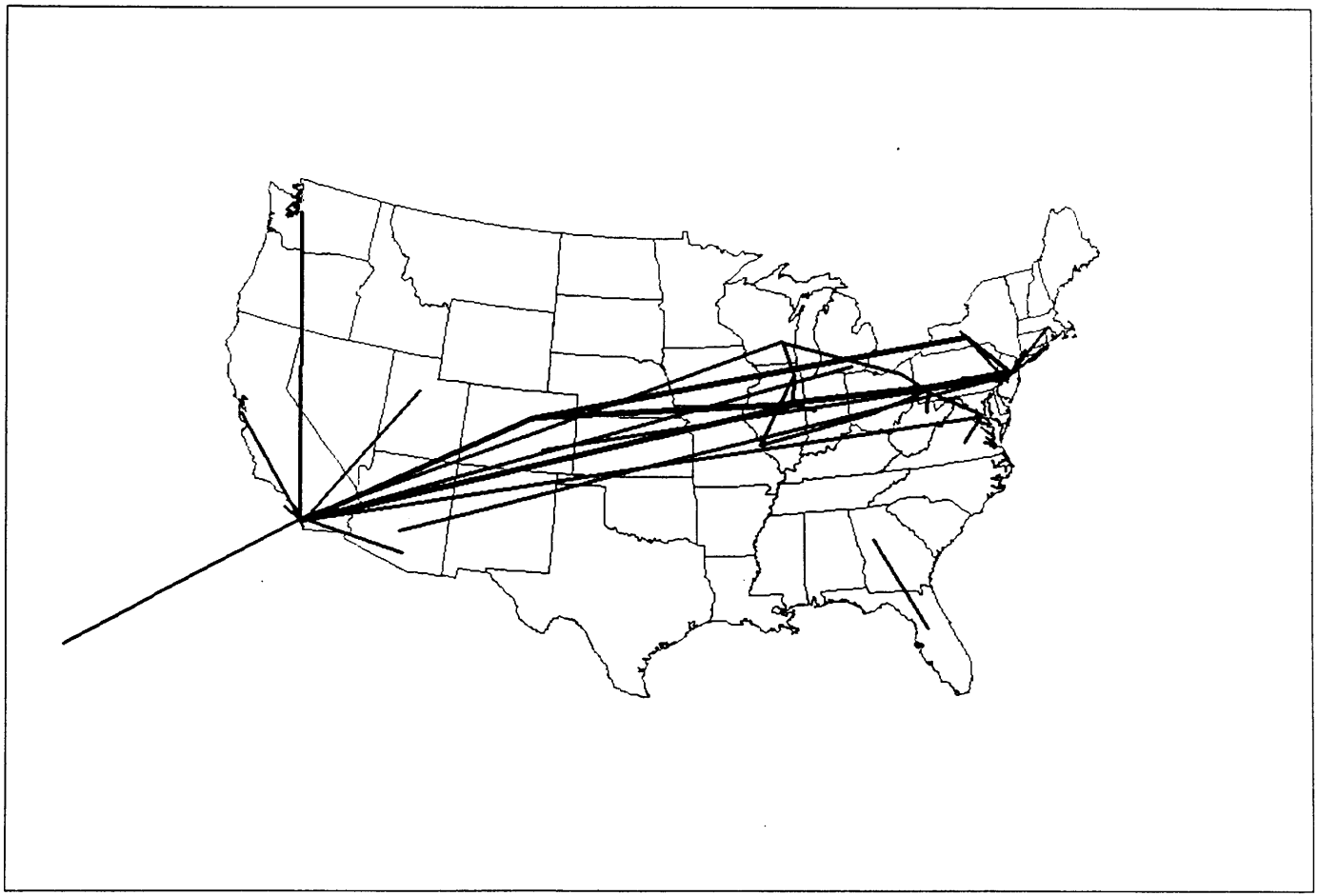
From: <hwb@mcr.umich.edu>
Subject: Pilot projects
Date: 20 Jul 86 15:47-EDT



File: [usa.pilot.doc]

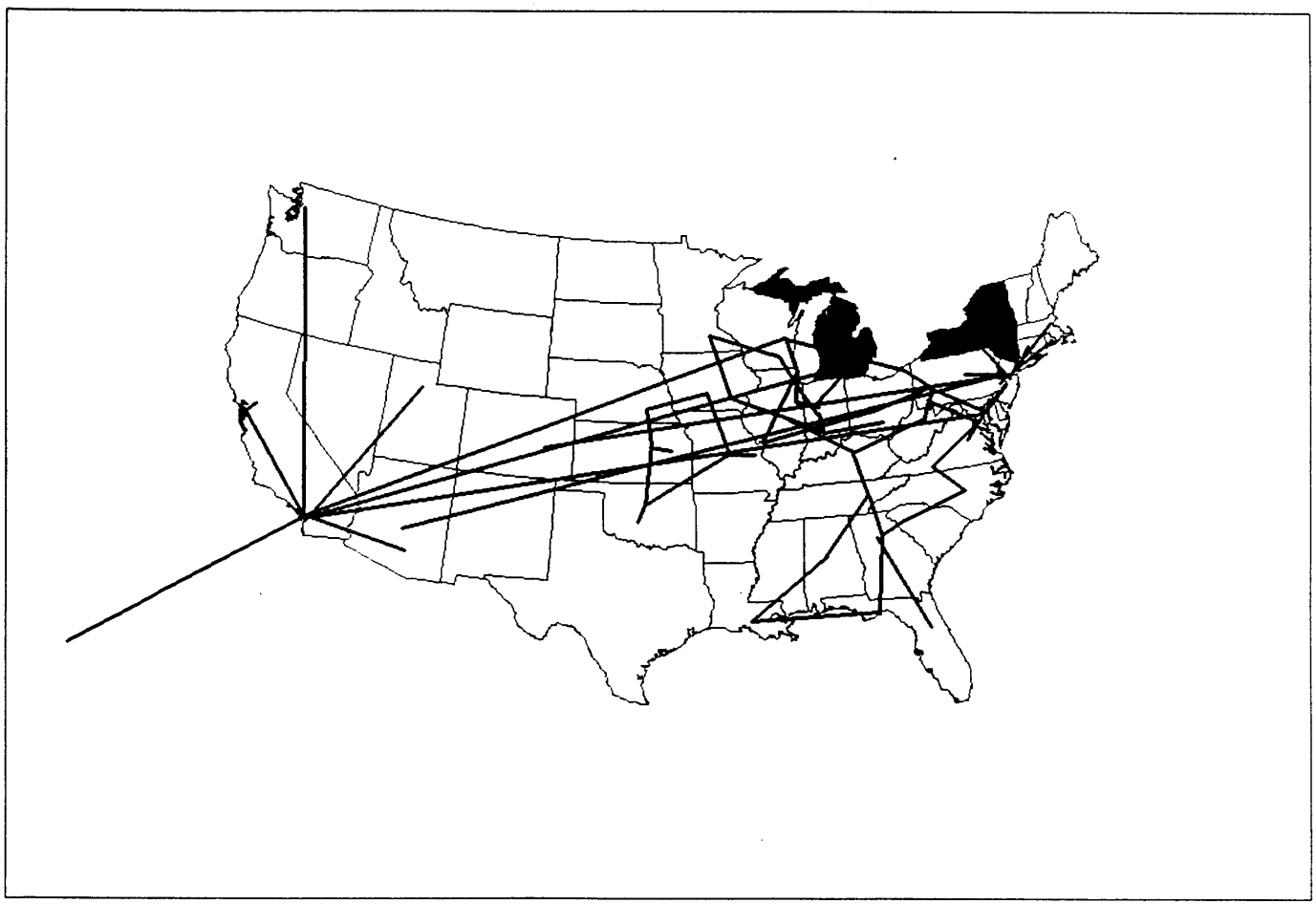
shell tool 3.0: /bin/csh

From: <hwb@mcr.umich.edu>
Subject: NSFnet backbone plus Consortia Networks
Date: 20 Jul 86 15:56-EDT

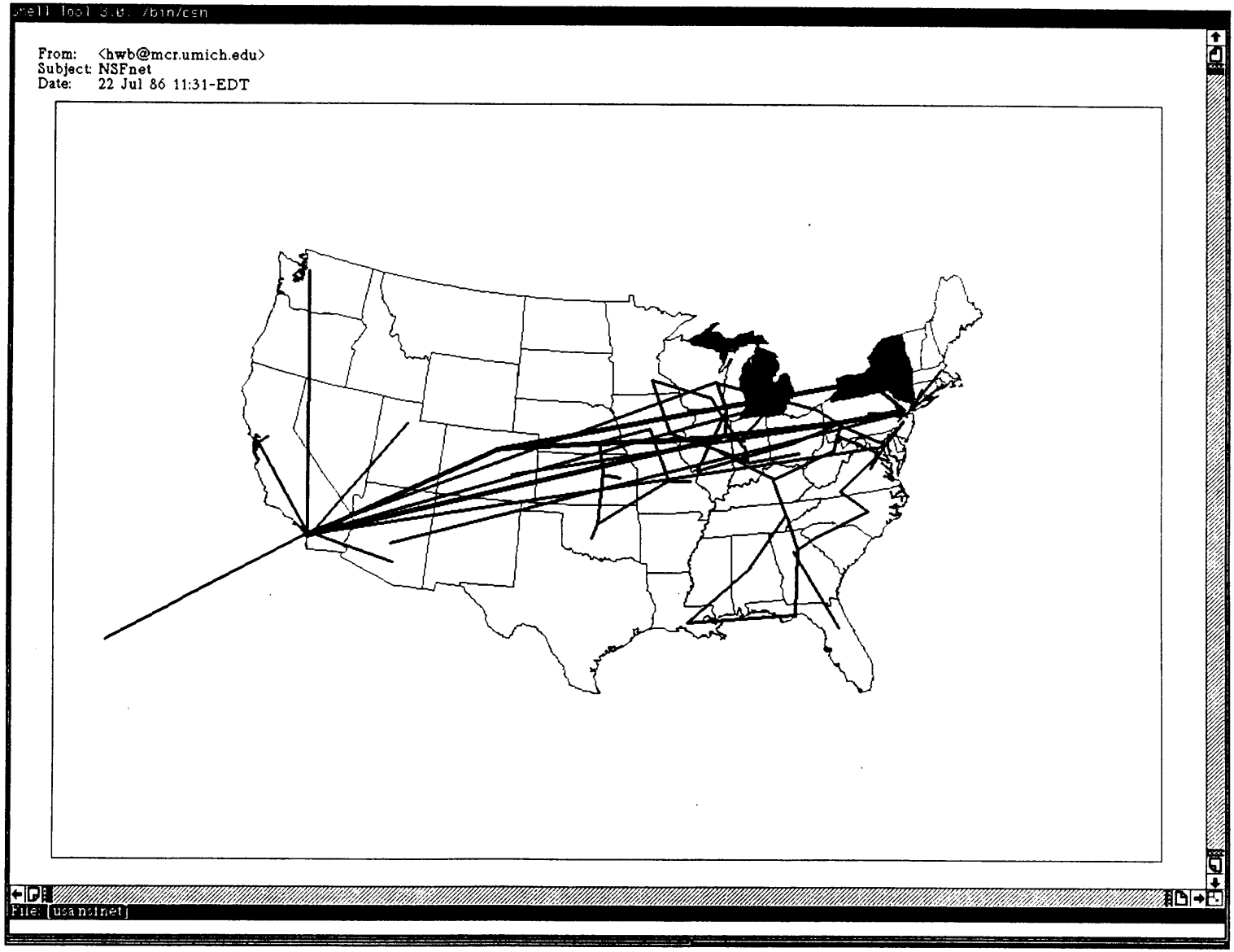


File: usa.backcons
usa.backcons read.

From: <hwb@mcr.umich.edu>
Subject: Non Backbone Nets
Date: 22 Jul 86 11:29-EDT

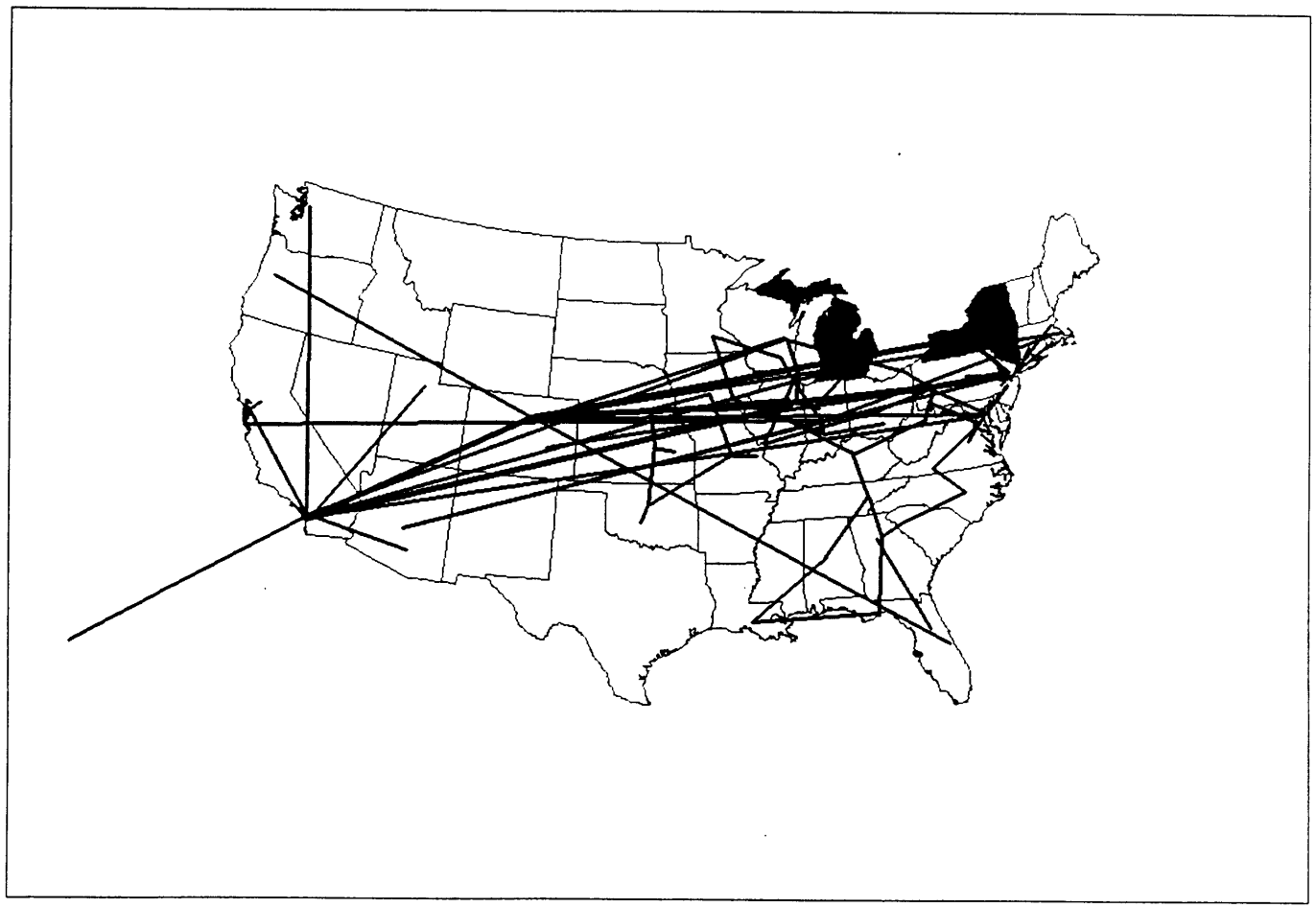


File usa.nonback.doc
usa.nonback.doc read.


From: <hwb@mcr.umich.edu>
Subject: NSFnet
Date: 22 Jul 86 11:31-EDT
File: usanstnet

Shell Tool 3.0: /bin/csh

From: <hwb@mcr.umich.edu>
Subject: NSFnet plus Pilot Projects
Date: 22 Jul 86 11:32-EDT



File: usa.nsfpil
usa.nsfpil read.

shell tool 3.0: /bin/csh

From: <hwb@mcr.umich.edu>
Subject: Expanded Internet: The Arpanet plus the NSFnet
Date: 22 Jul 86 11:33-EDT

File: usa.expinternet
usa.expinternet read.

2) UNIX 4.3 Networking Enhancements -

Mike Karels, UCB

Presentation Slides

4.3BSD Networking

Protocol Support

- multiple address, protocol families
  - Internet (host and port)
  - XNS
  - Unix (on-machine, tied to filesystem)
- multiple protocol families each with suite of protocols (datagram, stream, ...)
- generalized socket abstraction, IPC interface
- wide variety of hardware supported, multiple protocols/interface, multiple interfaces/host
- fairly well-defined internal interfaces
  - socket-protocol
  - protocol-protocol
  - protocol-link layer

4.3 Changes in Internet addressing Subnets

- RFC-950 subnet support:
  - network mask per network interface (by default, 8/16/24-bit network part only)
  - Within subnetted net, all subnets are treated as separate networks. (
  - ICMP address mask request support
  - subnet broadcasts (local wire) only

Broadcast addressing

- RFC-919: host part of all ones (issued since release of 4.2, which used host all zeros)

- 4.3 uses host of all ones as default broadcast address
  - may be set at run time per network
  - subnet broadcasts are net.subnet.ones unless set otherwise
  - All of the following are accepted on input:

    net.subnet.ones

    net.subnet.zero

    net.ones

    net.zeros

    all ones

    all zeros

Changes in IP

- IP options:
  - supported on output, per connection (setsockopt)
  - source routing understood
  - options updated when forwarded (4.2 updated, then removed)
  - IP saves source route on input, TCP uses for reply

Changes in IP

- IP forwarding
  - No forwarding or error reply from host with single interface
  - ICMP redirect sent when forwarding on incoming interface:
    - only if source on attached net
    - host redirect if routed by host or subnet of non-shared net
    - network redirect otherwise
    - ICMP errors addressed according to incoming interface
  - one-element route cache

Additional network changes

- Source addresses chosen according to outgoing network (4.2 used "primary" address if destination non-local).

- Identity of receiving interface recorded with incoming packets
  - ICMP needs for info request, mask request
  - ICMP uses for source address selection on errors
  - IP needs for redirect generation
  - needed to requeue IMP error messages
  - needed by Xerox NS

TCP changes

- round-trip timer fixed
- retransmissions not timed after connection
- only one segment sent when retransmitting
- faster backoff after first retransmission

- offered window never shrinks
- state is reset if peer shrinks window

- connections in FIN_WAIT_2 time out after user has closed
- new data not accepted after user close

TCP changes

- maximum segment size selection–
  - 4.2 always offered, accepted 1024-octet segments
  - 4.3 offers, accepts convenient size near interface packet size
  - if destination non-local (not on subnet of local net), default size is used (512 data + header)

Changes in TCP send policy

- sender silly-window syndrome avoidance fixed (was relative to receiver's window)
- persist logic fixed to handle non-zero window
- small packet avoidance– small sends accumulate during round-trip time (Nagle algorithm)
- larger send, receive buffers (4096 bytes) improves delayed acknowledgement performance
- source quench handling– decrease amount of outstanding data (Nagle)

Address Resolution Protocol in 4.3

- generalized for other protocols
  (still supports only 10 Mb/s Ethernet)

- ARP rejects and logs hosts with address that is the hardware broadcast.

- Most-recently received Ethernet address is used; allows hosts to recover after change of hardware address.

Address Resolution Protocol in 4.3

- ARP used to negotiate use of trailer encapsulations:
  - ARP trailer reply sent together with IP reply
  - ARP trailer reply sent in response to IP reply
  - either host may request to receive trailers
  - other hosts receive only normal packet encapsulation

4.2/4.3 Routing

- separation of policy and mechanism
  - kernel implements mechanism
  - intelligent routing process determines policy
  - simple hosts may be simple

- Kernel first-hop routing

  1. look for route to host
  2. look for route to network
  3. look for wildcard route (to default gateway)

- route may be direct (on attached net), or may contain gateway address.

Changes in Routing– Generic

- Kernel uses first route found in table (4.2 used route with lowest use count)
- Initial routes to connected nets installed by protocols, not link layer
- Old route is deleted if interface address is changed.
- Variable-sized table (larger with option GATEWAY). Hash uses mask rather than (signed) modulus.

Changes in Routing– Internet

- Fixes in ICMP redirect handling.
- An ICMP host redirect will install a new route to host without modifying route to network.
- Redirects, routing changes cause notification to protocols.
  - Current connections flush route cache, reroute on next send.
- Redirects are accepted only from current router.

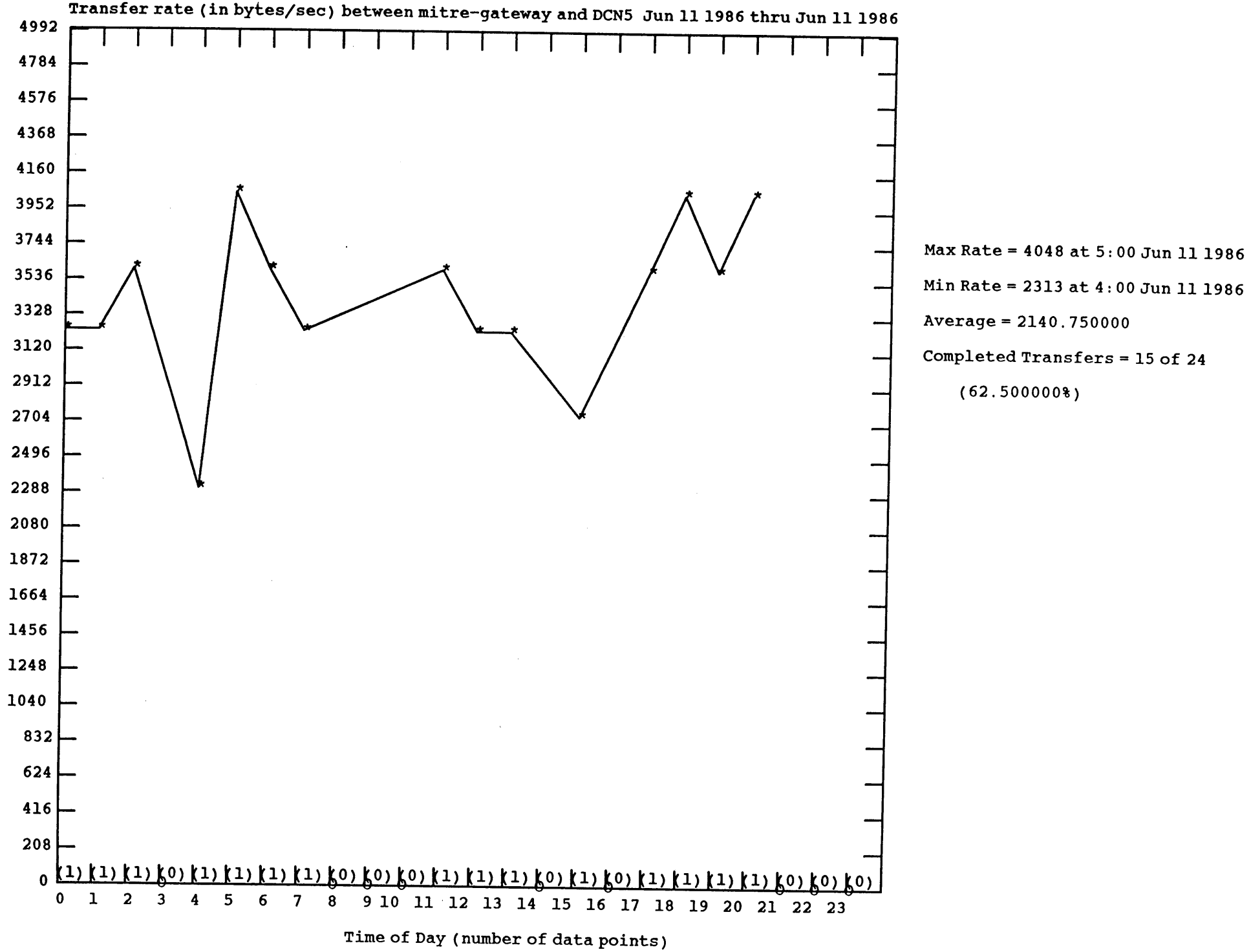
Routed (Unix routing daemon)

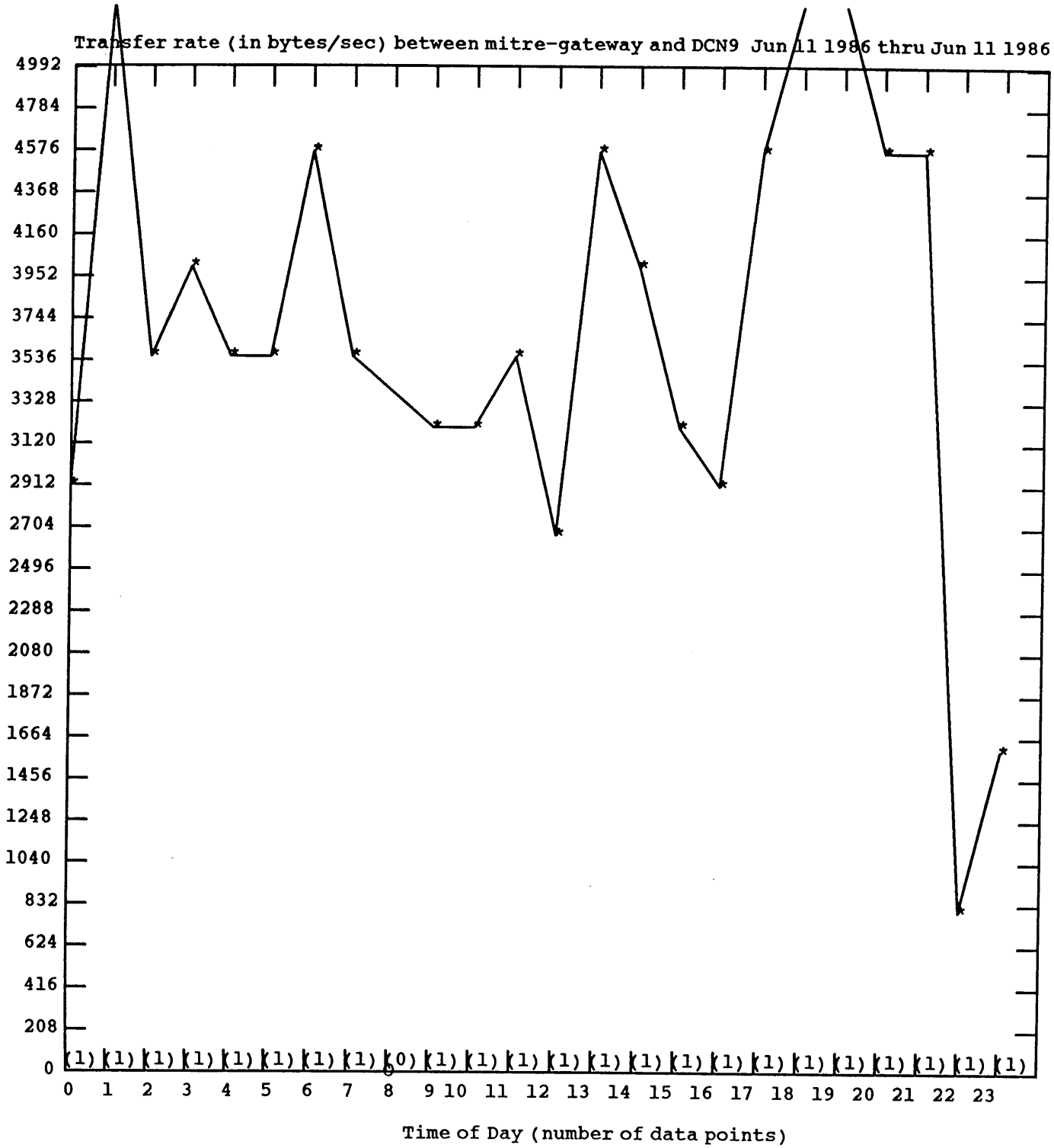
- Based on Xerox Routing Information Protocol
- gateways broadcast destination and hop count for known routes
- Changes in 4.3:
  - doesn't send external routing information (only internal routes plus a wildcard)
  - better support of point-point links
  - subnet support
  - subnet routes not sent outside of network
  - routing information not repeated on incoming network

3) Internet Performance Report -

Phill Gross, MITRE

Presentation Slides


Transfer rate (in bytes/sec) between mitre-gateway and DCN5 Jun 11 1986 thru Jun 11 1986
4992
4784
4576
4368
4160
3952
3744
3536
3328
3120
2912
2704
2496
2288
2080
1872
1664
1456
1248
1040
832
624
416
208
0
(1) (1) (1) (0) (1) (1) (1) (1) (0) (0) (0) (1) (1) (1) (0) (1) (0) (1) (1) (1) (1) (0) (0) (0)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Time of Day (number of data points)
Max Rate = 4048 at 5:00 Jun 11 1986
Min Rate = 2313 at 4:00 Jun 11 1986
Average = 2140.750000
Completed Transfers = 15 of 24
(62.500000%)



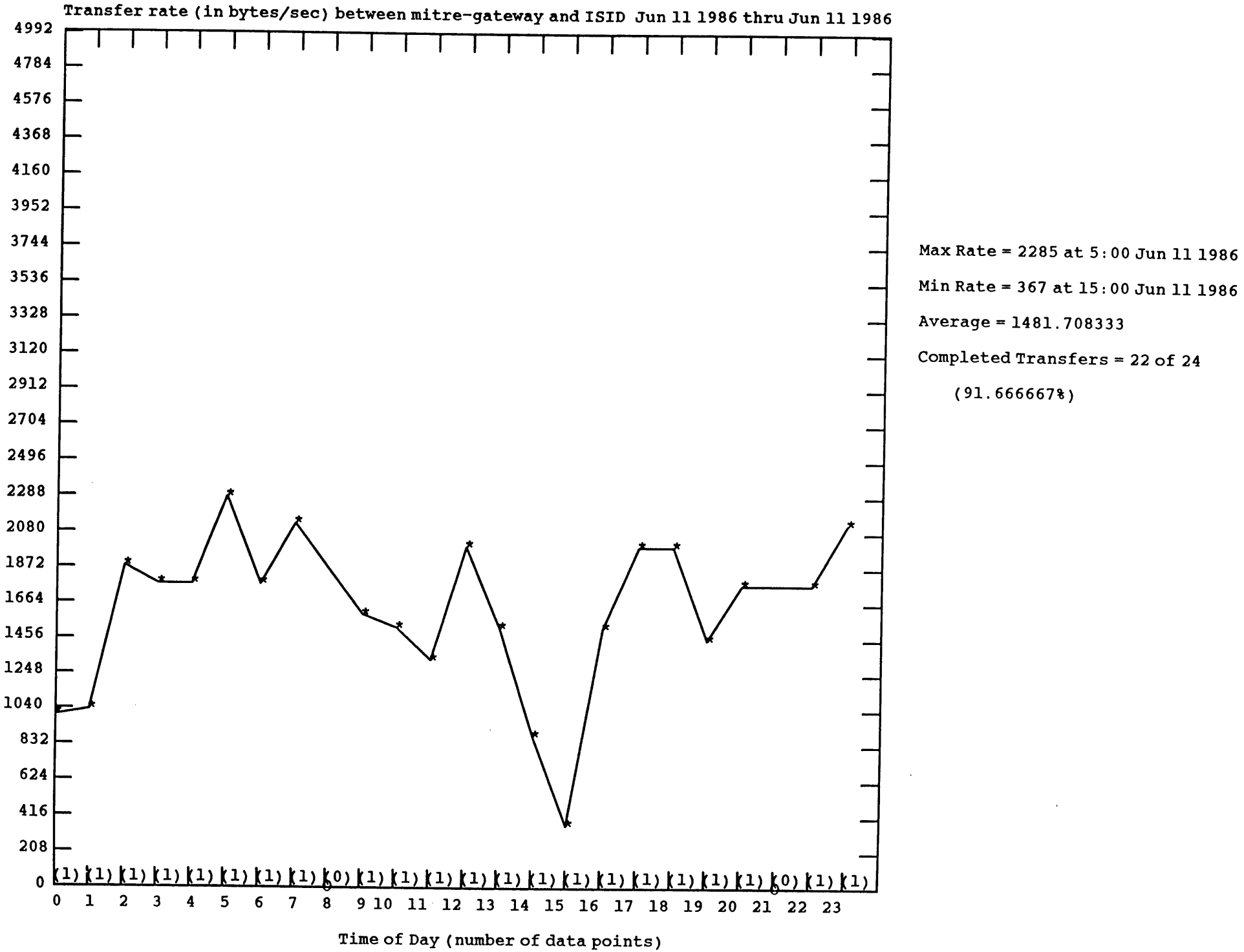
Max Rate = 5333 at 1:00 Jun 11 1986

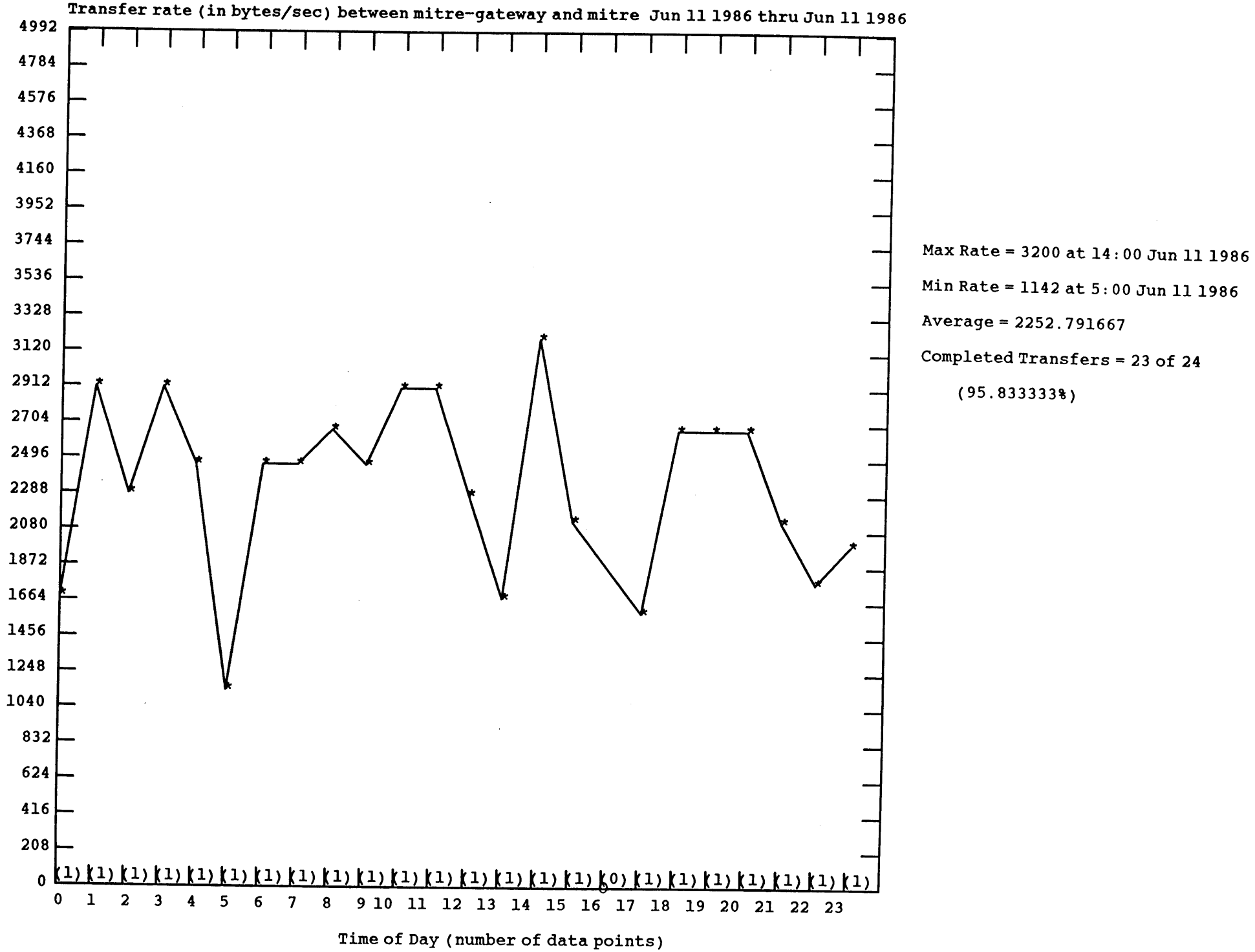
Min Rate = 800 at 22:00 Jun 11 1986

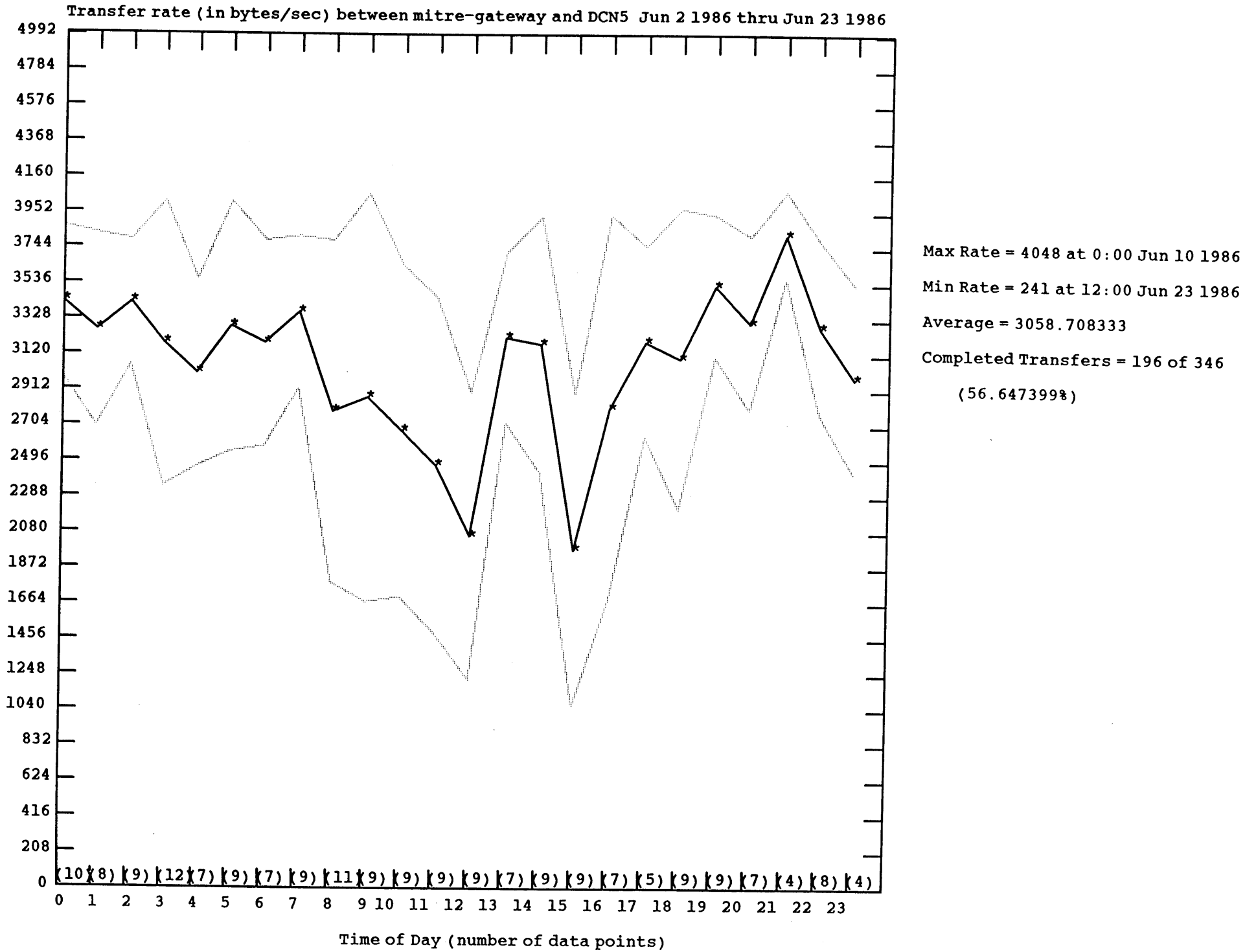
Average = 3546.375000

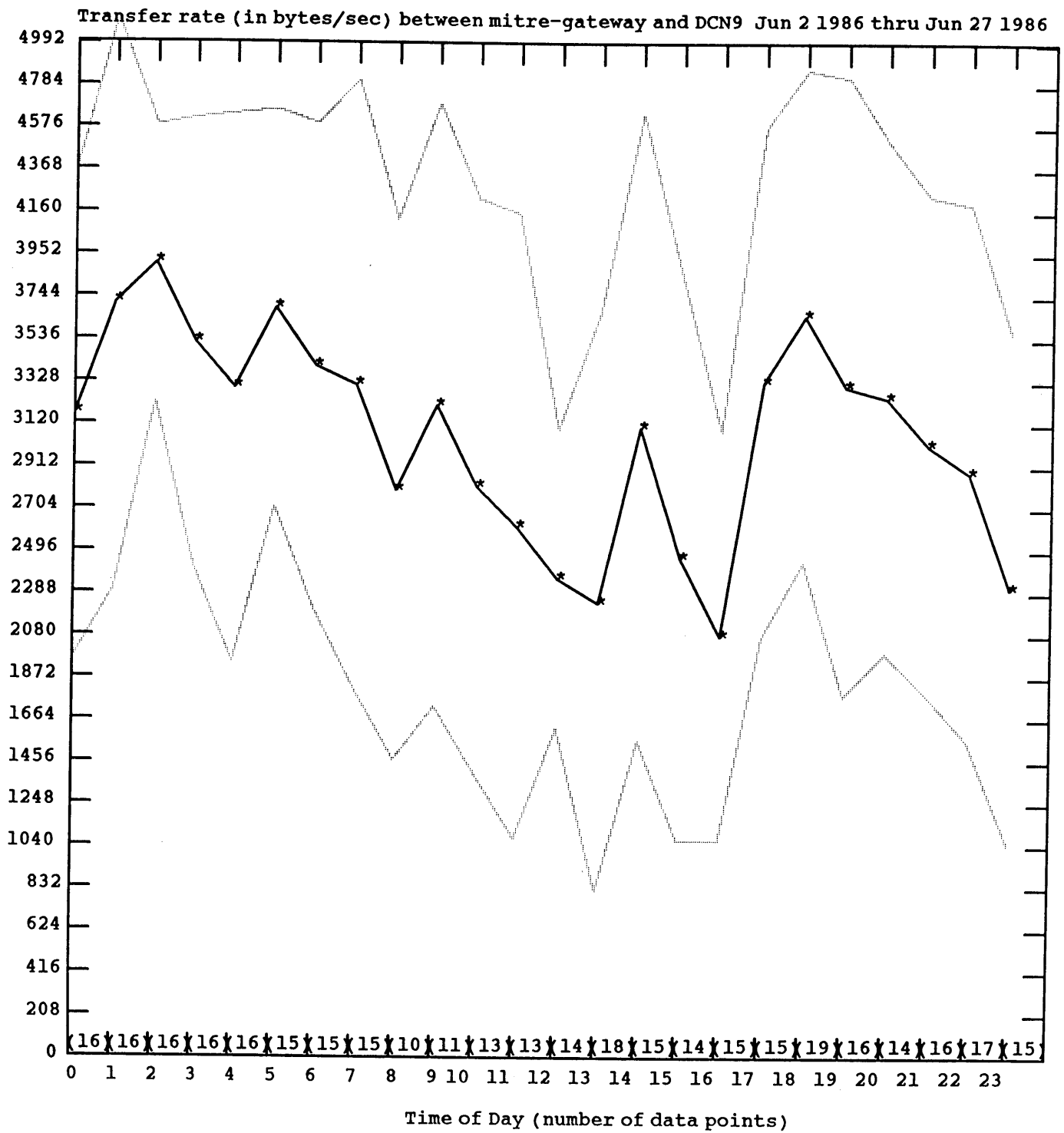
Completed Transfers = 23 of 24

(95.833333%)


Transfer rate (in bytes/sec) between mitre-gateway and ISID Jun 11 1986 thru Jun 11 1986
4992
4784
4576
4368
4160
3952
3744
3536
3328
3120
2912
2704
2496
2288
2080
1872
1664
1456
1248
1040
832
624
416
208
0
(1) (1) (1) (1) (1) (1) (1) (1) (0) (1) (1) (1) (1) (1) (1) (1) (1) (1) (1) (1) (1) (1) (0) (1) (1)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Time of Day (number of data points)
Max Rate = 2285 at 5:00 Jun 11 1986
Min Rate = 367 at 15:00 Jun 11 1986
Average = 1481.708333
Completed Transfers = 22 of 24
(91.666667%)


Transfer rate (in bytes/sec) between mitre-gateway and mitre Jun 11 1986 thru Jun 11 1986
4992
4784
4576
4368
4160
3952
3744
3536
3328
3120
2912
2704
2496
2288
2080
1872
1664
1456
1248
1040
832
624
416
208
0
(1) (1) (1) (1) (1) (1) (1) (1) (1) (1) (1) (1) (1) (1) (1) (1) (1) (0) (1) (1) (1) (1) (1) (1) (1)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Time of Day (number of data points)
Max Rate = 3200 at 14:00 Jun 11 1986
Min Rate = 1142 at 5:00 Jun 11 1986
Average = 2252.791667
Completed Transfers = 23 of 24
(95.833333%)


Transfer rate (in bytes/sec) between mitre-gateway and DCN5 Jun 2 1986 thru Jun 23 1986
4992
4784
4576
4368
4160
3952
3744
3536
3328
3120
2912
2704
2496
2288
2080
1872
1664
1456
1248
1040
832
624
416
208
0
(10) (8) (9) (12) (7) (9) (7) (9) (11) (9) (9) (9) (9) (7) (9) (9) (7) (5) (9) (9) (7) (4) (8) (4)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Time of Day (number of data points)
Max Rate = 4048 at 0:00 Jun 10 1986
Min Rate = 241 at 12:00 Jun 23 1986
Average = 3058.708333
Completed Transfers = 196 of 346
(56.647399%)



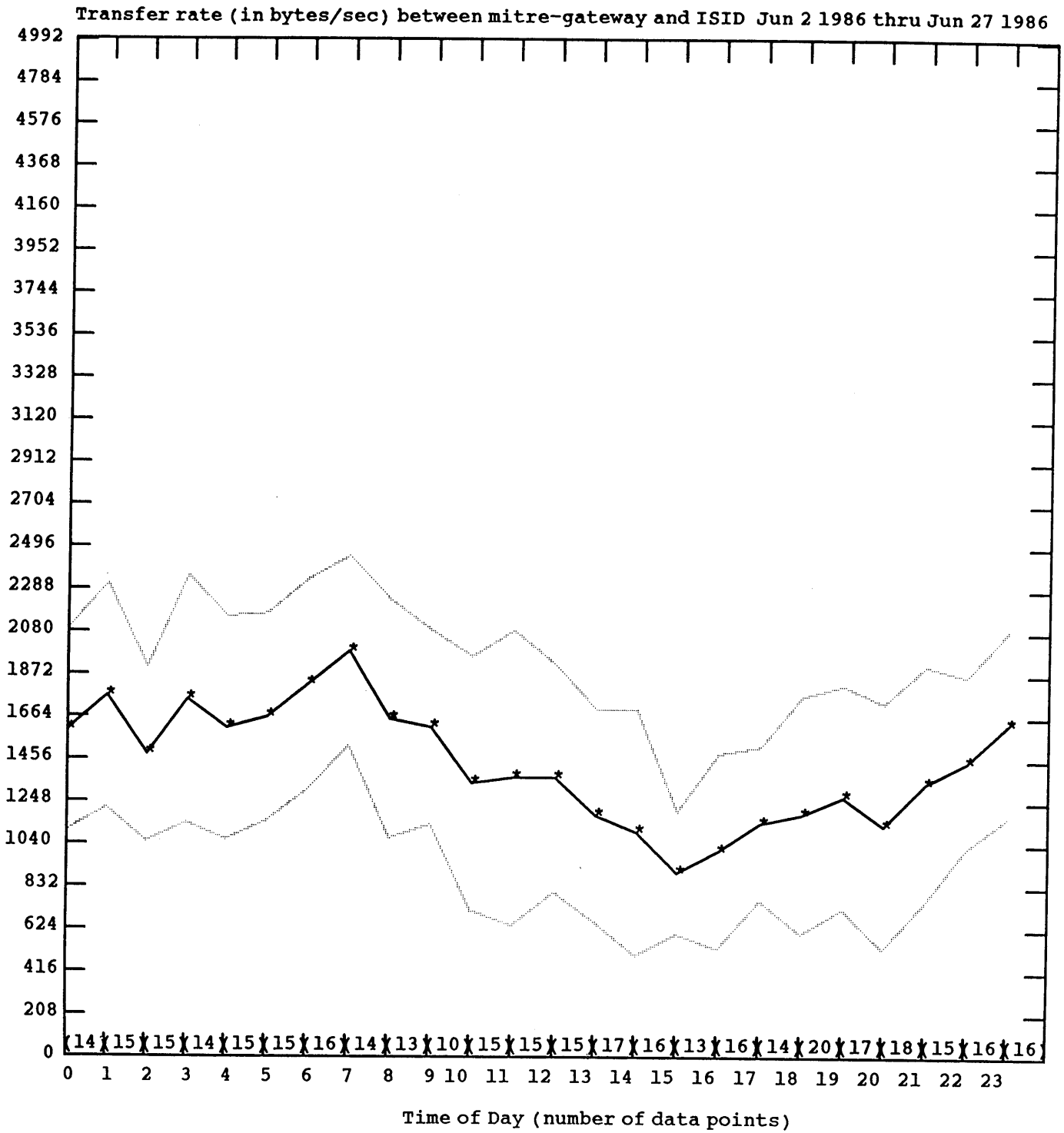
Max Rate = 5333 at 5:00 Jun 10 1986

Min Rate = 31 at 22:00 Jun 5 1986

Average = 3059.541667

Completed Transfers = 360 of 407

(88.452088%)



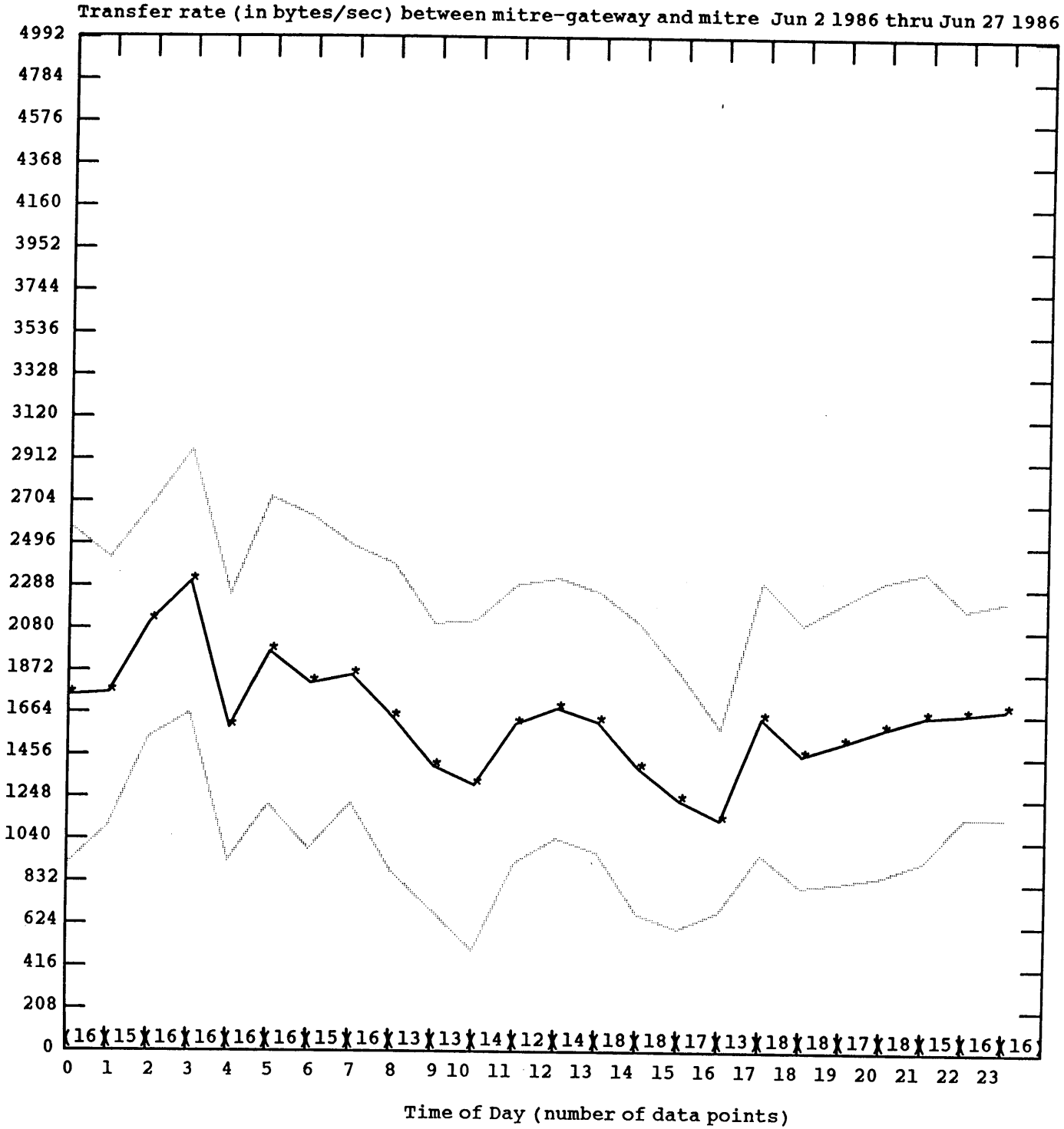
Max Rate = 2666 at 1:00 Jun 16 1986

Min Rate = 137 at 14:00 Jun 24 1986

Average = 1431.375000

Completed Transfers = 364 of 412

(88.349515%)



Max Rate = 3200 at 14:00 Jun 11 1986

Min Rate = 26 at 10:00 Jun 23 1986

Average = 1643.125000

Completed Transfers = 376 of 408

(92.156863%)

4) Internet Performance Report -

Marianne Gardner, BBN

Presentation Slides

Mailbridge	FEB-86	MAR-86
MILARPA	9.0	10.05
MILBBN	13.8	15.1
MILDCEC	7.4	8.5
MILISI	8.6	9.3
MILLBL	6.4	6.9
MILSAC	5.8	6.6
MILSRI	4.3	4.9

Mailbridge	APR-86	MAY-86
MILARPA	5.6	8.1
MILBBN	11.4	13.2
MILDCEC	8.0	9.3
MILISI	9.4	9.8
MILLBL	6.1	6.6
MILSAC	6.3	7.1
MILSRI	4.1	5.2

Mailbridge	JUNE-86	JUL-86*
MILARPA	5.8	5.6
MILBBN	11.5	11.4
MILDCEC	8.4	7.7
MILISI	10.6	9.7
MILLBL	6.1	5.4
MILSAC	7.3	7.0
MILSRI	5.0	4.2

*1st three weeks

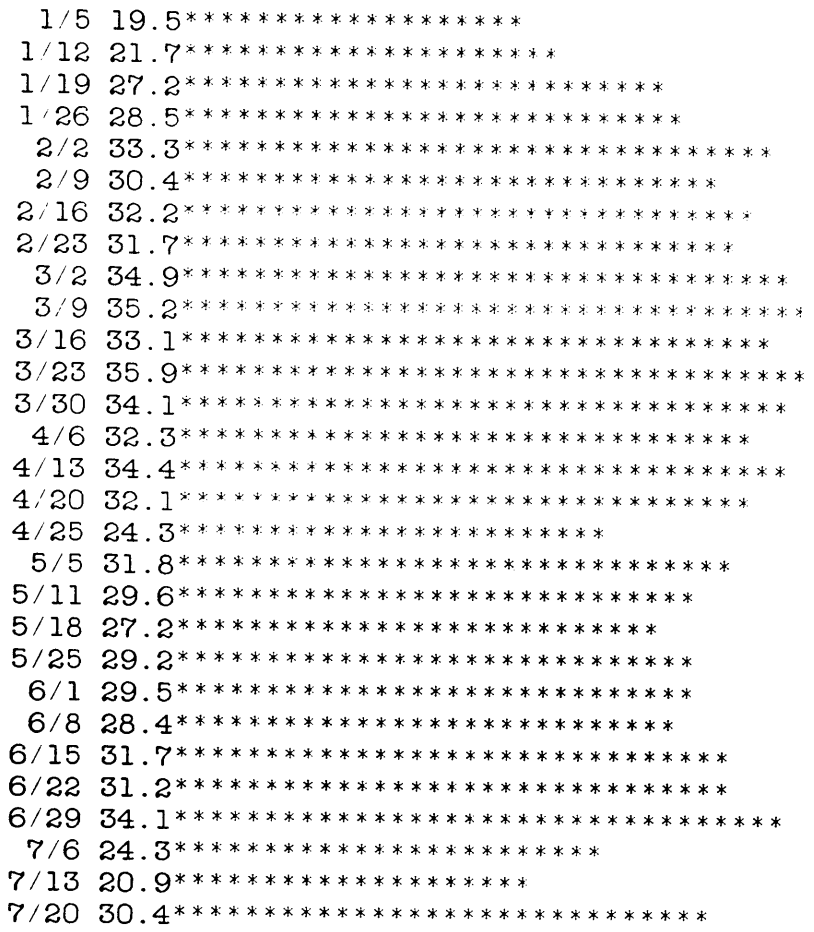
DROP RATES

Mailbridge	APR-86	MAY-86
MILARPA	4.3	2.6
MILBBN	7.9	5.0
MILDCEC	3.0	4.0
MILISI	4.1	4.1
MILLBL	4.3	4.3
MILSAC	2.3	2.7
MILSRI	3.1	2.2

Mailbridge	JUNE-86	JUL-86*
MILARPA	0.8	0.8
MILBBN	3.2	2.8
MILDCEC	2.0	2.2
MILISI	3.9	3.1
MILLBL	2.0	2.3
MILSAC	1.9	1.7
MILSRI	1.5	1.4

*1st three weeks

MAILBRIDGE WEEKLY THROUGHPUT. 1986
(million datagrams)



ARPANET BUSY HOSTS

From Fri Jul 11 1986 00:00:16
To Fri Jul 18 1986 00:00:16

	host	address	%intr	total	%net
1.	PURDUE-CS-GW	{ 37/2}	4.4	5.59	4.2
2.	ISI-GATEWAY	{ 27/3}	1.0	5.58	4.2
3.	ISI-MILNET-GW	{ 22/2}	0.6	5.16	3.9
4.	WISC-GATEWAY	{ 94/0}	3.1	4.94	3.7
5.	BBN-MILNET-GW	{ 5/5}	9.1	4.56	3.4
6.	DCEC-MILNET-GW	{ 20/7}	8.2	4.36	3.3
7.	MIT-GW	{ 77/0}	3.2	3.52	2.6
8.	SAC-MILNET-GW	{ 80/2}	1.9	3.40	2.6
9.	BBN-TEST3-GWY	{ 63/3}	100.0	2.63	2.0
10.	UCB-VAX	{ 78/2}	2.6	2.59	2.0
11.	SEISMO	{ 25/0}	0.0	2.23	1.7
12.	BBN-NET-GW	{ 82/4}	12.4	2.18	1.6
13.	GW.RUTGERS.EDU	{ 89/1}	2.6	2.16	1.6
14.	MC.LCS.MIT.EDU	{ 44/3}	0.2	2.15	1.6
15.	BBN-INOC	{ 82/2}	12.2	2.12	1.6
16.	SRI-MILNET-GW	{ 51/4}	25.2	2.11	1.6
17.	YALE	{ 9/2}	8.1	2.05	1.5
18.	CSNET-RELAY	{ 5/4}	2.3	2.01	1.5
19.	DCEC-GATEWAY	{ 20/1}	9.6	1.97	1.5
20.	LBL-MILNET-GW	{ 68/0}	1.5	1.94	1.5
	host totals		9.0	63.34	47.6
	network totals		12.3	133.13	

CONUS-MILNET BUSY HOSTS

From Fri Jul 11 1986 00:00:13
To Fri Jul 18 1986 00:00:13

	host	address	%intra	total	%net
1.	ISI-ARPA-GW	{103/0}	28.8	3.67	4.2
2.	BBN-ARPANET-GW	{ 49/2}	0.7	3.32	3.8
3.	AERONET-GW	{ 65/8}	0.7	3.30	3.8
4.	DCEC-ARPA-GW	{104/0}	1.0	3.14	3.6
5.	MIL-80-SHER1	{ 55/3}	92.1	3.07	3.5
6.	BRL	{ 29/0}	3.3	2.58	3.0
7.	SAC-ARPA-GW	{105/0}	0.9	2.47	2.8
8.	DECWRL-GW	{ 16/7}	1.8	2.46	2.8
9.	BBN-MINET-A-GW	{ 40/1}	1.1	2.10	2.4
10.	MIL-80X-SHER56	{ 55/4}	100.0	1.89	2.2
11.	SIMTEL20	{ 74/0}	2.0	1.74	2.0
12.	AMES	{ 16/0}	3.8	1.42	1.6
13.	NARDAC-NOLA	{109/4}	0.0	1.39	1.6
14.	SRI-NIC	{ 73/0}	2.3	1.34	1.5
15.	LBL-ARPA-GW	{ 34/3}	6.9	1.30	1.5
16.	STL-HOST1	{ 61/0}	32.7	1.18	1.4
17.	YUMA-GW	{ 75/3}	1.2	1.08	1.2
18.	ARPA-GW	{106/0}	4.8	1.07	1.2
19.	LLL-CRG	{ 21/3}	3.1	1.06	1.2
20.	DTRC	{ 81/3}	13.6	1.03	1.2
	host totals		16.9	40.73	46.6
	network totals		18.5	87.33	

SOURCE QUENCHES, A Case Study: June 19

Number of entries in the day's file
4522

ISI: Number of entries = 752

EST				
00:00	23	12:00	56
01:00	7	13:00	71
02:00	16	14:00	31
03:00	0	15:00	5
04:00	4	16:00	67
05:00	2	17:00	35
06:00	1	18:00	78
07:00	0	19:00	44
08:00	3	20:00	136
09:00	6	21:00	133
10:00	24	22:00	57
11:00	96	23:00	50

MILISI: Number of entries = 463

EST			
00:00	3	12:00	13
01:00	0	13:00	47
02:00	0	14:00	10
03:00	0	15:00	16
04:00	5	16:00	62
05:00	0	17:00	40
06:00	0	18:00	74
07:00	0	19:00	86
08:00	0	20:00	64
09:00	12	21:00	31
10:00	17	22:00	23
11:00	63	23:00	10

MILDCEC: Number of entries = 891

entries from ns1.cc.ucl.ac.uk = 769

Much as above but with intense periods:

5) Internet Measurement Criteria - Lixia Zhang, MIT

Presentation Slides

Internet Traffic Measurement

Two constraints in IP congestion control f

- control individual hosts directly
- a feedback control system

Why need a traffic measurement:

- verify control feasibility

  control response $\ll$ target chang

- Determine control parameters

  traffic average interval

  control stability / expiry time length

What to measure:

pick up a few heavily loaded gateways.

1. The distribution of host-to-host transfer durations

(control feasibility)

2. The number of source hosts when a congestion occurs at a gateway

(control overhead
control effectiveness)

3. The distribution of internet transmission delay

(control feasibility)

4. Gateway load diagram

How fast does the load change?

(average interval
control ⇓ feasibility)

What to measure continued:

5. Internet dynamics

How often routes change

gateway crash

How many packets dropped due to congesti

due to trans. erro

needed for control parameter

6. Network throughput: ARPANET

7.

An Alternative:

instead of controlling hosts directly control traffic between gateways.

Advantage:

increased control feasibility

Stabler traffic (by L < N)

shorter control response time

Disadvantage

constraint on the routing

fairness issue

mutual dependencies between gatew

a higher cost in gateway algorithm

(may not be

EGP Miscellanies

- EGP is a reachability exchange protocol

EGP is not a routing protocol.

EGP will not become a routing protocol

By definition of autonomous system we no longer have a global routing protocol

- Autonomous systems are connected by networks.

Gateways belong to AS's

Networks (should :) belong to AS

- An internet packet is routed piecewise by the IGP's as it travels through individual AS's.

Each AS sets a boundary on routing control.

How can we prevent routing loops in an internet which does not have a global routing control?

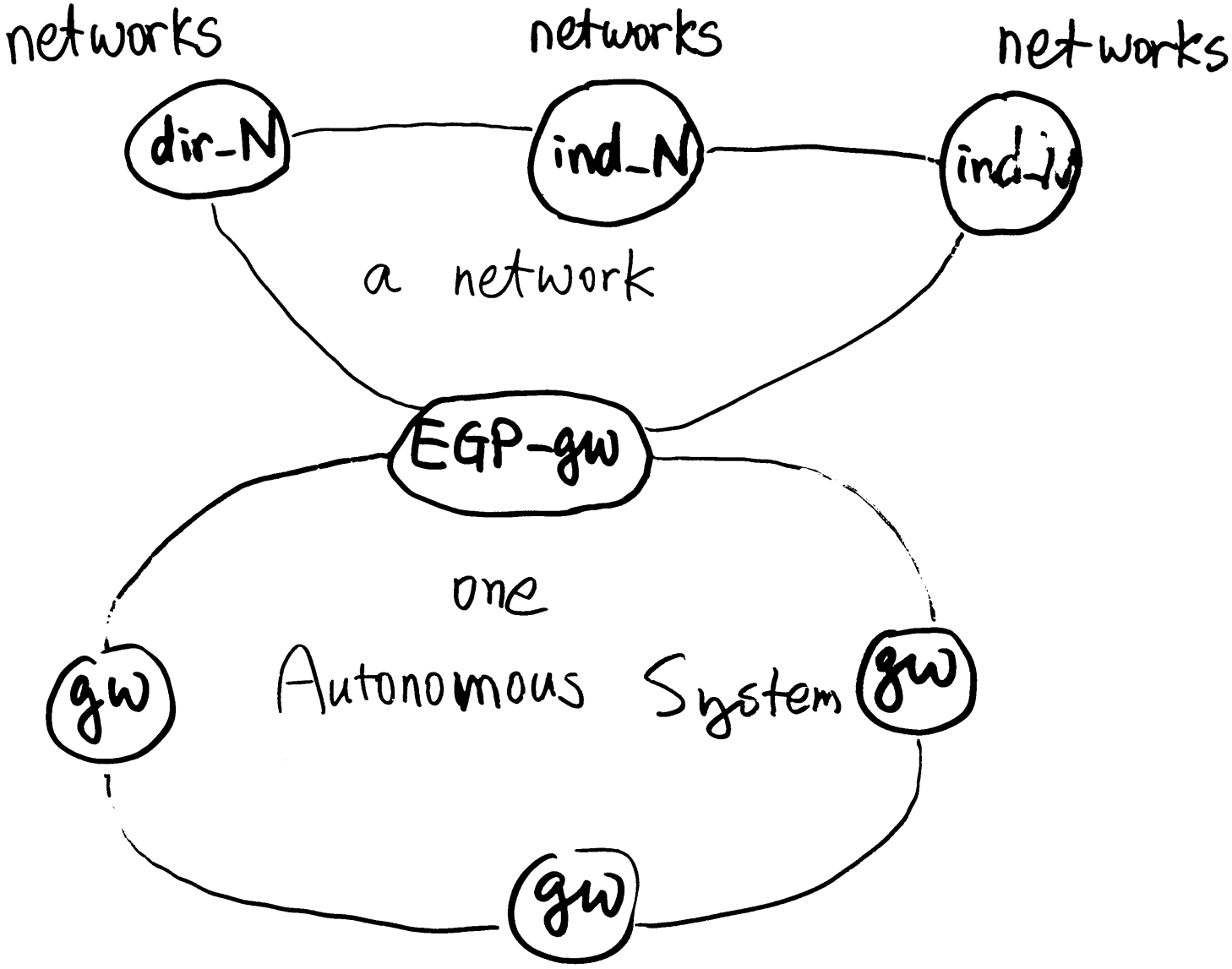
3 ways to attack the problem

1. set restriction on topology

2. require IGPs to have a magic way to avoid loops

3 restrict the contents of EGP NR messages

Conclusion:



IGP routing database, like GGP,

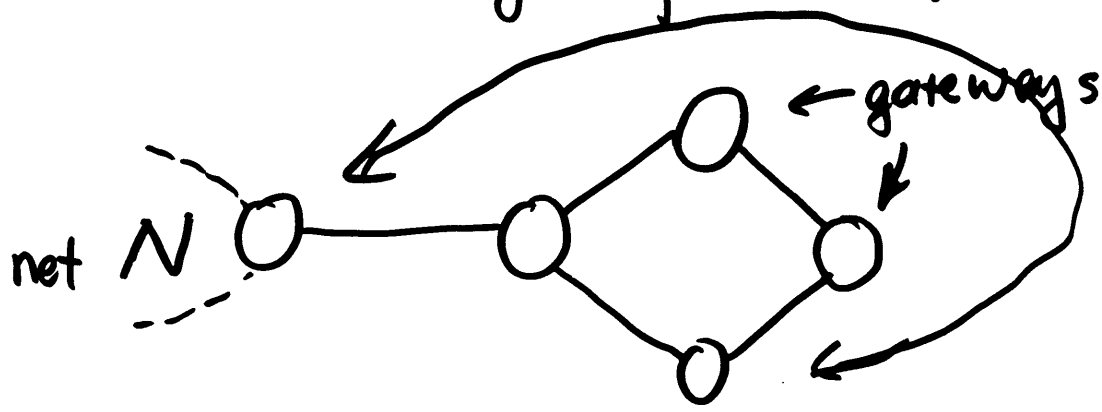
- has no **global** topology map
- pass information **between** neighbors only

- GGP routing information passing problem:

Good news goes fast;

bad news stays forever.

why?



- One solution: "Hold-down"

- "Hold-down"'s problem:

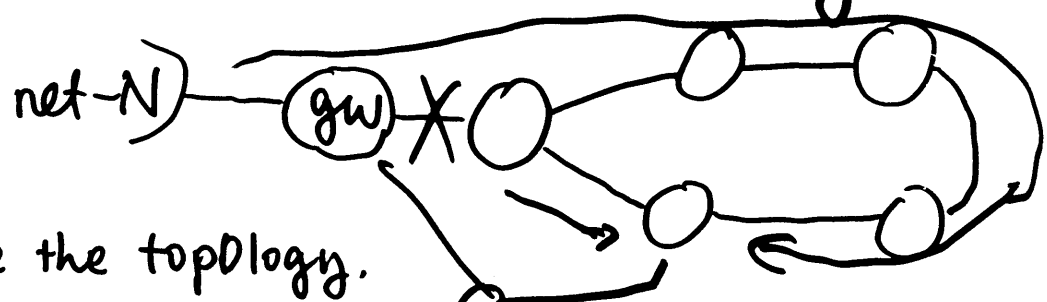
How long to hold-down?

One algorithm to make IGPs route loop-free with an arbitrary topolog

Basic ideas

1. A loop is created when the reachability of a net exists, but no route exists.

2. To avoid false reachability or looping
   - propagate info along each existing pat
   - Never tell net reachable info back to a gate who knows it already

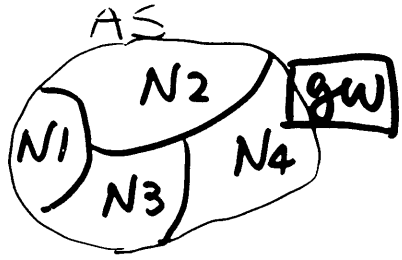


so despite the topology.
net-N's reachability is propagated in a tree.

3. When a path breaks at any point,
   - No further net-reachable info propagated from source
   - However the routing database has a memor

The algorithm: (working?)

1. Mark all reachability info with the originating gateway



2. Never propagate net-reachable info back to a gw who knows it

3. If the originating gateway says net-N unreachabl ("net-N reachable") propagate this unreachability in all directions through the internet.

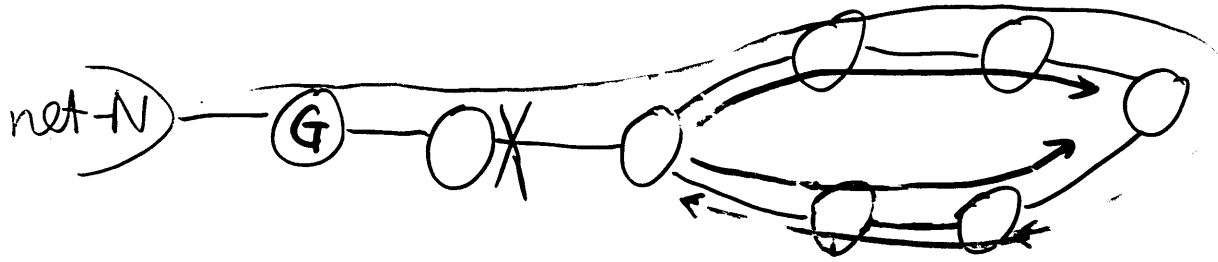
4. When a neighbor gateway, G, is detected down, propagate the net-unreachable info for all nets marked with the gateways seen in net reachability from G, including G.

(A path is broken, flush all routing databases

Comparison with hold-down:

Hold-down propagates bad news along th previous best path.

We propagate bad news in all directior

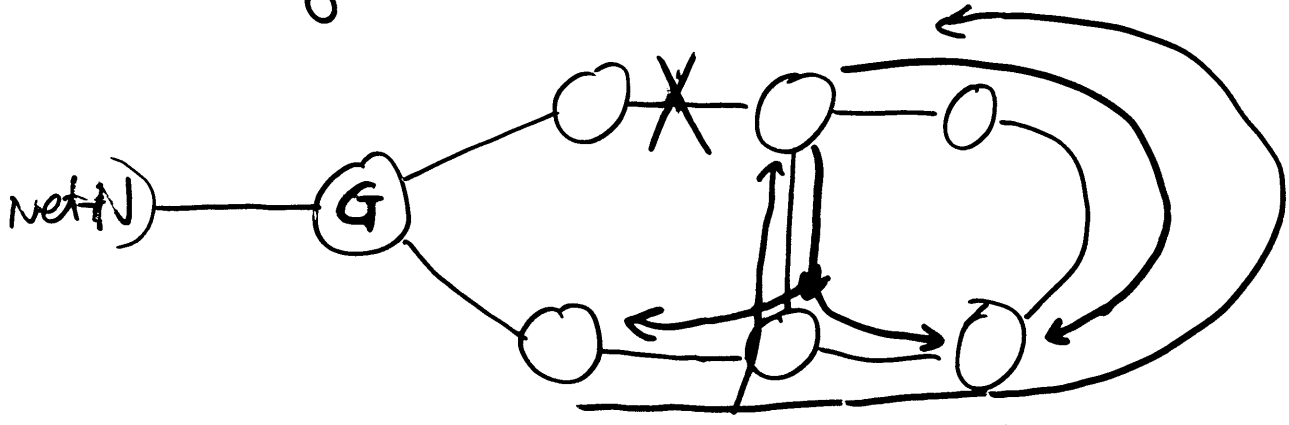


reachability path
flush paths

Advantage:

avoid estimating hold-down time

Disadvantage:



1. flush
2. re-establish reachabil

6) EGP Enhancements and Changes -

Mike StJohns, DDN

Presentation Slides

Internet Engineering Task Force

EGP Strikes Out Again

IETF 23-24 July

"Just a little software away!"

- Ground Rules
  - -- Not Disruptive
  - -- Within Scope of Current Standard
  - -- SHORT TERM FIX!!

Internet Engineering Task Force

EGP Strikes Out Again

IETF 23-24 July

"Just a little software away!"

- Problems
  - -- Update Size
  - -- Topology
  - -- Propagation Time of Updates

Internet Engineering Task Force

EGP Strikes Out Again

IETF 23-24 July

"Just a little software away!"

- Solutions
  - -- "Refine" Current Standard
  - -- Extend Current Standard

Internet Engineering Task Force

EGP Strikes Out Again

IETF 23-24 July

"Just a little software away!"

"Refine" Current Standard - Multiple Updates per Poll

Pro: Easy to do programming wise.

Con: Disruptive, 2 changes necessary.

Limited Gain

Slows propagation of reachability info

Internet Engineering Task Force

EGP Strikes Out Again

IETF 23-24 July

"Just a little software away!"

Extend Current Standard

- New "POLL" Message
- Simplex Transport Protocol Based on NETBLT
- Addition of AS # to Update Message
- Addition of Gateway Status Update Message

Internet Engineering Task Force

EGP Strikes Out Again

IETF 23-24 July

"Just a little software away!"

New "POLL" Message

- Subtype requests either standard or Gateway Update
- Specifies parameters for transport mechanism
- New message type in Version 2 EGP so will be ignored by non-compliant EGPs
- Can ask for all info, or info newer than a certain event

Internet Engineering Task Force

EGP Strikes Out Again

IETF 23-24 July

"Just a little software away!"

Simplex Transport Protocol Based on NETBLT

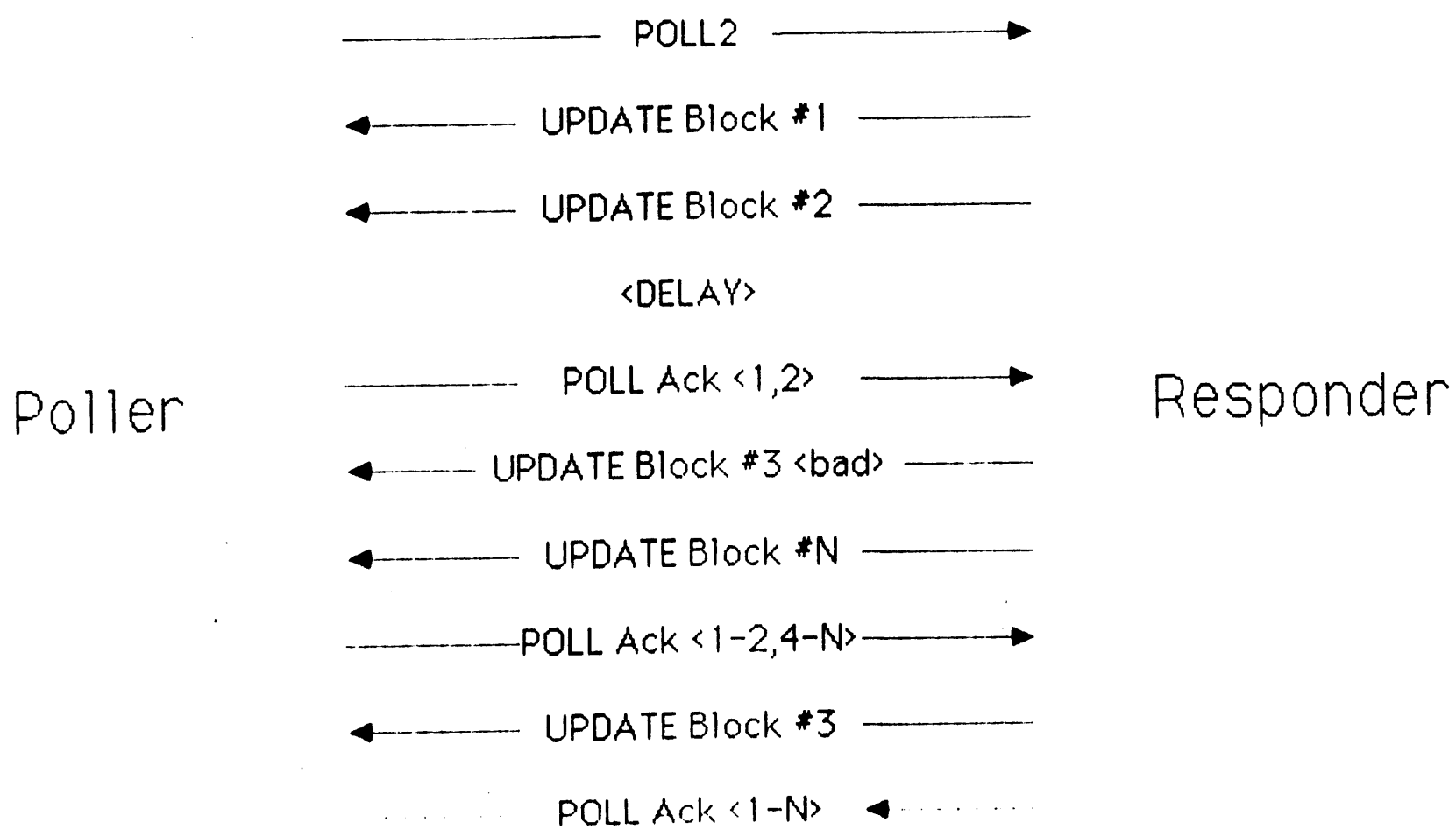
- Polling side identifies maximum block size as well as maximum send rate.
- Updating side sends updates in pieces
- Polling side sends acks (bit mask of good blocks) every X1 interval (long timer) or every X2 interval (short timer) if no block is received.

Internet Engineering Task Force

EGP Strikes Out Again

IETF 23-24 July

"Just a little software awag!"



Internet Engineering Task Force

EGP Strikes Out Again

IETF 23-24 July

"Just a little software away!"

Simplex Transport Protocol Based on NETBLT

Pro: No setup negotiation

Small amount of state information

Con: Memory expensive - buffer maintained until

complete update received

Internet Engineering Task Force

EGP Strikes Out Again

IETF 23-24 July

"Just a little software away!"

Addition of Autonomous System # to Update

- Seems to be needed, no great reason not to
- Should allow SOME topology restrictions to be eased

  even if all EGPs don't comply
- Further study necessary!

Internet Engineering Task Force

EGP Strikes Out Again

IETF 23-24 July

"Just a little software away!"

Addition of Gateway Status Update Message

- Simply reports UP/DOWN status of gateways
- Update smaller than normal update
- Can be used to recalculate reachability info

Internet Engineering Task Force

EGP Strikes Out Again

IETF 23-24 July

"Just a little software away!"

GOALS:

1. Draft RFC
2. IETF Review & Approval
3. Issue RFC
4. Implemented in Core Buttergates

Internet Engineering Task Force

EGP Strikes Out Again

IETF 23-24 July

"Just a little software awag!"

- Ground Rules
- Problems
- Solutions?
- Goals

7) Name Domains - Paul Mockapetris, ISI

The Domain Name System

Paul V. Mockapetris

USC Information Sciences Institute
Marina del Rey, California

Needs

Size:	new hosts, users, etc.
	network/internet connections
Distribution:	data and responsibility
Flexibility:	data types
	data representations
	host capabilities

Domain Axioms

Data is distributed.

Distribution is transparent to user.

Distribution control is distributed.

Data types are extensible.

Organization is hierarchical.

Hierarchy is extensible.

Domain Restrictions (temporary?)

System administrators are responsible for configuring system so that it works efficiently.

Updates are distributed by a refreshing discipline rather than an atomic update mechanism.

Abstract Database: Name Space

Tree with labels on nodes

Labels (except " ") are not unique

Name is path to root

Tree structure roughly corresponds to organizational structure

No distinction between nodes and leaves

Abstract Database: Resources

resource records (RRs) attached to nodes

zero or more RRs per node

RR = type

class

time-to-live (TTL)

resource data (RDATA)

RDATA varies with type and class

Abstract Database: Conventions

Case preservation

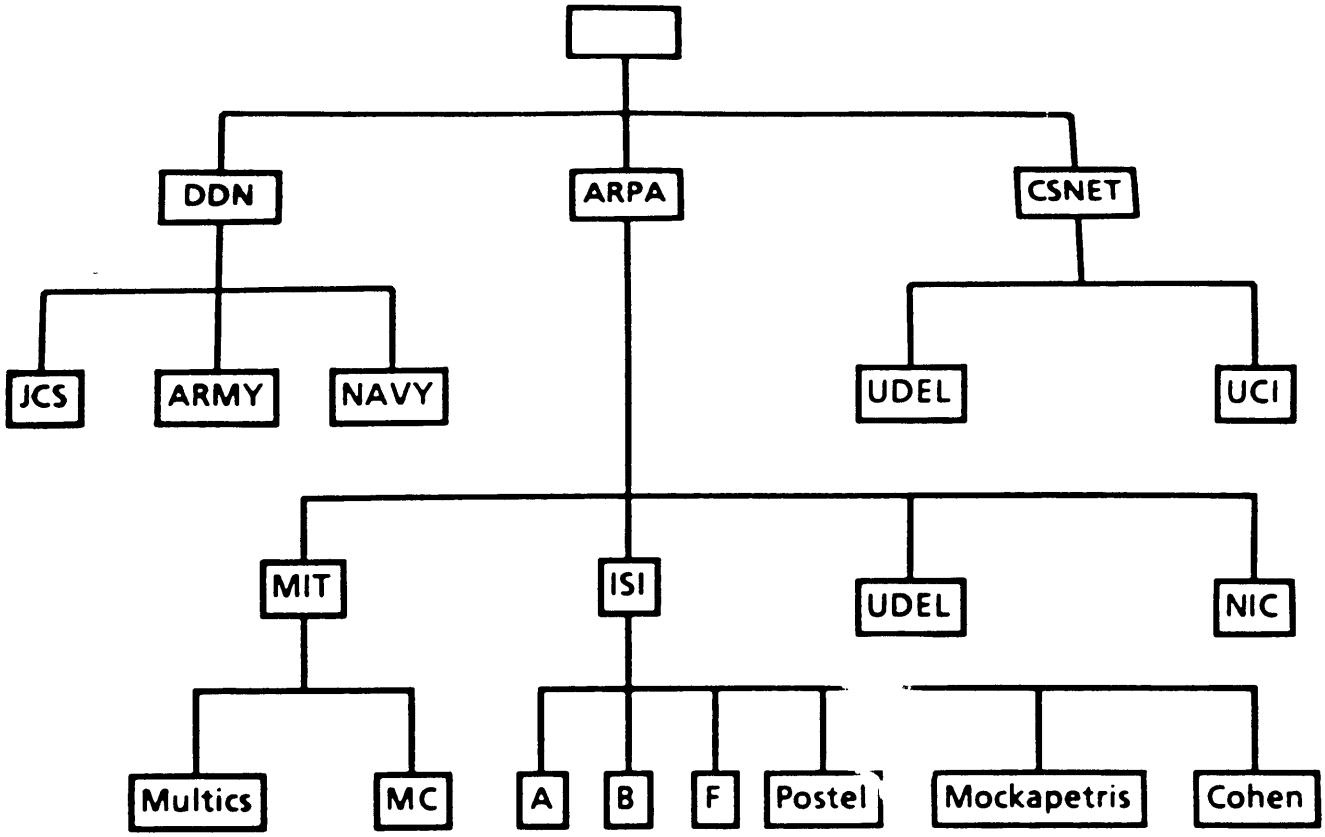
Case insensitive comparison

Multiple name printing rules

Mockapetris.ISI.ARPA => Mockapetris@ISI.ARPA

Mockapetris.ISI.ARPA => Mockapetris.ISI.ARPA.

Example Tree



Example RRs

Owner	Type	Class	RDATA
A.ISI.ARPA	A	IN	10.1.0.32
B.ISI.ARPA	A	IN	10.3.0.52
F.ISI.ARPA	A	IN	10.2.0.52
Postel.ISI.ARPA	MB	IN	F.ISI.ARPA
Mockapetris.ISI.ARPA	MB	IN	F.ISI.ARPA
Cohen.ISI.ARPA	MB	IN	B.ISI.ARPA

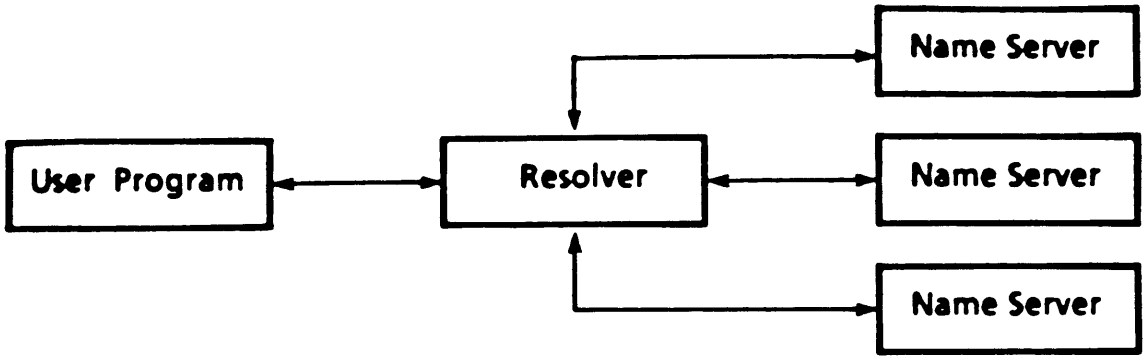
Queries

Simple:
Name,Qtype,Class => RRs

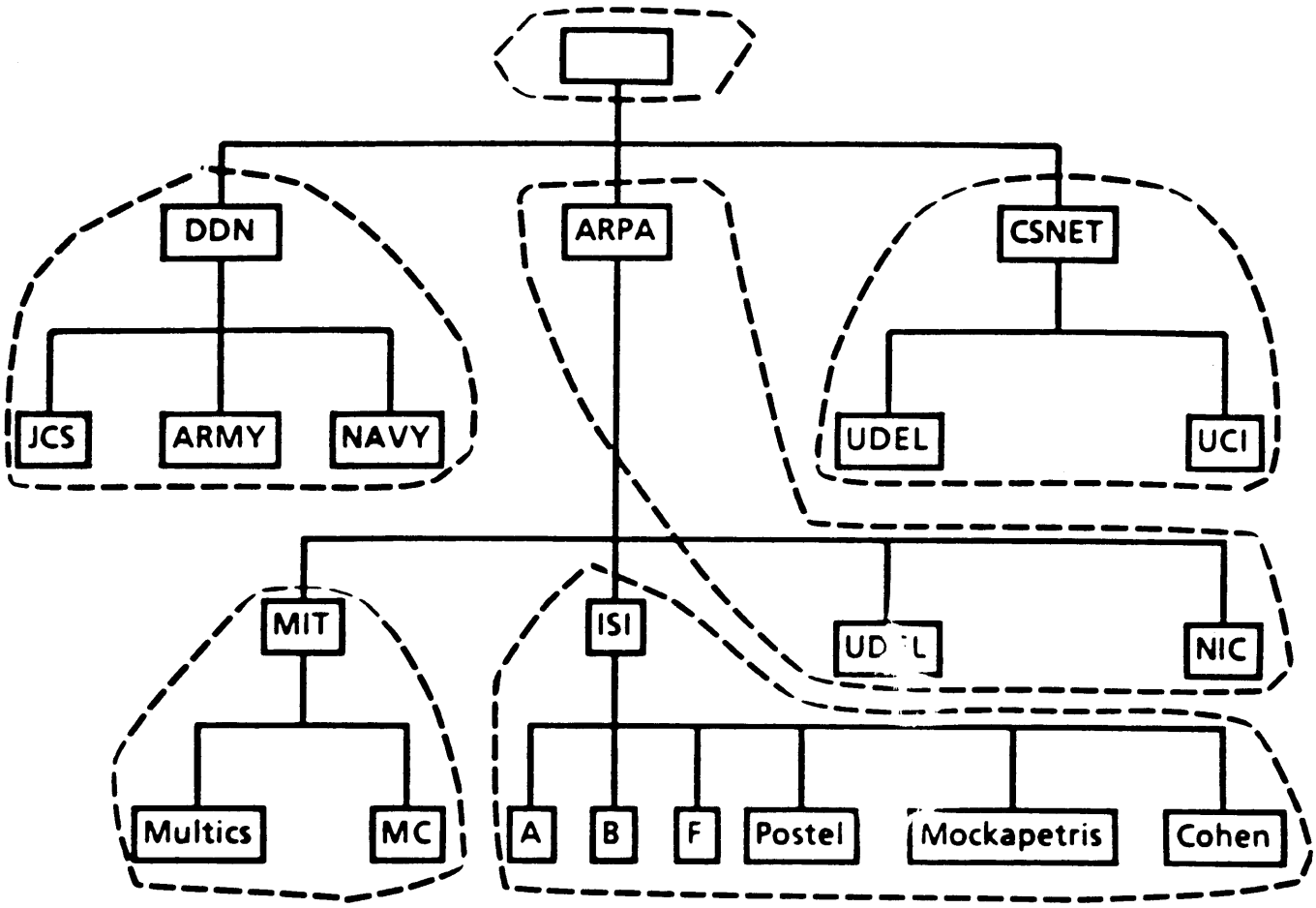
Completion:
Partial Name, Type, Class, Target Name => RRs

Inverse:
RRs => simple query

Distribution of Database: Agents



Distribution of Database: Zone Structure



Distribution of Database: Zone Contents

Authoritative data for zone contents

Zone marks (part of authoritative data)

Zone glue

Example zone

Owner	Type	Class	RDATA
ARPA	SOA	IN	NIC.ARPA
ARPA	NS	IN	NIC.ARPA
ARPA	NS	IN	F.ISI.ARPA
ISI.ARPA	NS	IN	B.ISI.ARPA
ISI.ARPA	NS	IN	A.ISI.ARPA
MIT.ARPA	NS	IN	XX.MIT.ARPA
A.ISI.ARPA	A	IN	10.1.0.32
B.ISI.ARPA	A	IN	10.3.0.52

Resolver logic

style is iterative

```
no-answer-yet: = true;
while no-answer-yet
do      begin
        send-query-to-best-NS;
        if authoritative-answer
        then no-answer-yet: = false
        else    add-knowledge
        end
```

Message Format

Header	ID, opcode, size, return code, etc.
Question	QNAME, QTYPE, QCLASS
Answer	RRs
Authority	RRs
Additional	RRs

Database Maintenance

Refreshing at intervals specified by zone master copy

Caching anywhere so long as rules are obeyed:

Cache one, cache all

timeout

careful with special QTYPEs

Status and Future

Connecting internets

Update management

The Domain Name System

What is it?

Distributed database

Protocol Specification

Name Servers

Name Resolvers

The Domain Name System

What is the client's view?

Tree structured name space with labels on nodes

Labels need not be unique

Normal tree structure is organizational, not artificial

Defined abstract data types

Each node has set of data elements

The Domain Name System

Sample RR data

```
ISI.EDU.          3600 IN SOA  VENERA.ISI.EDU. ...
                          NS   VAXA.ISI.EDU.
                          NS   VENERA.ISI.EDU.
                          MX   10 VENERA.ISI.EDU.
                          MX   20 VAXA.ISI.EDU.

VAXA.ISI.EDU.     3600 IN A    128.9.0.33
                          A    10.2.0.27
                          A    10.1.33.27
                          WKS  128.9.0.33  TCP ...
                          WKS  10.2.0.27 TCP ...
```

The Domain Name System

What is the administrator's view?

Name space is extensible

Control for subtrees can be delegated at any node

Assign time-to-live (TTL) for each data element, or use default to control external caching

Local policies can be "tuned" within global rules (e.g. replication, timeliness vs. overhead)

Automatic redundant copies, search guidance

The Domain Name System

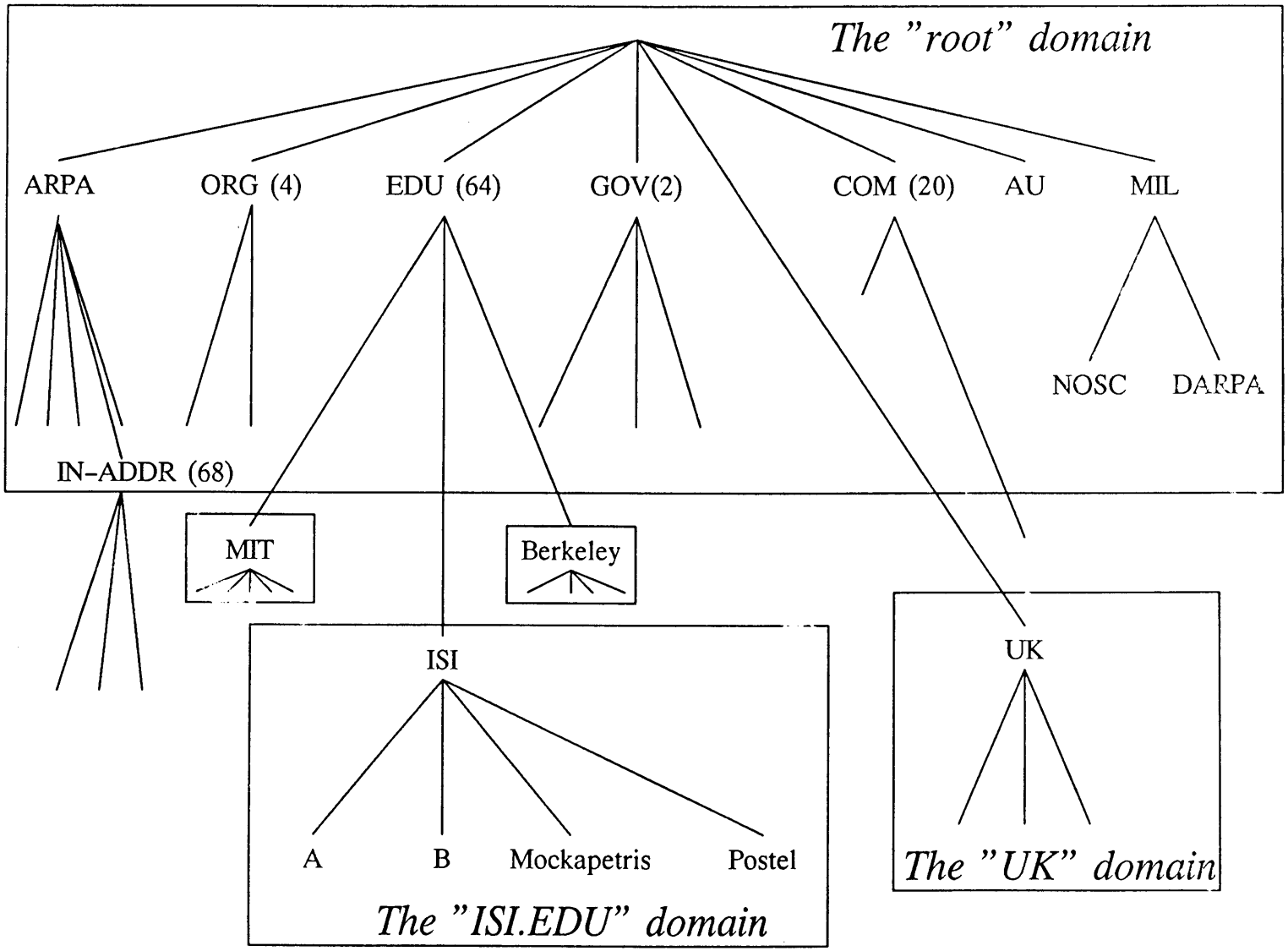
What are the internal mechanisms?

Redundant nameservers answer queries for local data or direct search to other name servers

Resolvers direct queries to local servers and follow referrals as required

Resolvers cache results, but delete expired data from cache

The Domain Name System



The Domain Name System

Current Status

The root domain is served by 4 redundant servers, provides name service to over 150–500 client hosts, some as only mechanism, others on experimental basis

Over 50 domains are delegated

The Domain Name System

Applications

Replace centralized host table information

Provide large databases (e.g. per user)

Provide for changing database (e.g. mobile host addresses)

Provide lower level for higher level systems (e.g. CCITT directory services for MHS)

Domain Problems

1. new root on 26
2. new format
3. diode gateways
4. domain police
5. MAIL (MX) conversion
6. TRANSMISSION SELECTION & TIMEOUTS

Domain Research (aka big problems)

1. UPDATE / AUTHENTICATION / SECURITY (NEED MODEL)
2. Negative response caching
3. Partial names
4. ISO/CCITT/IFIP Name Service
5. RR DESIGN

8) Internet Capacity Planning- Bob Hinden, BBN

Presentation Slides

Internet Capacity Planning
or
How to Manage Growth

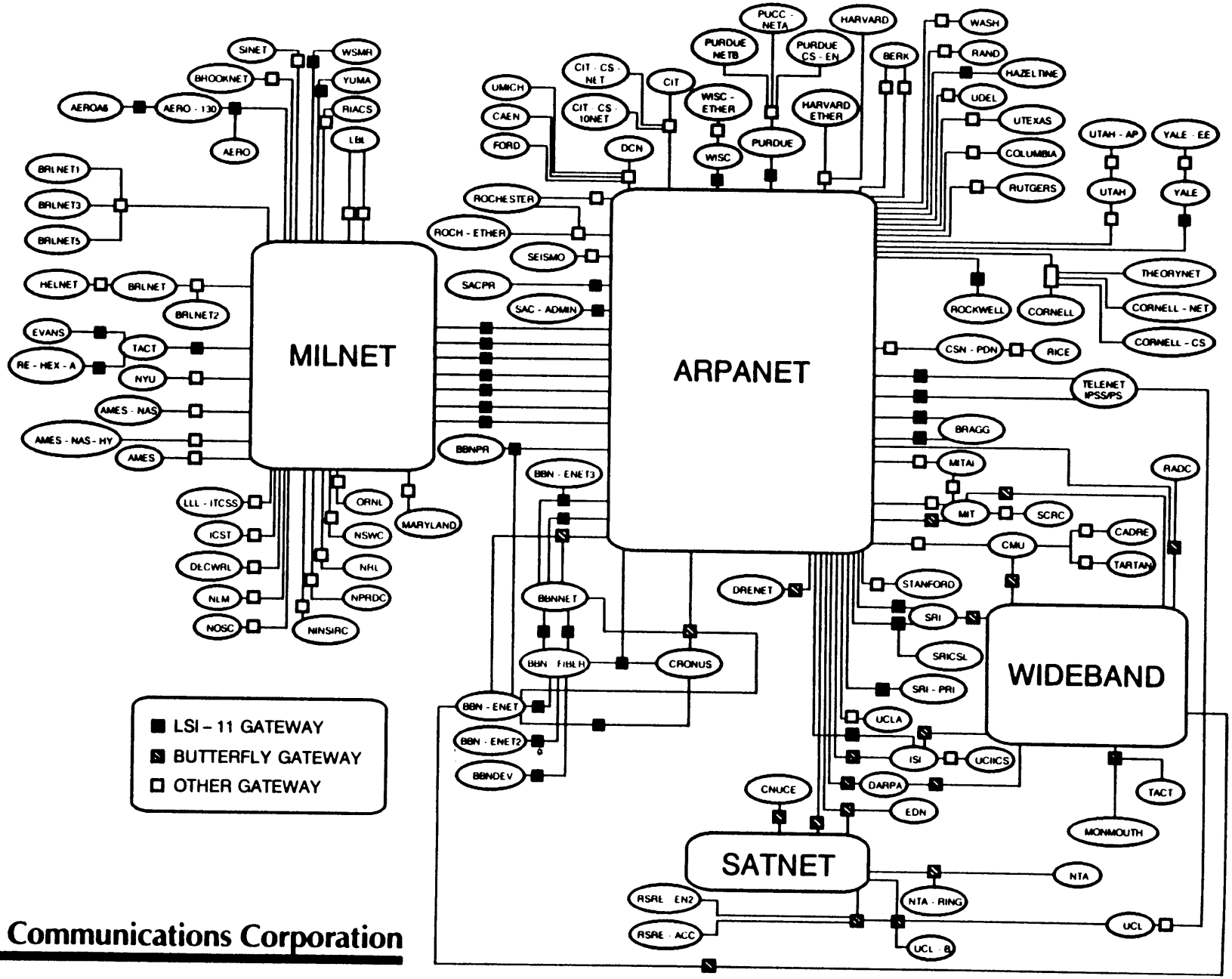
Robert M. Hinden

Manager Gateway Development

BBN Communications Corporation

WHERE WE ARE NOW
July 1986

- ~135 Operational Networks
- ~ 470 Assigned Networks
- 180+ Million Packets / Weeks in LSI-11 Core



BBN Communications Corporation

JULY 1986

CORE GATEWAY STATUS

- LSI-11 Gateway

150 Networks	Rel. 1008	
300 Networks	Rel. 1008.1	(Fall '86)

- Butterfly Gateway

1000 Networks	Rel. 3
(250 Internal, 750 External)	

- Mailbridge Gateways

Rel. 1008	Operational	LSI-11
Rel. 1008.1	Fall '86	LSI-11
Rel. 1009	Late '87	Butterfly

NON-CORE GATEWAYS

- Bridge
- Proteon
- Ford Multinet
- CMC
- CISCO
- Fuzzball

GROWTH PROJECTIONS

- Networks

150	Now
300	1988
1000	1990
10,000	1995

- Protocols

DOD IP	Now
ISO IP	1988
Flow Based IP	1990

- Type of Service Routing

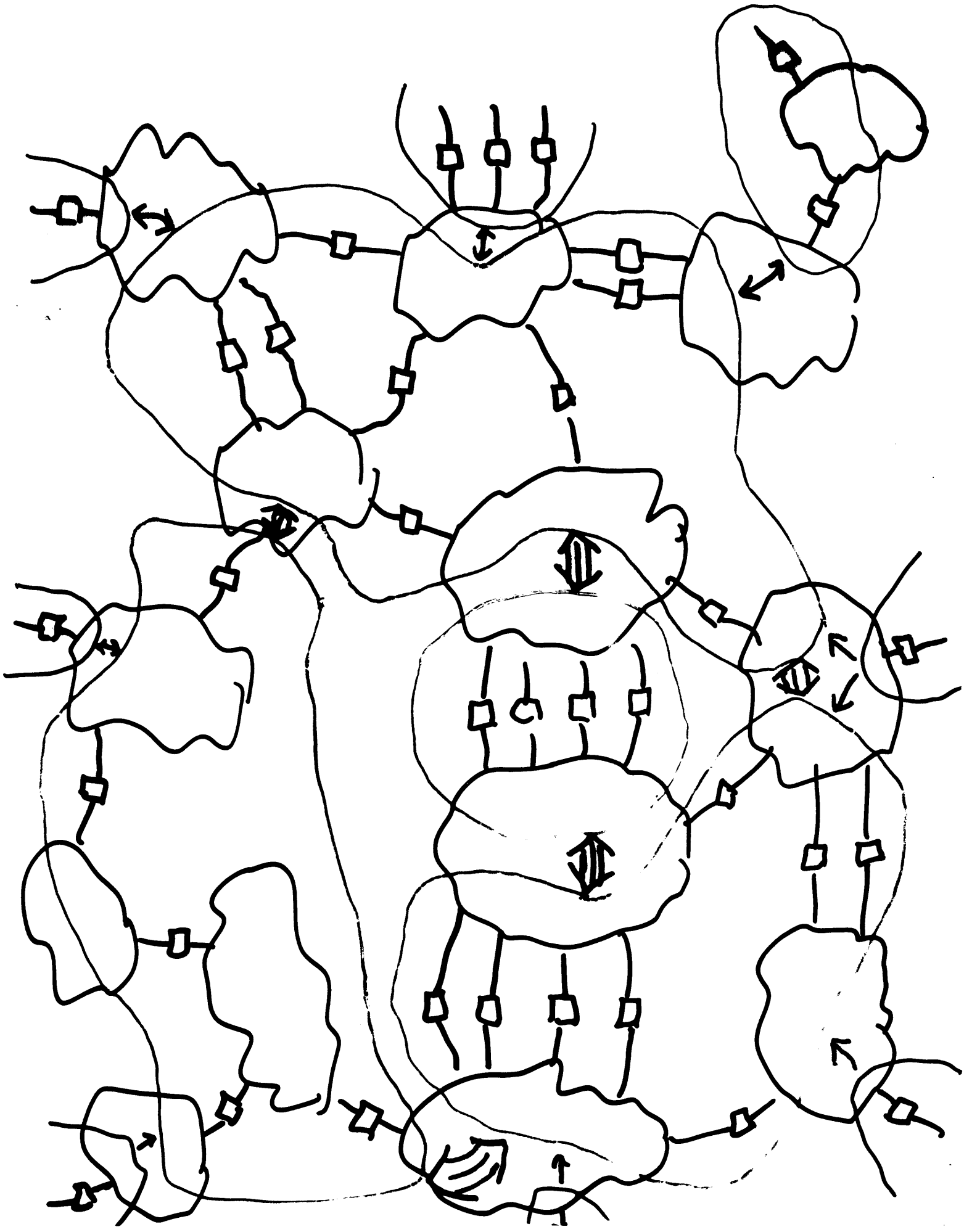
Min. Delay	Now
Throughput	1988

- Topology

Single Core w/ Stubs	Now
Linear Multi-System Core	1987
Mesh of Large A.S.	1989

Internet Routing

- Inter A.S. Routing - IGP
  - GGP
  - SPF
  - XNS
- Stub - Core Routing
  - EGP
- Core - Core Routing
  - "New EGP"



Core - Core Routing

- "Real" Routing Protocol
- Mesh of Autonomous Systems
- Type of Service,
- Addressing for multiple IP Protocols
- 10,000 Networks
- multiple Vendor support

How to Build

- Very well Specified
  - functions, time, ...
- One Set of Original Code
  - Common Language - "C" ?
  - Machine Independent Implementation
- Includes Remote Maintance Fonctions
  - Not Optional
- Certified
- Self Testing

EGP EVOLUTION

1) BIGGER UPDATES
   INCREMENTAL UPDATES

2) TOS
   WILD CARDS
   MULTIPLE IP's

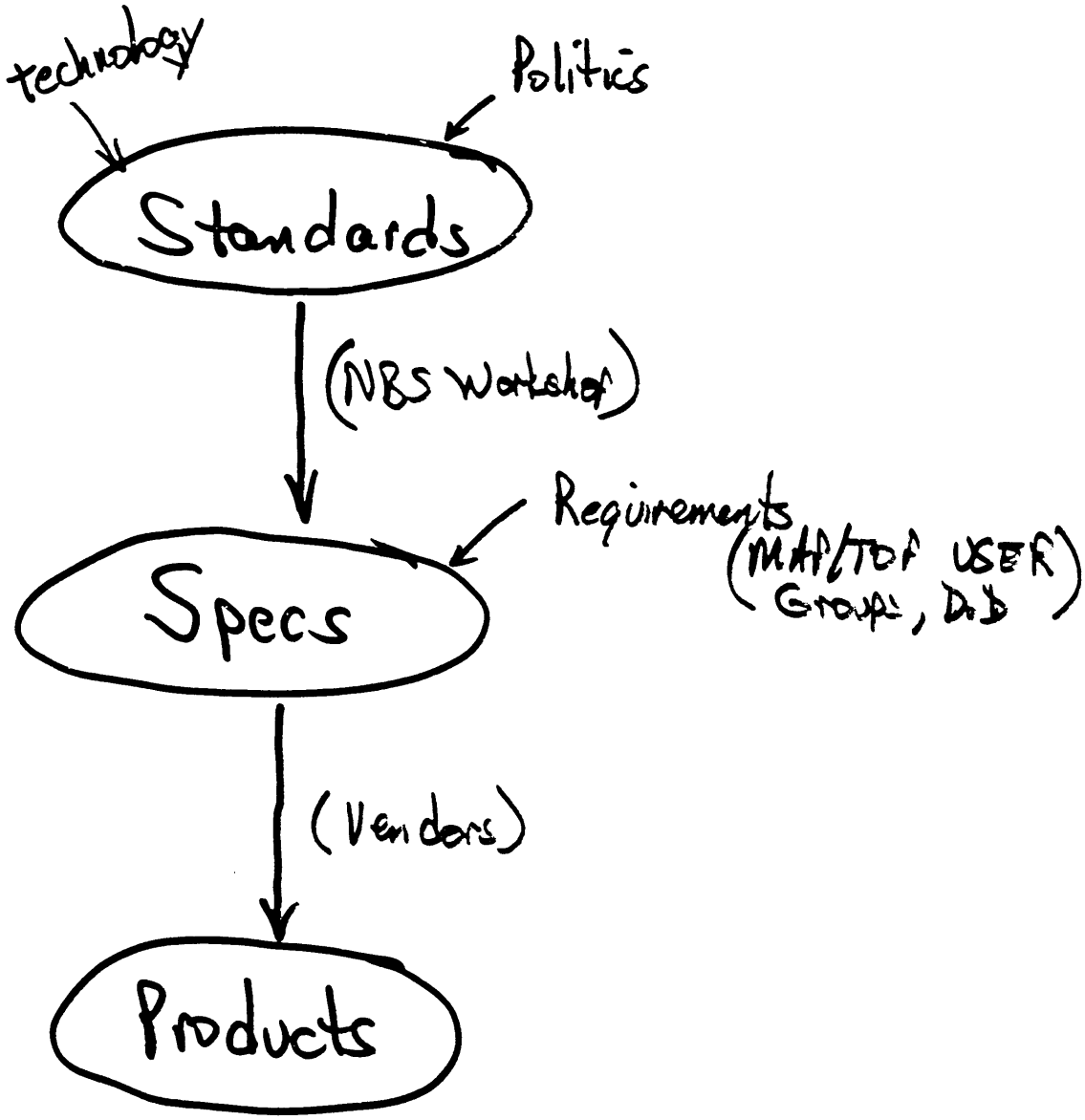
3) INCOMPLETE INFORMATION

9) ISO Transition Planning - Phill Gross, MITRE

Presentation Slides

Status of ISO + DoD/ISO Work

- Mitre Dual Protocol Host
- NBS
  - MAP/TOP/Impl. Workshop, OSInet
  - Routing Protocol Design (DCA)
  - DoD/ISO Appl. GWs (DCA)
- INARC
  - DoD Addr. in ISO IP (RFC986)
  - Framework for multiprotocol switching in Internet GWs
- ISO Marches On (Actually ANSI X353.3)
  - Routing Framework
  - Routing Arch.
  - IS-IS Protocol(s)


technology
Politics
Standards
(NBS Workshop)
Requirements
(MAP/TOP USER Group, DoD)
Specs
(Vendors)
Products

Status of ISO Standards

Complete

ISO 8473	IP
ISO 8473/DAD1	Underlying Service
ISO 8473/PDAD2	Formal Description (ESTL)
ISO 8348	Network Services Def.
ISO 8348/AD1	Connectionless NSD
ISO 8348/AD2	NSAP Addressing
ISO 8648	Internal Org. of Net. Layer
SC6-N4053	ES-IS Protocol (DP in 10/86)

Status (Cont)

In Progress

→ Routing Framework
- Exact Def of Routing
- Intersection of Routing Framework with Management Framework
  - When to use Net Layer vs. Appl. layer for routing
  - Motivation for ES-IS vs IS-IS split

→ Routing Architecture
- technical and administrative framework for Routing

→ IS-IS Protocol(s)
- The routing protocol itself (finally)

Events

- X3S3.3 meets for 1 week every 2 months

- NBS organized the "IS-IS Workshop" (April 29, 1986)

- Ad hoc Routing Arch. meetings

  - May 20-21, 1986
  - August 5-8, 1986

Proposed Architecture

Level 0 : ES-IS

Level 1 : Intra-Domain IS-IS

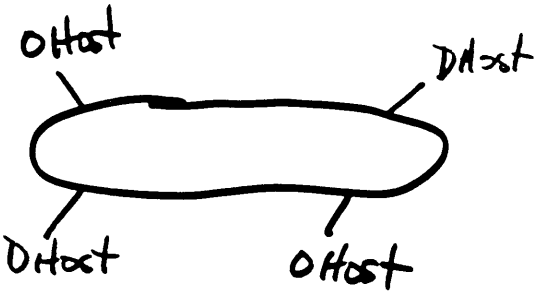
Level 2 : Inter-Domain IS-IS

Differences + Issues

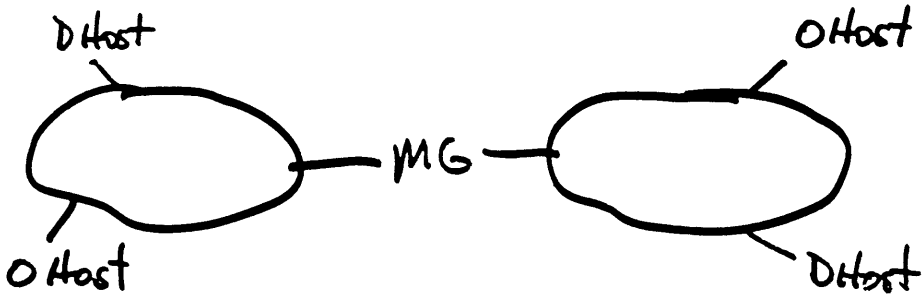
- Domains are Political + administrative
- Networks are part of Domains
- Have yet to address reachability vs. routing
- Not extensible past 3 levels

Target Capabilities (Crawl)

1)



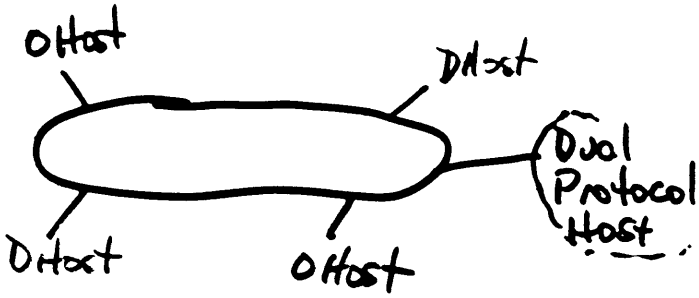
2)



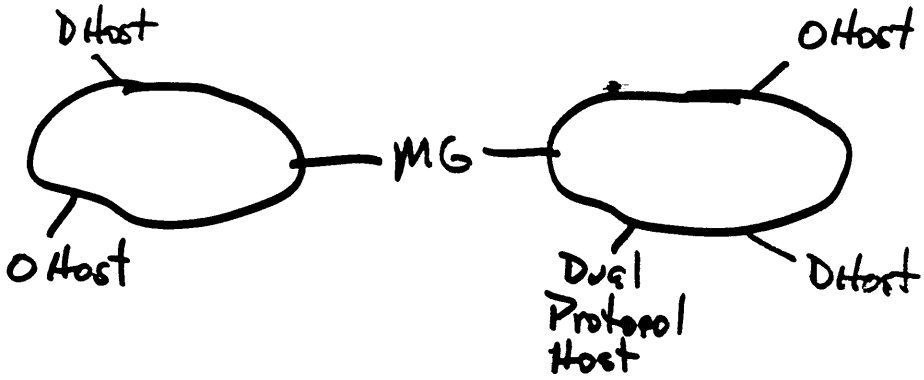
Closed Communities

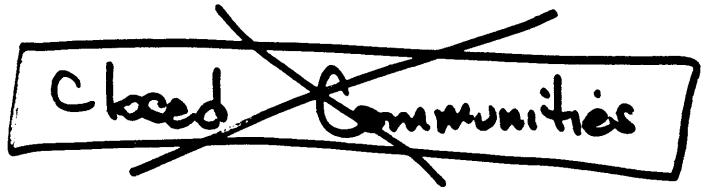
Target Capabilities (Crawl)

1)



2)

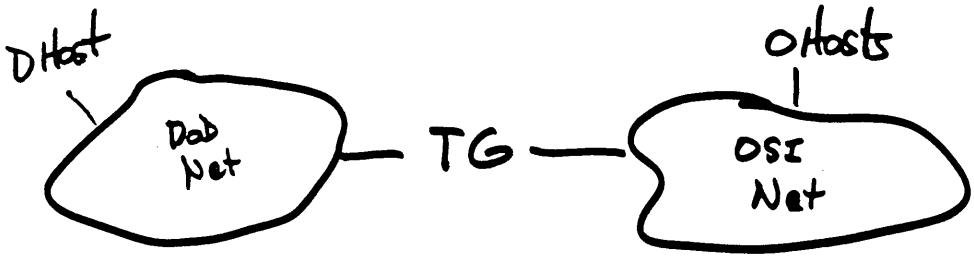




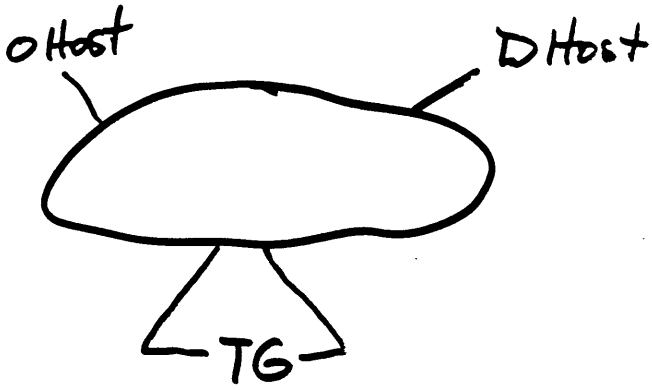
limited interoperability via
Staged FTPs & double login

Target Capabilities (Walk w/limp)

3)



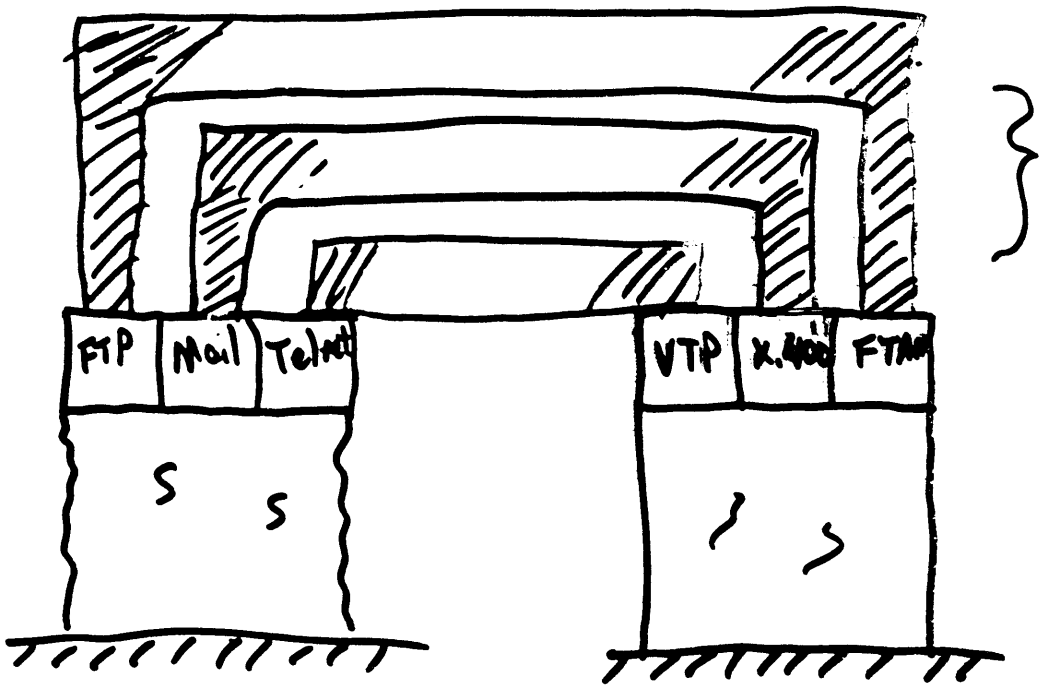
OR



NBS Appl. layer GW

Digression on TG

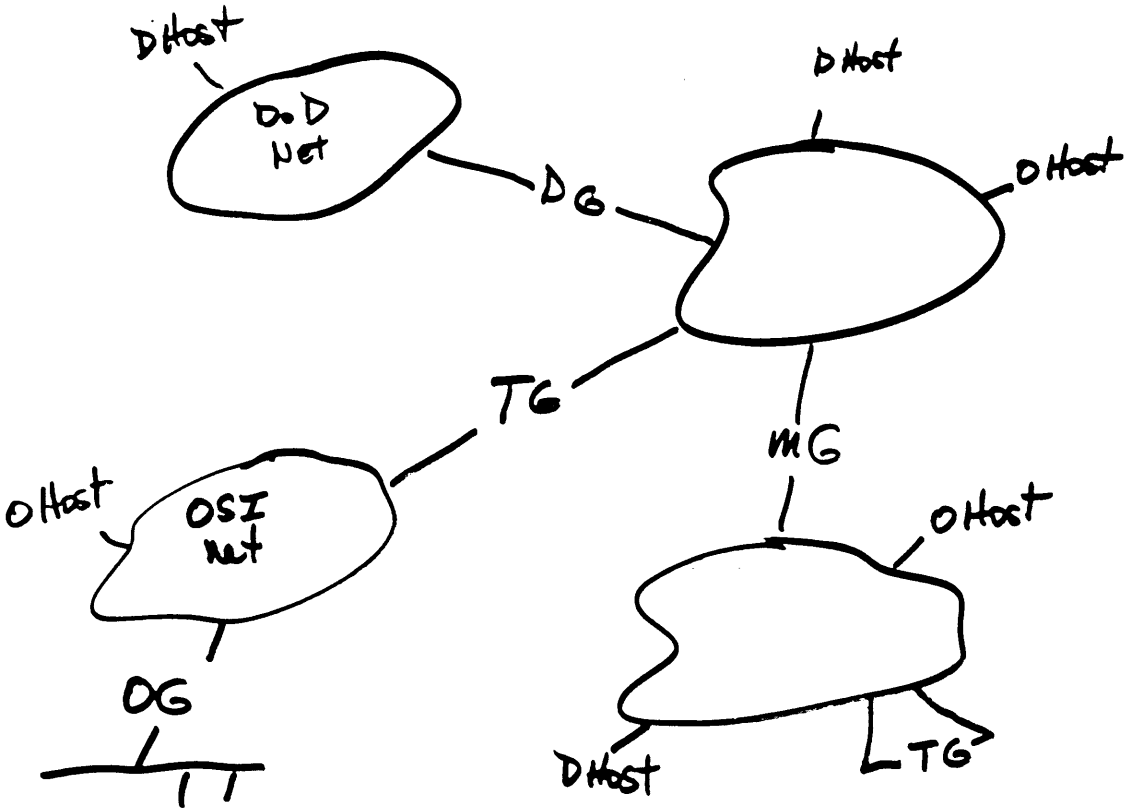
TG = NBS Application Layer Gateway



- Funded by DCA/DCEC.
- We need status report on this effort.

Target Capabilities (cut + run)

4) Teranet



Assumptions

- All accomodation will be on Port of DoD.
- No DHosts ~~on pure OSI nets~~ behind OGates are dual communities.
- No mixing of Protocol Stacks

Requirements

1) Hosts need to Demux IPs
(if not dual protocol, can simply dump unrecognized packets)

2) GWs need to demux and route IPs
(plus (1) above)

3) Need to solve addressing and routing issues or we'll be stuck with source routing like current mail gateways

eg smith%Psuvax.Bitnet@Wiscvm.Arpa

[Presumably, NBS is resolving questions like protocol translation vs. staging]

4) All this and Andy Rooney...

Appendix B - Additional Material

1) NSFnet Mail Archives, provided by David Mills (UDel)

2) Initial Delay Experiments with the University Satellite Network (USAN), Hans-Werner Braun (UMich)

Additional Material

1) NSFnet Mail Archives, provided by David Mills (UDel)

Additional Material

From @DCN6.ARPA:mills@dcn6.arpa Sun Jun 22 14:35:34 1986
Received: from DCN6.ARPA ([128.4.0.6]) by trantor.UMD.EDU (5.5d/umd.04)
        id AA29985; Sun, 22 Jun 86 14:34:58 EDT
Message-Id: <8606221834.AA29985@trantor.UMD.EDU>
Date: 22-Jun-86 18:25:15-UT
From: mills@dcn6.arpa
Subject: Letter from the swamps
To: jennings%pucc.bitnet@wiscvm.wisc.edu, dleonard%pucc.bitnet@wiscvm.wisc.edu,
        vanb@nrl-acoustics.arpa, estradas%sdsc.bitnet@wiscvm.wisc.edu,
        choy@ncar.csnet, krol%uiucvmd.bitnet@wiscvm.wisc.edu,
        levine@a.psy.cmu.edu, alison@devvax.tn.cornell.edu,
        swb@devvax.tn.cornell.edu, farber@huey.udel.edu, hwb@umich1.arpa,
        petry, steve@brl.arpa, craig@devvax.tn.cornell.edu,
        catlett%ncsavmsa.bitnet@wiscvm.wisc.edu, dd0t@te.cc.cmu.edu,
        mills@dcn6.arpa, louie, ntag@csnet-sh.arpa
Status: R

Folks,

Verily comes now a progress report on the NSF Backbone installation, along with notes of interest on the USAN integration, otherwise known as the Great ARP Wars. The source for much of the material herein is Hans-Werner Braun, who along with me spent over a week of battle lasting until 3 AM on several occasions.

Current Status

As you know, fuzzballs are in place now at several sites, but not all. Most of the lines have been checked out, but not all. The DMV serial interfaces received with the original shipment were the incorrect model, as you know, but sufficient numbers of the correct model have been borrowed to bring up three sites, SDSC, NCAR and Cornell. The UIUC fuzzball is also up, but accessable only via USAN, which will be explained later. Rumor is that CMU may also come up with borrowed interfaces. I do not know the status of the permanent replacements.

Hans-Werner and I spent a couple of days on the telephone with SDSC, NCAR and Cornell folks to identify and correct several snafus, including instances where previously distributed interface and modem configuration information was lost or ignored. Nevertheless, the three sites can now exchange traffic with each other and their attached Ethernets.

Since the Ethernets homing the Backbone gateways are not directly accessable from the Internet, real magic was necessary for monitoring and control during the debugging period. Hans-Werner cast a spell in the form of a fuzzball gateway between UMICHnet, which has a rickety path via DCN-GATEWAY to the world, and the USAN Vitalink/TransLAN Ethernet. It turns out that many of the Backbone Ethernets are also connected to USAN, so a backdoor path (with glorious loops and snarls) is available. Without this magic it would have been darn near impossible to detect and correct some exotic bugs that occured. However, the USAN configuration quite literally amounts to a party line to which is connected the Ethernets of some nine major institutions, so taming the gateway produced some surprising and important lessons on its own.

Several minor bugs were discovered in the fuzzware during the debugging period. The symptoms, diagnosis and resolution of these will be summarized in another message. New versions of the fuzzware should be distributed early next week to the active sites. Additional distributions can be expected as the

configuration stabilizes and the full set of system features is activated. Minor changes to refine monitoring and control procedures can also be expected as operational experience accumulates.

The Next Steps

At this point it is most important to identify and resolve issues on how client hosts at the various sites are to use the Backbone and whether and how the Backbone is to be connected to the Internet (ARPANET, et al). The issues quickly become incredibly complex when the Backbone is considered in context with all the other connectivity involving USAN, ARPANET and other nets. In addition, the debugging effort has revealed widespread discrepancies in etiquette practiced on the various client Ethernets, so some modification to their hosts may be required.

I believe the optimum strategic course right now is to avoid constraining the options. For instance, while CMU has been designated the primary interchange point between the Backbone and ARPANET, whatever changes are necessary there to support this interchange should be generalizable and apply to other sites at a later date. In addition, monitoring and control support programs built by Cornell should be rehostable (at least within the bsd Unix community), so that these functions can be made redundant. As experience and use accumulates, some access restrictions may become necessary, perhaps using existing fuzzball features or additional ones requiring only minor software changes.

The most important issue right now is to stabilize the interchange paths between the Backbone and aassociated nets, including USAN, the various Ethernets and the ARPANET. It is vital that this be done in a robust manner so as to insure reliable access for debugging. The USAN has proved extraordinarily useful when other things break, but is perhaps best limited to monitoring and control use. Therefore, priority attention should be given reliable ARPANET access. It turns out that such might be the only ARPANET access for some of the sites, at least for the time being. Note that access for certain sites is now being provided on a strictly temporary basis via USAN, UMICHnet and DCN-GATEWAY.

Perhaps the most useful technical contribution right now would be a module to be added to the Cornell gateway, later to the CMU gateway, that would allow routes to be exchanged between these gateways and the local Backbone fuzzball. While this could be done in several ways, such as extending EGP out the "back door," building in RIP support in the fuzzball, etc., the greatest functionality would be achieved if such a module participated in the fuzzball routing algorithm. This is not considered difficult and, indeed, a skeleton of such a module has already been built for our DCNet Sun workstation. The module would then interact with the EGP and RIP data bases as required. This is the same strategy practiced at DCN-GATEWAY, which happens to be a fuzzball (but does not support RIP).

A description of the fuzzball routing algorithm is in RFC-891 along with a formal definition of its state machinery; however, as might be expected, that description is somewhat out of date. A summary of the operation and details of the packet formats will be provided in another message. I would be glad to provide additional details as required to any stoutheart willing to build peer support for bsd Unix.

:Dave

From @DCN6.ARPA:mills@dcn6.arpa Sun Jun 22 15:39:41 1986
Received: from DCN6.ARPA ([128.4.0.6]) by trantor.UMD.EDU (5.5d/umd.04)
        id AA00241; Sun, 22 Jun 86 15:39:02 EDT
Message-Id: <8606221939.AA00241@trantor.UMD.EDU>
Date: 22-Jun-86 19:26:00-UT
From: mills@dcn6.arpa
Subject: Hello devils and other daemons
To: jennings%pucc.bitnet@wiscvm.wisc.edu, dleonard%pucc.bitnet@wiscvm.wisc.edu,
        vanb@nrl-acoustics.arpa, estradas%sdsc.bitnet@wiscvm.wisc.edu,
        choy@ncar.csnet, krol%uiucvmd.bitnet@wiscvm.wisc.edu,
        levine@a.psy.cmu.edu, alison@devvax.tn.cornell.edu,
        swb@devvax.tn.cornell.edu, farber@huey.udel.edu, hwb@umich1.arpa,
        petry, steve@brl.arpa, craig@devvax.tn.cornell.edu,
        catlett%ncsavmsa.bitnet@wiscvm.wisc.edu, dd0t@te.cc.cmu.edu,
        mills@dcn6.arpa, louie, ntag@csnet-sh.arpa
Status: R

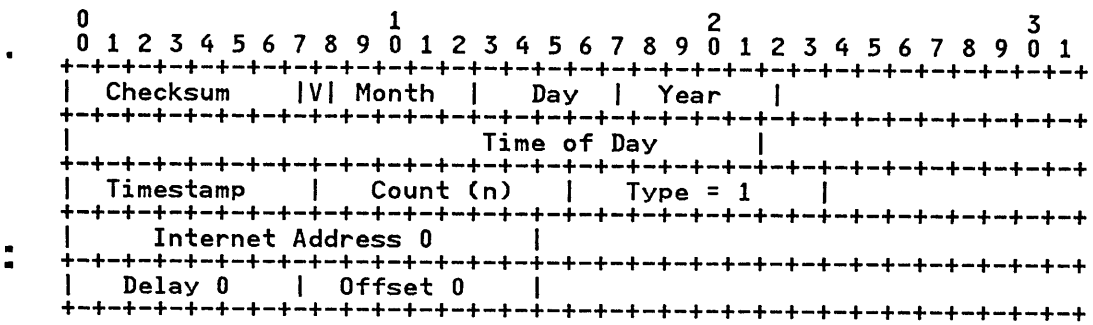
Folks,

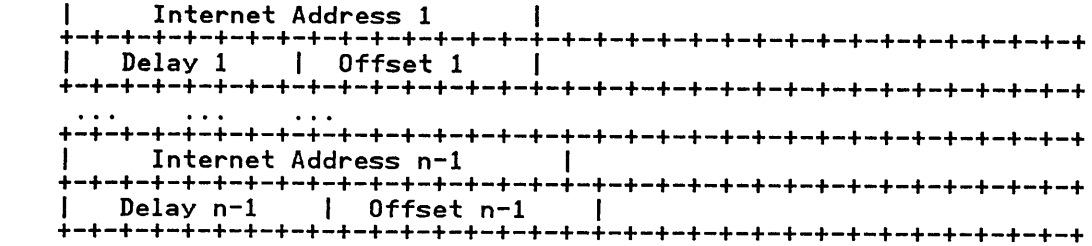
Following is a capsule summary of the DCN Hello routing algorithm as might be implemented in a bsd Unix daemon. A full discussion of the design and implementation in the fuzzball is given in RFC-891, but the reader is cautioned that much liquid has dribbled over the dikes since then. In particular, the particular format of the Hello messages suggested here does not appear in that document. Also, since the document appeared the logical-address translation function in the fuzzies has been changed in dramatic ways. Further details may appear in later messages.

A peer speaking the Hello algorithm pokes a Hello message onto the Ethernet at intervals of 15-30 seconds or so. It contains the current time and date and a list of routing entries, including the address of an Internet entity, the estimated roundtrip delay to that entity and the estimated offset of its logical clock. For the purposes here, the Internet address can be considered a network number (host portion set to zeros).

In the Internet header portion of the Hello message the protocol ID field is set to 63, which has been reserved for local use. The time-to-live field is normally set to two, which is safe for interworking across the cable, but leads to early demise if one of these things inadvertently escapes across a gateway. The source address is that of the sender, while the destination address is the broadcast address of the particular net or subnet. The Ethernet destination address is, of course, the broadcast address of all ones.

Following is the detail format of the Hello message:





Checksum

This is the 16-bit, ones-complement, Internet standard checksum and covers the entire data portion of the IP datagram.

V

This is a one-bit field normally set to zero, but set to one if the date (Month, Day, Year) and/or Time-of-Day fields are not valid for some reason, such as before time synchronization has been achieved.

Month, Day, Year

These are five-bit fields containing the month of the year, day of the month and year (relative to zero = 1972), respectively.

Time of Day

This is a 32-bit field containing the number of milliseconds since midnight UT or the current day.

Timestamp

This is a 16-bit field used in calculating precise roundtrip-delay estimates for point-to-point links. For Ethernets, the value is not used and can be set to zero.

Count

This is an eight-bit field containing the number of routing entries to follow.

Type

This is an eight-bit field containing a code identifying the format of the routing entries to follow. For the format shown here, this value is one (1).

Routing Entries

Each routing entry of a Type-1 Hello message consists of eight octets. The first four octets contain the Internet address, which for gateways would ordinarily consist of the net/subnet address with the host portion set to zeros. The next two octets contain the roundtrip delay estimate in milliseconds, while the final four octets contain the (signed) clock-offset estimate in milliseconds.

For the purposes of interworking over Ethernets, The minimum delay is assumed 100 milliseconds and the clock offset can be set to zero. It is normally

considered polite to include all of the routing entries, even if some are unreachable at the moment. For this case, an unreachable destination is indicated by a delay value of 30000 (30 seconds).

The Hello-message sender does not include the address of its own net among the routing entries, since the receiver(s) deduce this address, along with the delay and offset information, from the information in the first part of the Hello message. To be more precise, the receiver calculates the delay and offset to the sender, then adds each routing entry in turn to these figures in order to obtain the delay and offset on the entire path from receiver through sender to each entity indicated.

In general, it will not be necessary to exchange complete routing information on all nets in the universe between Hello peers. The easiest way to do this is to associate a default path with one of the routing entries sent by one of the participants. For instance, A bsd Unix gateway to the ARPANET might include an entry for net 10 (ARPANET) in its routing entries, which would be interpreted as default by a co-cable gateway of the NSF Backbone system and then passed to the other gateways using its native routing algorithm.

It may be appropriate for gateways at several sites to engage in the above practice, in which case the routing for each site will flow along the minimum-delay path as determined by the routing algorithm. Each ARPANET gateway may indicate any or all of the NSF Backbone nets in EGP Update messages to the core system without ambiguity, under the contrived assumption that its co-cable NSF Backbone gateway, as well as the others connected to it, are "directly reachable" in the RFC-904 sense. This is the contrivance exploited by RFC-985, which is certainly relevant to this discussion.

Dave

From @DCN6.ARPA:mills@dcn6.arpa Sun Jun 22 17:00:49 1986
Received: from DCN6.ARPA ([128.4.0.6]) by trantor.UMD.EDU (5.5d/umd.04)
        id AA00282; Sun, 22 Jun 86 17:00:27 EDT
Message-Id: <8606222100.AA00282@trantor.UMD.EDU>
Date: 22-Jun-86 19:42:25-UT
From: mills@dcn6.arpa
Subject: Backbone tatoo
To: jennings%pucc.bitnet@wiscvm.wisc.edu, dleonard%pucc.bitnet@wiscvm.wisc.edu,
        vanb@nrl-acoustics.arpa, estradas%sdsc.bitnet@wiscvm.wisc.edu,
        choy@ncar.csnet, krol%uiucvmd.bitnet@wiscvm.wisc.edu,
        levine@a.psy.cmu.edu, alison@devvax.tn.cornell.edu,
        swb@devvax.tn.cornell.edu, farber@huey.udel.edu, hwb@umich1.arpa,
        petry, steve@brl.arpa, craig@devvax.tn.cornell.edu,
        catlett%ncsavmsa.bitnet@wiscvm.wisc.edu, dd0t@te.cc.cmu.edu,
        mills@dcn6.arpa, louie, ntag@csnet-sh.arpa
Status: R

Folks,

In answer to a message from Scott, the fuzz do buzz with certain changes in their routing algorithm, instigated upon observation of the unbelievable deluge of rwhoms, random broadcasts, ARPchucks and just plain crud observed on the various local Etherthings. Following is a non-exhaustive summary:

1. In accordance with my message to tcp-ip on Etherbunnies, the fuzz now include a "Martian filter," which flushes all packets with reserved-address

destinations down the black hole. Thus, local broadcasts intercepted on one Ethernet cannot escape to another via the Backbone.

2. Packets black-holed to a reserved address will not result in any ICMP messages at all, whether redirects or unreachables. The fuzz will not respond to ARP requests for such addresses; however, the fuzz at present will gratuitously broadcast the local routing information via Hello message (see companion message) once in a while to the local-subnet broadcast address (presently to the zeros variant - ugh).

3. The fuzz have become somewhat paranoid in the recognition of and belief in ARPs and ICMP redirects. They disbelieve anything with a broadcast Ethernet source address (!) or which attempt to update a fixed gateway or Ethernet destination address in the configuration tables. Many other checks were installed to harden the critter against exotic abuse and to render it safe for the USAN multiplexor.

4. Hans-Werner and I found what appears to be an exotic lockup situation in the DDCMP link-level hardware. We believe this can be fixed by adjusting the Hello intervals and output timeouts so that the same values are not used in both directions on a single link.

5. In an experiment to synchronize the Backbone net to time standards at UMICHnet and DCnet, H-W found that large jitters on the USAN link created problems with the Backbone reachability protocol, lending additional entries to the Hacker's Cookbook on Exotic Ways to Jam a Network.

6. The fuzz can be configured individually to support "promiscuous ARP" (aka "ARP hack"). I have recommended that promiscuity be curtailed, except where local conditions suggest otherwise. The fuzz will not in any case respond for destinations it cannot reach, are reserved or are down, as determined from the routing algorithm.

Hans-Werner may wish to comment on the detail routing and the configuration tables, since these required not a little ingenuity to develop, deploy and test. Turns out I will be at NASA Ames tomorrow (Monday), but will not be free to attend the meeting.

Dave

From @DCN6.ARPA:mills@dcn6.arpa Tue Jul 1 18:05:03 1986
Received: from DCN6.ARPA ([128.4.0.6]) by trantor.UMD.EDU (5.5d/umd.04)
id AA20386; Tue, 1 Jul 86 18:03:37 EDT
Message-Id: <8607012203.AA20386@trantor.UMD.EDU>
Date: 01-Jul-86 21:45:39-UT
From: mills@dcn6.arpa
Subject: On swamprats and fuzzygators
To: jennings%pucc.bitnet@wiscvm.wisc.edu, dleonard%pucc.bitnet@wiscvm.wisc.edu,
vanb@nrl-acoustics.arpa, estradas%sdsc.bitnet@wiscvm.wisc.edu,
krol%uiucvmd.bitnet@wiscvm.wisc.edu, levine@a.psy.cmu.edu,
alison@devvax.tn.cornell.edu, swb@devvax.tn.cornell.edu,
farber@huey.udel.edu, hwb@umich1.arpa, petry, steve@brl.arpa,
craig@devvax.tn.cornell.edu, catlett%ncsavmsa.bitnet@wiscvm.wisc.edu,
dd0t@te.cc.cmu.edu, mills@dcn6.arpa, louie, ntag@csnet-sh.arpa
Status: R

Folks,

Yes, we have met the alligators among the swamprats, and they are us.

Please forgive the somewhat tart tone of this message. In spite of that, it is meant in a sincere, constructive spirit. I spent about twenty years in the university environment and about ten in the industrial one, so I can detect when a test plan is constructed with nobody in the room who understands the technical details involved.

Old Business

Realistically, the milestone charts of Ed's message will cost $200K-$300K in lost usage, which would get somebody fired if that happened around here. This is silly - everything on that chart can be accomplished with the designated team responsible for the installation of the network hardware and software, in particular Cornell, Maryland and Michigan, with a couple of us buzzards hovering on the updrafts.

There is a very real effort necessary for training and hands-on problem solving, but that can be accomplished without delaying operational readiness. There is also messy detail work to get all the sites completely checked out with the right hardware and software components, but this is best accomplished by one-on-one interactions between the installation team and other site personnel, even if that does require telephone tag. At $3000 per day in lost usage my callback slips have very real barbs on them. After a week of all-nighters with Hans-Werner fighting Etherflak from broken campus networks, I ordered a stock of curare for those barbs. After three weeks trying to get local nonsense done (formatting floppies, distributing new versions, etc.), I may go catch the frog myself.

Please let the professionals with plenty of experience and gobs of enthusiasm do their thing and get the network running. You will not find them again in such a dedicated mode until the next big swamp gets drained. They need support, encouragement and a noose around local site support staff (who have been most helpful, but somewhat hard to find sometimes). Their task is to shake down the configuration bugs and demonstrate that the hardware and software components operate correctly, at least up to the Ethernet transceivers. Much of the testing suggested in Ed's message must be performed by these guys, since the details have to be engineered with respect to specific system features and constraints, while using wierd access paths for control and monitoring.

There will come a time, usually called the coming-out party, when correct operation of the network must be demonstrated to the skeptics that have to run and use the thing (and, not incidently, to pay for it). This is the time when the site support staff, especially Cornell, will need hands-on experience and training from the gurus, which keeps them out of mischief during the subsequent shakedown cruise as well. Here is where the weekly conference call may become useful, although I think an active mail exchange via a suitable distribution list is even more useful. Here is also when issues like testing schedules become relevant.

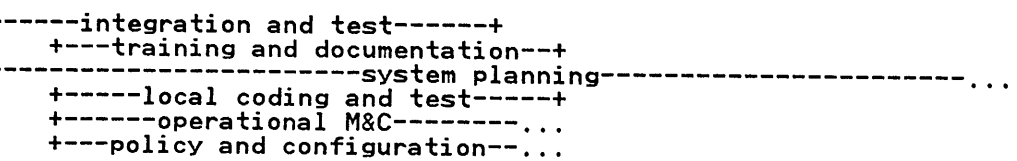
I very much like Ed's weekly update and hope it is continued. When we do this kind of thing in the DARPA community (e.g. SATNET, FATNET, SURAN, etc.) each "contractor" is obliged to contribute to regular reports, which not only serve to propagate status and alerts, but serve to document the progress as well. Doing it electronically is easy, immediate and, incidently, often serves as a robustness test as well (e.g. give everybody a mailbox on a Backbone fuzzy).

New Business

Now we get down to the swamprats and fuzzygators. This project, dear hearts, will get splashed on the pages os Science as a fiasco unless the local-access and routing issues get resolved real quick so the thing can begin earning revenue. Instead of messing in the details of how to get the network running, for which there is a willing and highly qualified staff already in place, the primary action item should be how to connect this thing to the users and the world. This should involve the network gurus at the various sites and probably a good deal of discussion, both locally and within this group as well. The issues go beyond meetings; however, and may involve some coding exercises (such as the Hello daemon I suggested previously or maybe EGP hacks like RFC-975). If we need test plans or white papers, here is where they would be most useful.

Turning to the hazards of milestones (my hat to Ed, who braved those rapids), I offer the following:

Figure 3



1. Integration and Test. Install all hardware and communications components and verify correct operation. Install and debug initial software configuration. Conduct acceptance tests (see below). Primary participants are Maryland, Michigan and lots of frogs.

2. Training and Documentation. Conduct workshops and bakeoffs as required to demonstrate how to operate and manage the net. Train operators and maintenance personnel how to spot problems and trace their causes. Label and inventory everything in sight. Obtain and file a complete set of hardware and software documentation, including user's guides, reference manuals and program listings. All sites and gurus participate, with coordination by Cornell.

3, System Planning. Conduct discussions and meetings as required to resolve issues in global routing and local connectivity. Resolve conflicts in subnetting procedures, broadcasting and so forth. Study and resolve issues for connecting to ARPANET and regional nets. Primary participants are local wizards and gurus.

4. Local Coding and Test. Construct miscellaneous system monitoring and control programs as required. Construct and/or modify gateway tables (EGP, RIP, Hello) and so forth. Modify local hosts if required and fix broken Etherthings. Particpants include SDSC and CMU, with assistance on-call from wizards and gurus.

5. Operational M&C. Develop operational monitoring and control procedures, including the collection and distribution of regular reports and exception events. Determine and document procedures for handling line

outages and equipment failures. Provide trouble-reporting information to users and support staff. Designate primary point-of-contact personnel for use by the NIC, INOC and local sites. Establish and maintain global-information data base such as gateway names and addresses (NIC) and coordinate with local sites. This one sent, with love, to Cornell.

6. Policy and Configuration. Establish standing committee to review network policy and configuration. Evolve procedures for connecting new participants (regional nets?), formulating routing policies and resolving problems. This will be an ongoing effort necessary to the end of time or the swamp gets drained, whichever happens last. This one should be remanded to the NSF (NTAG?) or the Corporation for Open Internet (CON), whichever happens first.

Acceptance Testing

In order to help get the initial testing off the ground, I offer the following fiendish list of confidence builders, commonly called drop tests. Drop tests should be run once at least two of the sites have achieved reasonably stable hardware and software environment. They are designed to determine whether the hardware and software can cope with unexpected outages and unforseen time-dependent behavior. The tests most definitely should not be done by committee and preferably only with guru on duty and on-site hands on switchboard and fusebox. Once this phase of testing is complete it is likely that on-site assistance will seldom be required for ordinary software updates and testing.

The following tests can be made while the fuzzball is isolated from the net.

1. With hardware and software running normally, kill mains power to individual units (CPU, modems, operator terminal, etc.) separately and together. Make sure system reboots automatically and returns to full operational status without manual intervention. Be especially careful about the operator terminal, since without a watchdog timer the CPU can halt forever if the terminal loses power. This test verifies correct CPU, clock and floppy controller configuration, as well as correct disk configuration (startup files).

2. With hardware and software running normally, plug and unplug communication lines (including the operator terminal). Place modems in local and remote loopback and return to normal operation. Make sure system returns to full operational status each time and without manual intervention. Verify system behavior under loopback conditions. The "status" field of the "show <device>" billboard should indicate nonzero and Hello packets should be recorded as sent and received. This test verifies correct modem and cable configurations.

3. Enter the command "reset reset" and verify that the system reinitializes itself (but does not reboot). This is the usually the first thing an operator would do if something breaks and involves a checksum computation on the read-only portions of memory. Enter the command "reset reboot" and verify the system reboots itself as in Test 1 above. These tests verify correct memory/device addressing, as well as reveal configuration problems which may modify read-only portions of memory.

The following tests can be made with at least two fuzzballs on the net.

1. While watching from one fuzzball, perform the above tests on another one

and verify that full connectivity is restored after each test. These tests verify correct link-restart sequences and may reveal possible lockup states requiring staggered timeouts, for example. Just for grins, reset/reboot/repower two or more fuzzballs at the same time.

2. With one fuzzball halted, use the command "set <device> on 200" on the link to it from another fuzzball. Verify the halted fuzzball reboots and resumes normal operation, following which the "200" bit is automatically turned off. This test verifies correct configuration of the CPU and serial interface for this function.

3. Ordinarily, as fuzzballs are connected to the network, the reachability and routing algorithms will operate automatically to sustain connectivity. The correct functioning of these algorithms can be verified by observing the billboards produced by the "set host host", "set <device> show" and "set <device> status" commands. Use the HELP program to obtain further information on the use of these commands. In the case of the DMV11 serial interface, see the source listing for GATDMV.MAC for further information on the status display. These tests verify correct operation of the device drivers, as well as the reachability and routing algorithms.

4. Run the XNET program and use its "connect <address>" command to establish a connection to another running fuzzball. The response will ordinarily be a "port unavailable" comment, which verifies correct packet-level routing and addressing functionality. Then use the "xnet" command to cause the other fuzzball to reboot itself, just as if the "reset reboot" were used by its operator. This test verifies correct network-level routing, addressing and maintenance functionality.

The following tests assume two or more fuzzballs on the net and that the above tests have been run successfully. These tests can be performed either from one of the fuzzballs, a host attached to one of the local nets serviced by a fuzzball or from some other Internet host, assuming that Internet access to the net has been provided to at least one of the fuzzballs, perhaps via USAN.

1. Use the fuzzball PING program (or equivalent in another host) to confirm connectivity and measure delays between the fuzzballs and various Internet hosts. Observe the contents of the ARP cache ("set host" commands) at the relevant gateways to verify correct functioning of the local-routing mechanisms (ARP and ICMP). Use an Ethernet monitor to confirm friendly behavior on the client Ethernet, especially the (lack of) response to gratuitous broadcasts, Hello messages and so forth. Determine whether promiscuous ARP is supported and verify correct operation in that case. Determine whether subnets are supported and verify correct operation in that case.

2. Open a TELNET connection to each fuzzball and log in using designated name and password. Verify all local functions can be performed via the TELNET server program. While logged into a fuzzball open a TELNET connection from there to somewhere else to test the TELNET user program. Do the same thing with the FTP server and user programs and (carefully) with the SMTP user and server programs. Use the HELP program for user information on these programs. These tests verify correct transport and application-level functionality necessary to distribute updates of the system software.

3. Identify an appropriate time server and domain-name server and adjust the fuzzball tables accordingly (HOSTS.DAT). The eventual plan should be to use the Maryland WWVB clock as the master reference for the system; however, in

the interim the network time can be slaved to one of the fuzzballs and the NETCLK program used to obtain time from a convenient Internet time server. This is necessary in order to coordinate the various traffic reports, traps and log files. Access to a domain-name server is not strictly necessary; however, this does conform to the general Internet model, provides general mail support and allows more detail to be included in the system logs. The mail suppport provides a convenient place to stash system monitor and control messages, as well as a place to dump gripes and bug reports. Verify correct functioning of these components by exchanging test messages between the fuzzballs and other Internet hosts.

4. With an operator present at one or more of the fuzzballs, open a TELNET connection to port 87 (TALK port) and verify real-time, two-way connectivity via the operator terminal. This is a useful maintenance feature and avoids tying up telephones for long periods. The mail system is also useful as an order wire.

When testing of the entire network has progressed to this point, its basic functionality and survivability has been demonstrated. The next series of tests is designed to stress the traffic-handling capability and verify correct functioning of the resource-management and fairness algorithms. There are lots of ways this can be done, most involving some sort of somewhat-broken, local-net fire hose (a Unix system will do), one of which is outlined below. Be advised, however, that the fuzzballs are not particularly lush with buffers and can be overhwhelmed fairly easily with vast numbers of fire hoses on their client nets. The design philosophy has intentionally been biased towards minimum delay at the expense of loss rates. Thus, the normal response to congested resources such as packet buffers is to drop packets. Also, be advised the routing algorithm is sensitive only to baseline delays; in other words, it does not change in reponse to traffic loads or congestion. Finally, in its present form, the fuzzball does not send ICMP source-quench messages.

The following tests assume some or all fuzzballs operating on the net and that all have successfully completed the above tests. The tests are designed to use the native testing tools of the fuzzballs themselves. Use the HELP program for additional information.

1. Contrive multiple instances of the PING program as traffic generators and measurement hosts. This program can generate packets of constant length or uniformly distributed over a selected range and measure roundtrip delays and loss rates. The data can be displayed in real time, collected in a histogram or saved in a file for later analysis.

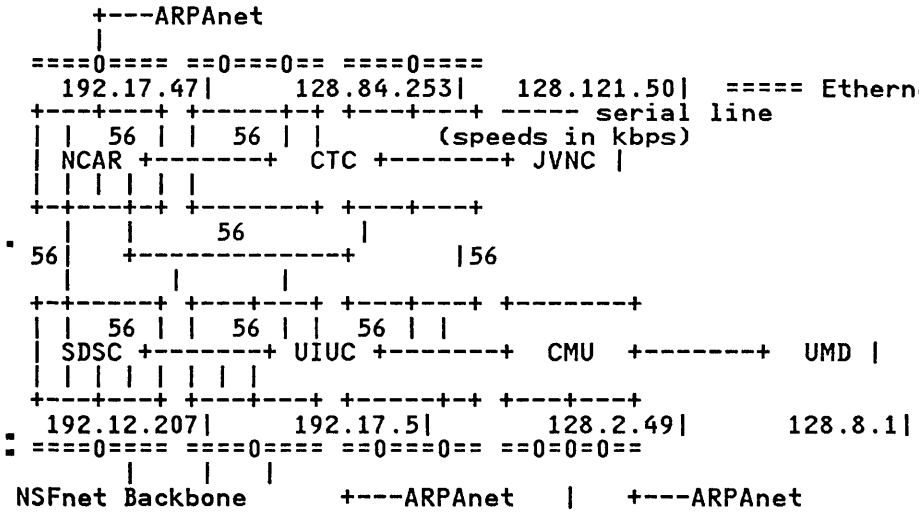
2. Contrive multiple instances of TELNET, FTP or SMTP to simulate real traffic. Use the "set inp ccb" commands to reveal TCP retransmission timeouts, packet traces and similar information. Trace information can be recorded in the system log for later analysis. Other information can be included in the system log using the "out log:" command.

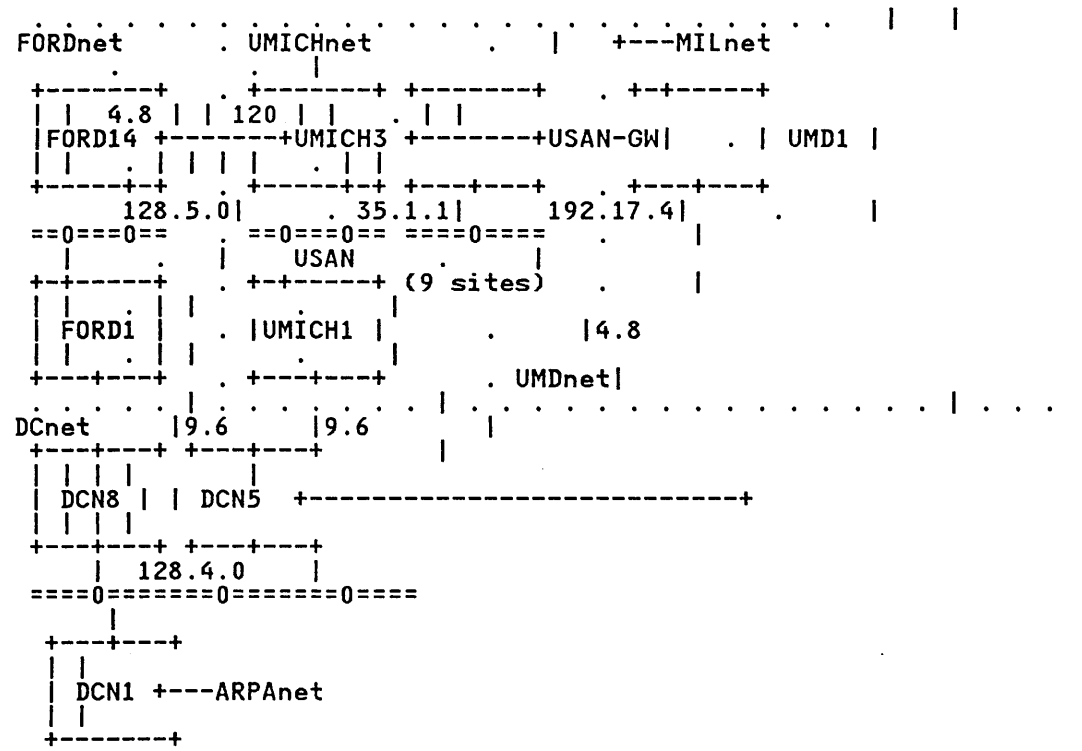
There are many directions a testing program can take at this point; however, once testing has progressed to this point, it is probably wiser to declare a coming-out party and start collecting fares with real users.

Dave

From tcp-ip-RELAY@SRI-NIC.ARPA Mon Jul 7 03:04:57 1986
Received: from SRI-NIC.ARPA ([10.0.0.51]) by trantor.UMD.EDU (5.5d/umd.04)

```
        id AA29099; Mon, 7 Jul 86 03:02:53 EDT
Message-Id: <8607070702.AA29099@trantor.UMD.EDU>
Received: from D.ISI.EDU by SRI-NIC.ARPA with TCP; Sun 6 Jul 86 22:18:17-PDT
Received: FROM DCN6 BY USC-ISID.ARPA WITH TCP ; 7 Jul 86 00:51:09 EDT
Date: 07-Jul-86 04:33:16-UT
From: mills@dcn6.arpa
Subject: On routing freedom and statutes of liberty -or- Ether is a gas
To: tcp-ip@sri-nic.arpa
Status: R

Folks,

Don't start the following story unless you enjoy solving puzzles and have a
few minutes to study and reflect on the issues. Be advised it is highly
technical, not without personal bias and may leave some of you with elevated
cranial energies. I especially would like our ANSI/ISO protocol designers to
take this thing into their committees and spread mischief (Overheard in X3.S3:
"Ya mean those Internet buzzards are doing THAT!?").

The cast of characters includes the NSFnet Backbone, which is now being
installed between six Supercomputer sites, plus a bunch of Ethernets at those
sites. Some of the sites are also interconnected via the USAN net, which uses
a multiple-access Vitalink/TransLAN satellite channel, and some are connected
to the ARPAnet via the Ethernets and other nets and gateways. The Backbone
gateways, as well as some of the USAN and ARPAnet gateways involved, consist
of LSI-11 "fuzzball" systems, which are reasonably good players of the ARP and
ICMP game, as well as run their own routing algorithm.

The following schematic shows the configuration of these swamps expected by
late August. All of the nodes and lines shown are already installed, except
for some Backbone line-interface equipment. All of the non-Backbone nodes
have been in operation for some time, while three of the Backbone nodes (SDSC,
NCAR, Cornell) are presently interconnected and in operation. Only the
fuzzballs are shown, since this discussion concerns primarily them; however,
you should recognize there are lots and lots of other hosts on these nets and
that some nets include many different media besides Ethernets.
```





The players are:

NCAR National Center for Atmospheric Research
CTC Cornell Theory Center (hardware integration and net operator)
JVNC John von Neumann Center
SDSC San Diego Supercomputer Center
UIUC University of Illinois/University of Chicago (project management)
CMU Carnegie-Mellon University
UMD University of Maryland (software integration)
UMICH University of Michigan (installation and test)
DC Fuzzball creche in Vienna, VA
FORD Ford Scientific Research Labs in Dearborn, MI

At the moment, all of the ARPAnet/MILnet gateways except DCN1 (aka DCN-GATEWAY) include only their own directly-connected client nets in EGP updates sent to core gateways. DCN1 temporarily includes some of the other nets for debugging and test purposes. Therefore, traffic for the other nets must wander to DCN1 and swamp-fuzzball trail to USAN and Backbone trunks (UMD is presently not operational). Eventually, Backbone connectivity may be provided to the ARPAnet by one or more gateways at CMU, CTC or UMD as well. Also at the moment, the Ethernets at some Backbone sites are connected via USAN to each other and via USAN-GW at UMICH to the Interworld. As you can see, there will be a lush supply of routes available, with many sites enjoying connectivity via the Backbone, USAN and ARPAnet paths simultaneously.

Believe it or not, traffic actually flies these airways, although sometimes

landing in very strange airports. The fuzzballs shown operate an adaptive routing algorithm which selects primary routes based on minimum delay and can select alternate routes based on IP type-of-service field and other factors. Casual observation of the DCnet Ethernet reveals there may be a lot of local-site problems yet to solve. For instance, I spotted two hosts on Cornell local nets working each other via the DC Ethernet (!?!) the other day. Ya gotta see to believe.

The ring of hosts and Ethernets at DCnet, UMICHnet and FORDnet has been a valuable prototyping facility, which is its primary service function. Alternate routing in case of failure requires ICMP Redirect and ARP functions to work properly, both in the fuzzballs and in other network hosts, which are represented by a wide variety of VMS, Unix and related systems. A problem in this area is what prompted this message.

Recently, Hans-Werver Braun of UMICH and I endured scary experiences while teaching Etherbunnies, fuzzygators and other strange creatures to swim in these swamps. Especially enlightening was USAN, with its nine rambunctious Ethernets, all piled onto the same channel and babbling everything from DECnet and XNS mumbles to Unix rwho shrieks zipping about like lost cosmic particles. It took us two weeks at DefCon 5 to harden the fuzzball silos (which added a couple of dB of their own routing broadcasts) and make space safe for Backbone debug and test. Some of the lessons learned have already been broadcast for public enjoyment and may even have medicinal value.

Additional observations are herewith submitted for your entertainment, education and as a basis for comments and suggestions. What I hope we all get out of this exercise is not so much a blueprint for how to deal with the incredibly complicated brushfires we know are circling the horizon (to quote Dave Clark), but rather as an experiment and proof-of-concept suggesting generic issues that need further study and resolution before we send out the fire brigade.

Multiplexed Ethernets

Most of the interesting lessons were learned on USAN, which radio amateurs will recognize as similar to the pileup when Pitcairn Island shows up on 20 meters. The USAN design was never intended to handle the bedlam of nine simultaneous flocks of squawking rhos, rips and other broadcast honkers, much less the blatant ICMP squawks from the poor creatures that don't understand them. The problem is, of course, that those of us who jimmied the Ethernet protocols while draining our own swamps never thought of making a single cable safe for multiple nets and subnets at the same time. Hans-Werner and I found it useful to forget that the cable is normally protected from crosstalk between neighbor nets and pretend it is a willy-nilly-access packet-radio channel instead.

You may not like this model and prefer a much more carefully regimented and regulated approach. We would like to understand what-if first and learn all we can before the wires are insulated and the lessons have to be relearned under meltdown conditions when the insulation wears off somewhere. Even if we agree multiplexed Ethernets are beyond the pale, this exercise may reveal how to harden the hosts and gateways against rogues (and jammers) that might occur from time to time. Therefore, we simply connected everything up and let the packets fly.

Hans-Werner and I are still catching our breath, some wheezes of which can be found in the following. We are putting together an almanac, which may appear

as a footnote to RFC-985, to teach lessons something like the following: Multiplexed Ethernets are extremely complex and delicate, but may represent a useful solution in exceptional cases. If you are silly enough to contemplate such folly, do it in the following way...

Type-of-Service Routing

Our research community has been stalking the type-of-service routing issue for awhile now and may have fallen in the wrong pits. It turned out to be easy for the fuzzballs to take a first cut at this simply by extending the address fields in the routing tables to include the IP type-of-service field (actually, the extension consists of a single octet, with each bit corresponding to each of the eight combinations of the Throughput, Delay and Reliability bits).

The hard part begins when you try to make ARP, ICMP error messages and ICMP Redirects work in that model. The goal would be to maintain possibly separate routes for every combination of type-of-service specification. The main problem is that the packet formats are not rich enough to distinguish this information to the detail required. In addition, much of the existing host software must be changed in nontrivial ways (e.g. the ARP cache gets an extra octet, etc.). The fuzzball routing data structures already include such provisions, but the algorithms have not been evolved to exploit them yet.

Alternate Routes

It was our intention to explore the consequences of trying to provide alternate routes should something quit and where not all paths were monitored by the fuzzball routing algorithm. For instance, if a backbone trunk were broken, it might be possible to route out the ARPAnet and back in again (please table administrative discussion - we just want to explore how the darn thing might work, not whether it would be allowed or not).

As an experiment, The fuzzballs were fiddled so that, if an internal link controlled by the fuzzball routing algorithm broke, traffic could be directed via a designated backup path. This was tested in the DCN5 - UMD1 link, with designated backup path via the ARPAnet/MILnet gateways on each net, which is ordinarily controlled by EGP and the core system. The effect is that, when the DCN5 - UMD1 link is up, traffic flows via it, but, when the link is down, traffic flows the long way around via ARPAnet, MILnet and thousands of gateways.

This was a cute experiment and worked just fine; however, its success depends on carefully engineered tables and delicate assumptions about the functionality and dynamics of both the fuzzball and EGP algorithms. More study and experiment is needed here.

Subnets

All of the class-B nets shown in the schematic are subnetted, with the third octet interpreted as the subnet number. The class-A UMICHnet is subnetted as well. Good subnetting practice is to avoid the use of subnet-number fields containing all zero bits or all one bits, since this can lead to confusing interpretation of broadcast scope, for example. DCnet and FORDnet fall victim to this observation. I can't believe for a minute that vendor products without subnetting capability will have a long life in this era.

Broadcast and Don't-Know Addresses

The Internaut Handbook specifies that the local-subnet address with all ones in the host portion should be interpreted as "broadcast" and that the address with all zeros should be interpreted as "don't know." Under these interpretations, the former is legal only in destination-address fields, while the latter (used by a diskless workstation while RARPing for its address, for example) is legal only in source-address fields. Where subnets are in use, the scope of this interpretation extends only to the given subnet, if the subnet-number field is neither all zeros or all ones, and to the entire network of subnets if either or all zeros or all ones.

We observed numerous abuses of this model, including the use of zeros as the broadcast address (older bsd Unix) and various tangles with broadcasting in a subnet environment. In point of fact, it doesn't matter which convention is followed in a particular subnet, as long as all hosts and gateways on that subnet understand which is in use, as well as the subnet mask, and agree never to propagate either local-broadcast or local-don't-know datagrams beyond the gateways. The same consideration applies at the network-of-subnets level, of course.

For reasons that should be evident from the following, I believe the use of zeros and ones in the subnet-number field and the above interpretation should be avoided in favor of explicit broadcast agents. One implication of this model is that broadcasts would never be propagated by a gateway. I further believe that a very careful coupling should be maintained between the semantics of the Ethernet broadcast/multicast addresses and IP broadcast addresses; otherwise, the implementation is forced not only to carry all kinds of wierd semantics up the protocol stack, but its error detection is seriously handicapped as well.

A paranoid receiver may well check that, when an IP packet with an Ethernet broadcast/multicast destination address arrives, the destination-address field must contain the IP subnet-broadcast address or the packet is discarded. This would have the effect of disallowing random Ethernet broadcasts to designated IP destinations if the sender had a broken or unimplemented ARP, for example. It would also disallow cross-net routing broadcasts, such as the fuzzballs use to manage USAN routing. More thought is needed here.

Broadcasting Semantics

Nothing we found exhibited stranger behavior than the broadcast semantics of the various Ethernets. The most disruptive thing by far was the tendency of some receivers to "helpfully" relay broadcast packets with destination IP address other than the receiver to their "intended" destination. An innocent rwho from a host on one subnet of a multiplexed cable ignites instant abuse of the gateways if the hosts on another subnet do this. In extreme cases the network can fall into a debilitating, oscillatory state (called meltdown), where the entire cable bandwidth is consumed by these packets.

A general principle of nuclear engineering is that reactor meltdown is possible only if more energy gazouta the reactor than gazinta it. Ethernet meltdown cannot occur if it can be gauranteed that no more than one gazouta packet can ever be produced by a single gazinta packet at a node. Thus, a forwarded broadcast packet (hereby named a Chernobyl packet - you first heard it here) can produce meltdown only if more than one receiver is involved. Of course, if a Chernobyl packet is assigned an Ethernet broadcast address, the meltdown would occur within milliseconds.

I believe no receiver (host or gateway) should ever forward broadcast packets onward to a subsequent destination, unless the receiver is an explicitly designated broadcast agent which explicitly understands and maintains the spanning trees and routing algorithms necessary to reach the intended destination without meltdowns.

What are the groundrules of a broadcast service? Those such as rwho, rip and fuzzball routing do not involve an explicit reply from each of possibly many receivers. Those such as ARP, RARP and ICMP Address Mask may produce a meteor shower of responses if multiple receivers can respond. In some cases where efficiency can be sacrificed for reliability, meteor showers may be acceptable, but in others this would be disruptive. A case might be made for datagram services like the above and maybe others, but this would be silly for connection-oriented services. Experiments with broadcast TCP service and unhardened fuzzballs led to hilarious scenarious leading to marginal meltdown, suggesting that a multiple-destination semantics may have to be carried up the protocol stack anyway, at least for error detection.

ICMP Error Messages

The Internaut Handbook suggests ICMP error messages should be returned to the ultimate source if a datagram cannot be delivered to its intended destination. Some hosts interpret this literally and return ICMP error messages if the protocol or port fields do not match a service provided by the receiver. We discovered this quickly in the case of the fuzzball routing algorithm, which broadcasts Hello messages on protocol 63 (private use) from time to time, each one creating a shower of ICMP error messages.

I believe no receiver (host or gateway) should ever return an ICMP error message unless it can determine with fair reliability that any other copies that might be wandering around the network will be routed by that receiver and that the same ICMP error message would be produced in each case. This is a general statement and applies to scenarios other than broadcast. One implication, for example, is that ICMP error messages must be considered non-deterministic, since one path can be temporarily stoppered-up while the routing algorithm is thrashing and duplicates are successfully negotiating another path.

With respect to broadcast, ICMP error messages should never be sent in response to a received broadcast packet. Another way of looking at it is that no message should ever be sent if its semantics would involve the broadcast address as the source address (IP or Ethernet) in the packet.

Subnet Masks

No gateway implementation known to me (except the fuzzball as of today) supports the ICMP Address Mask messages described in RFC-950. If a gateway joins two subnets, it has to know which subnet the request is for, so it can determine the proper mask. If the requestor knows its own address, it can broadcast the request and the gateway(s) can determine which subnet from the source IP address of the request. If it does not know its own address, it must use RARP to find it (the alternative suggested in RFC-950 is simply too bizzare to contemplate in this environment).

This is the only known case that requires broadcast of an ICMP message, which not only complicates the implementations, but suggests that maybe something is broken in the model. The real problem is that the ARP/RARP semantics are simply not rich enough and should be expanded to include the mask in the first

place. For instance, a RARP reply should include not only the specific host address, but also the address mask associated with its subnet. Also, if promiscuous ARP is used to bypass a specific gateway address, an ARP reply should include the address mask associated with the subnet the address belongs to (if known).

The requisite semantics and packet formats might not be hard to provide, since ARP and RARP are quite generic. While adding the subnet mask, the type-of-service mask should also be added. The ICMP Address Mask messages should be junked.

Martian Filters

The idea of Martian Filters originated as a weapon to combat datagrams carrying bogus destination addresses (like 127.0.0.0), which can be emitted by broken bsd Unix systems. These datagrams sometimes escape their local net and are found wandering about the Internet and creating a significant hazard to swamp navigation. An unbelievable brew of this stuff was found sloshing over USAN, which prompted my earlier message on this subject.

Martian Filters search for "reserved," "broadcast" and "don't-know" addresses identified in the Assigned Numbers list and discard datagrams (without ICMP error notification) to those destinations, unless addressed to the host/gateway directly. Use of this filter prevents the recipient from forwarding a broadcast datagram back onto the cable or sending disruptive ICMP error messages in the case of unknown protocols or ports appearing in broadcast datagrams. We found these filters to be absolutely necessary to avoid chaos (pun) on USAN.

When subnets are in use, a receiver may not know that a particular IP address in fact represents a broadcast, except that it presumably came in on a datagram with a broadcast address in the Ethernet destination-address field. The Martian Filter we used is subnet-independent and filters out only the well-known network-level broadcast and don't-know addresses. However, the fuzzballs, at least, know what subnet they are on and grab local-net broadcast and don't-know addresses before they can be sent onward and create mischief.

If it can be agreed that the network-of-subnets semantics mentioned above (i.e. disallowed) is correct, then the broadcast and don't-know filters can be implemented more efficiently and effectively. More study and consensus is needed in this area.

Asymmetric Paths and Redirects

>From the above diagram you can see that, when airport DCN8 closes, flights to FORD1 can continue via DCN5, UMICH1/3 and FORD14 as determined by the the routing algorithm. ARPs for FORDnet will then be returned by DCN5 and all other hosts will return IMCP Redirect messages as expected. Now, it turns out that some silly host on UMDnet thinks the slickest airway to FORD1 is out MILnet, back in DCN1 and radar vectors to FORD1, but the DCnet controllers know the smoothest air is via DCN5 and UMD1. Well, all that asymmetry works, too, until such time as DCN8 opens up again.

When the routing algorithm finds that DCN8 is open, everything shuffles as expected in DCN5 and DCN8; however, the other DCnet hosts sharing the Ethernet may not realize this, since their minimum-delay path is still via the Ethernet. Ordinarily, these hosts will update their routing tables correctly when they send traffic to FORD1, since DCN5 will redirect that traffic to

DCN8.

The problem lies in the question: Which local-net host (Ethernet address) does DCN5 send the redirect to? If it doesn't know better, it simply looks in its routing tables for the sender (UMD host) and determines the route via DCN5 to UMD1, but it would be nonsense to send the redirect there. However, the incoming traffic actually came via DCN1, so the redirect should really be sent to it instead.

Good implementation technique tries to separate protocol layers cleanly, which suggests Ethernet addresses be propagated no further up the stack than absolutely necessary. Since an ICMP redirect does, after all, have an IP header, the temptation is to treat it like other ICMP messages as a wart on the side of the network (IP) layer. But other ICMP messages don't have this insiduous coupling to the local-network (Ethernet) layer, so no provisions were made in my original design to save the Ethernet addresses at all.

Well, I shambled back to the hanger and tinkered the fuzzware, so now all controllers are on the same frequency. My solution was to remember the Ethernet source address as each Ethernet packet arrives and stuff it in the destination address of the outbound ICMP Redirect message.

Some of you may remember my previous messages which argued against propagating Ethernet addresses up the stack and suggest I got what I deserved. I am still opposed in principle to spreading local-net layer semantics (like broadcast addresses) outside that layer and believe such semantics should be re-created at the network level (e.g. use IP broadcast address) as necessary. Unfortunately, for reasons mentioned a paragraph or two back, I may have to eat them words.

Dave
---

2) Initial Delay Experiments with the University Satellite Network (USAN),

Hans-Werner Braun (UMich)

Additional Material

9 June 1986

Hans-Werner Braun

University of Michigan

(HWB@GW.UMICH.EDU)

Initial delay experiments with the University SAtellite Network (USAN)

This paper documents the initial delay experiments done with the University SAtellile Network (USAN). USAN includes the following eight sites:

- National Center for Atmospheric Research (NCAR), Boulder, Colorado
- Oregon State University, Corvallis Oregon
- University of Illinois,Urbana, Illinois
- University of Maryland, College Park, Maryland
- University of Miami, Miami, Florida
- University of Michigan, Ann Arbor, Michigan
- University of Wisconsin, Madison, Wisconsin
- Woods Hole Oceanographic Institution, Woods Hole, Massachussetts

NCAR is the hub of the USAN network, broadcasting data to all the remote sites at 224 kilobits per second. The remote sites usually have a 56 kilobits per second dedicated backlink to the hub. The anticipated topology is illustrated in figure (1). At the time of this writing only NCAR, Illinois, Maryland and Michigan are operating. The Michigan site became usable in late May.

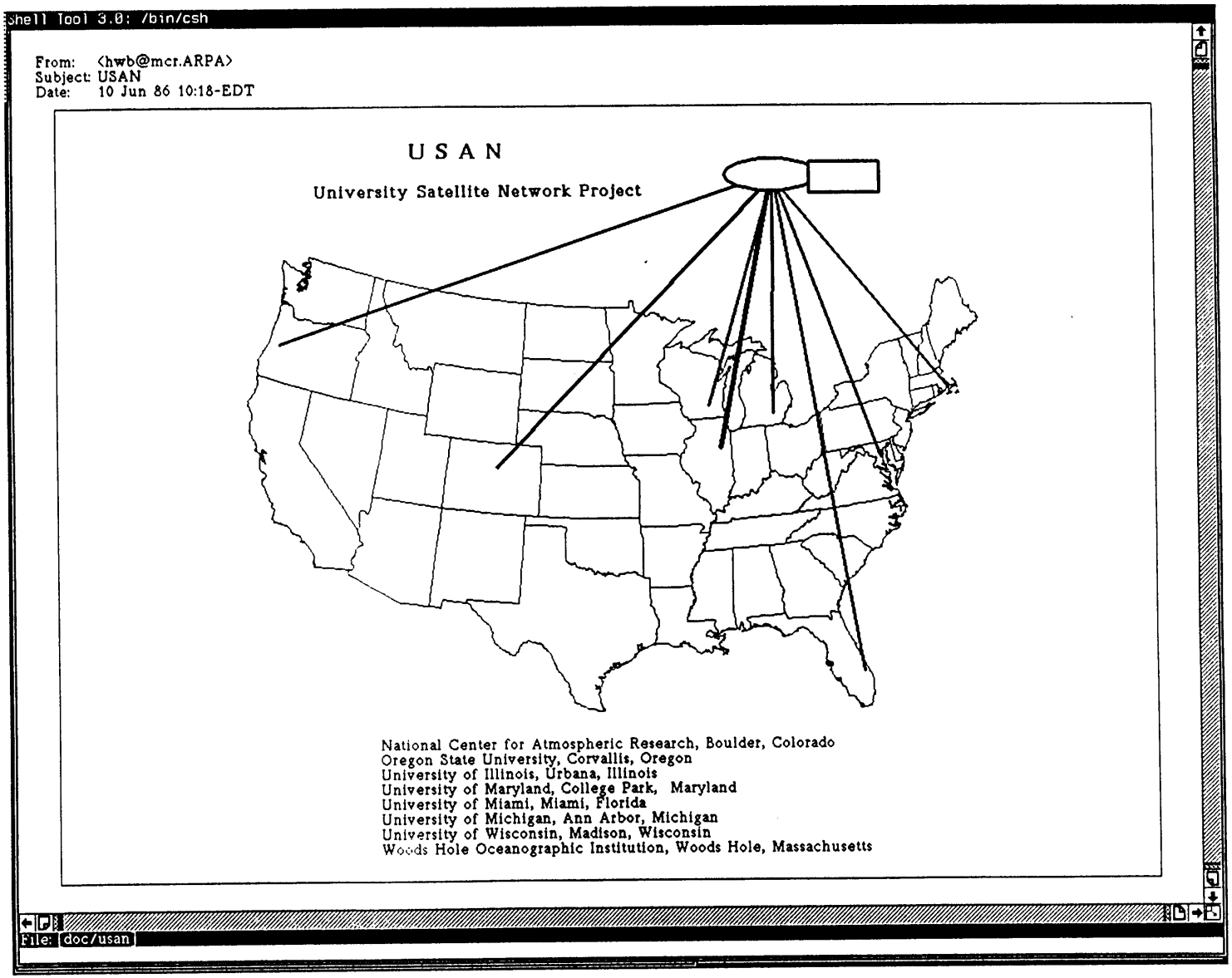

Shell Tool 3.0: /bin/csh
From: <hwb@mcr.ARPA>
Subject: USAN
Date: 10 Jun 86 10:18-EDT
USAN
University Satellite Network Project
National Center for Atmospheric Research, Boulder, Colorado
Oregon State University, Corvallis, Oregon
University of Illinois, Urbana, Illinois
University of Maryland, College Park, Maryland
University of Miami, Miami, Florida
University of Michigan, Ann Arbor, Michigan
University of Wisconsin, Madison, Wisconsin
Woods Hole Oceanographic Institution, Woods Hole, Massachusetts
File: doc/usan

Figure (1)

Figure (2) illustrates Michigan's current connectivity. The Michigan Ethernet has a gateway to the DCNet in Virginia and Maryland, which in turn is connected to the Arpanet, to UMDnet, and to Fordnet. Fordnet connectivity exists as an alternative path from the University of Michigan to DCNet. Also connected to the Michigan Ethernet is an SDSC Remote User Access Computer that acts as a gateway to the MFE protocol-based SDSCnet. Michigan's current primary gateway is connected via a 150 kilobits per second HDLC link to an access gateway (USAN-GW) into the USAN Ethernet. The USAN Ethernet is connected to the Vitalink TransLAN bridge, which itself links to the USAN satellite groundstation equipment. Much more equipment, irrelevant for this document, is on Michigan's Ethernet, e.g., an Apollo network local to the University of Michigan with about 150 nodes, and, of course, the Merit Computer Network itself is also attached.

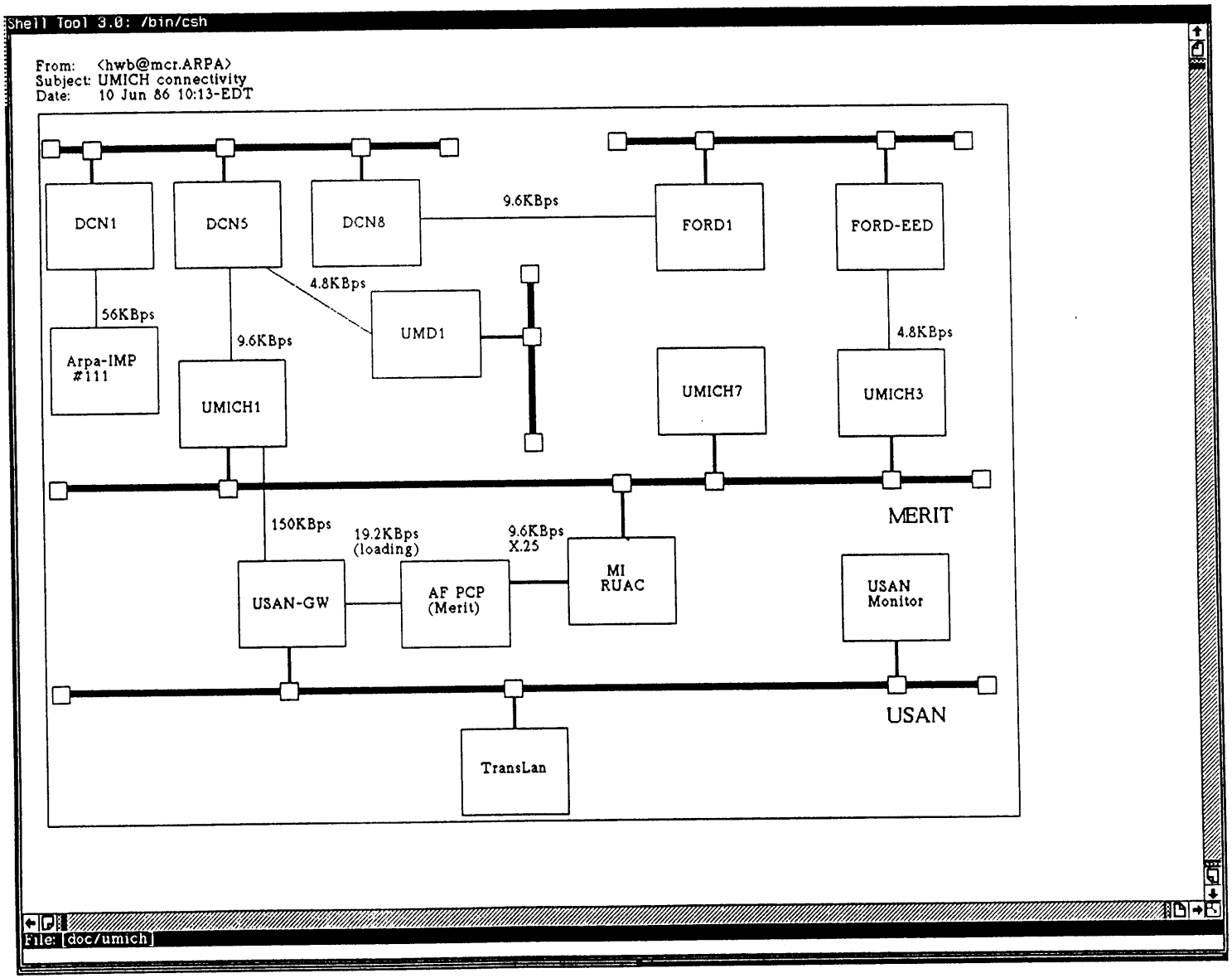

Shell Tool 3.0: /bin/csh
From: <hwb@mcr.ARPA>
Subject: UMICH connectivity
Date: 10 Jun 86 10:13-EDT
DCN1
DCN5
DCN8
9.6KBps
FORD1
FORD-EED
4.8KBps
56KBps
UMD1
Arpa-IMP #111
9.6KBps
UMICH1
UMICH7
UMICH3
4.8KBps
MERIT
150KBps
19.2KBps (loading)
9.6KBps X.25
MI RUAC
USAN-GW
AF PCP (Merit)
USAN Monitor
USAN
TransLan
File: doc/umich

Figure (2)

Figure (3) outlines how the USAN fits into the current NSFnet topology with its attachment to NSFnet backbone nodes. It should be pointed out that the Fordnet is only used for local network research and has nothing to do with the NSFnet per se. The existence of the DCNet/Fordnet/UMICH triangle has proven worthwhile for the implementation of an NSFnet-like environment prior to the installation to the NSFnet itself and has aided NSFnet development.

A possible scheme for attaching USAN and the NSFnet backbone to one Ethernet is outlined in figure (4). An NSFnet backbone hub could either be connected to a production Ethernet, or there might be a gateway between the two. These Ethernets will have hosts attached that declare one or more gateways as their default gateways. These default gateways, in addition to making routing decisions, can function as access gateways to the concatenated USAN TransLAN Ethernet.

The specific environment applicable to this report is illustrated in figure (5). UMICH1 is a gateway node attached to an Ethernet at the University of Michigan, using the Internet network number "35". UMICH1 is connected to USAN-GW via a 150 kilobits per second link, where USAN-GW is logically homed into the USAN Internet network, at net "192.17.4". USAN-GW attaches to another Ethernet that is also connected to the TransLAN equipment and therefore to the USAN itself. The two remote ends used here were NCAR and UIUC. NCAR has a Sun workstation connected to their Ethernet, as well as one of the NSFnet backbone nodes. The Sun has an Internet address in the same USAN range, while NSF-NCAR belongs to net "192.17.47". The Illinois

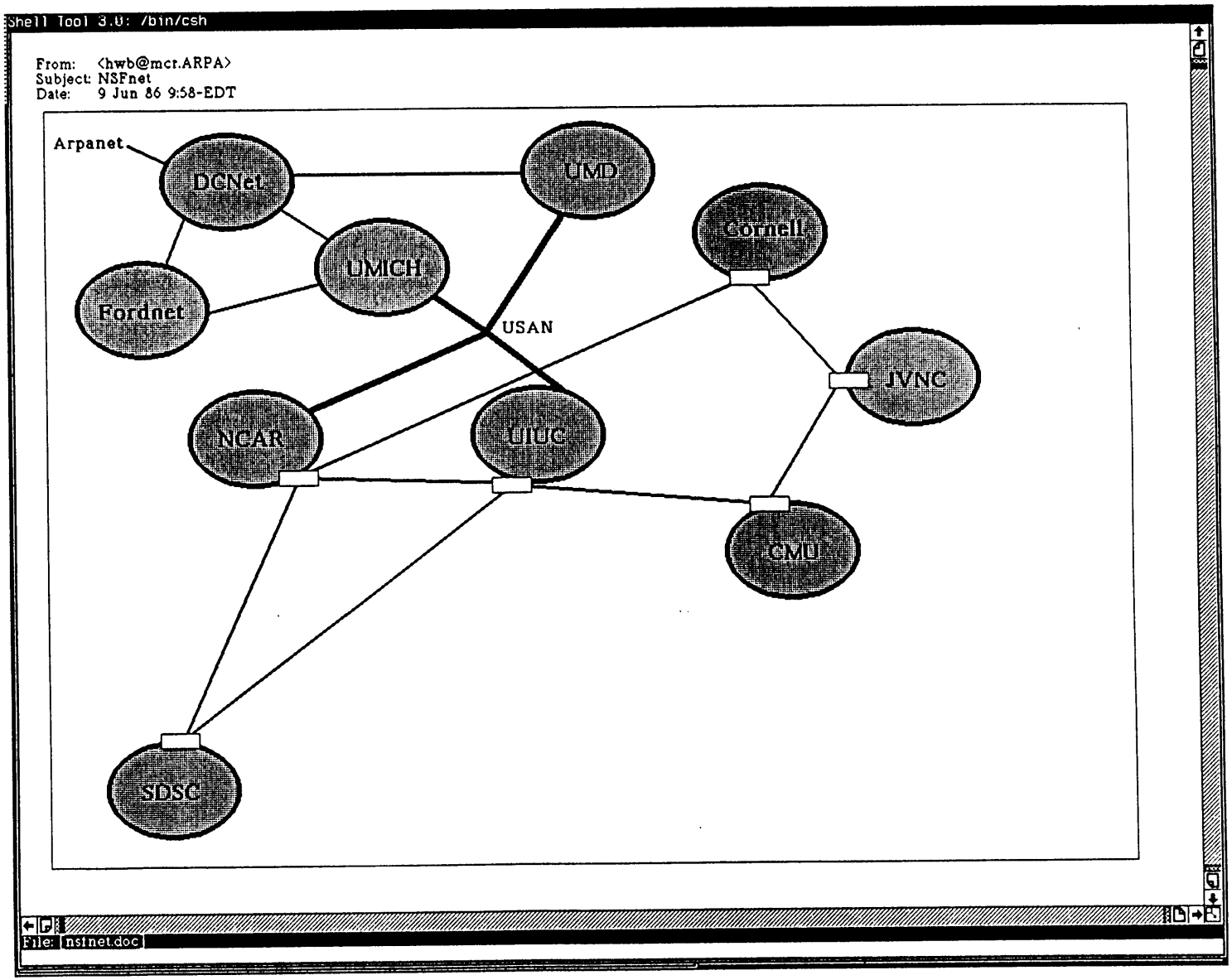

Shell Tool 3.0: /bin/csh
From: <hwb@mcr.ARPA>
Subject: NSFnet
Date: 9 Jun 86 9:58-EDT
Arpanet
DCNet
UMD
Cornell
UMICH
Fordnet
USAN
JVNC
NCAR
UIUC
CMU
SDSC
File: nsfnet.doc

Figure (3)

node in the diagram is also homed into the USAN address range. Connectivity among these three nodes is already possible, even though tunnel-routing techniques had to be applied to allow connectivity to the NSF-NCAR as well as to the Illinois node, since both nodes had no routing set up for the net "35".

It will very likely turn out that Type of Service routing will play a major role in the USAN/NSFnet environment, where it will become necessary to base routing decisions on a specific application. A good example here is the transfer of files, where it is important to have high throughput, and the fairly high satellite delays are not disturbing. On the other hand an interactive application might require low delays and therefore a surface-bound circuit would be chosen over a satellite link. It might even prove worthwhile to send the FTP data acknowledgements via surface links while transmitting the data itself via satellite. This would partly compensate for situations where TCP windows close in high delay environments and where the data transfers becomes bursty.

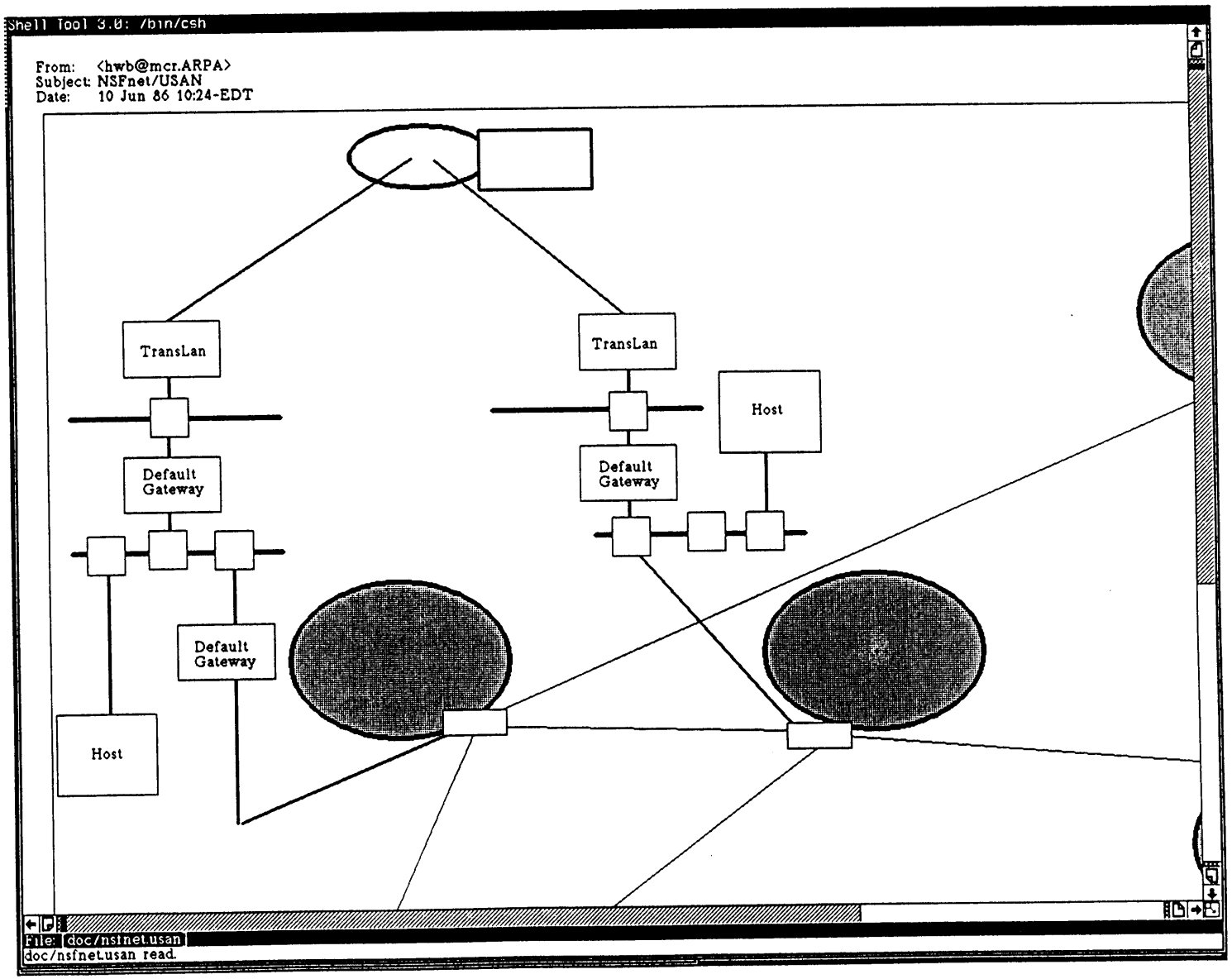

Shell Tool 3.0: /bin/csh
From: <hwb@mcr.ARPA>
Subject: NSFnet/USAN
Date: 10 Jun 86 10:24-EDT
TransLan
TransLan
Host
Default Gateway
Default Gateway
Default Gateway
Host
File: doc/nsfnet.usan
doc/nsfnet.usan read.

Figure (4)

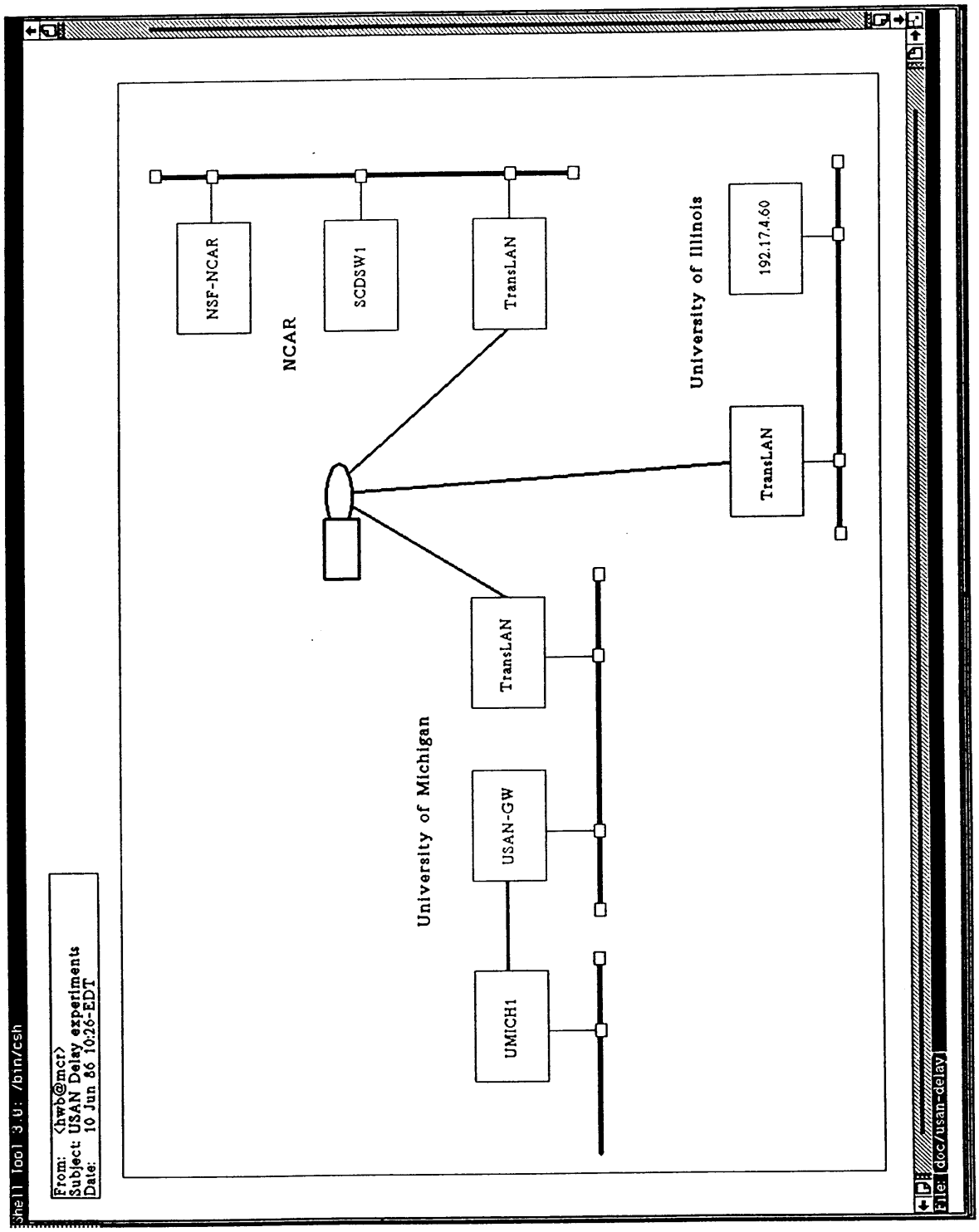

shell tool 3.0: /bin/csh
From: <hwb@mcr>
Subject: USAN Delay experiments
Date: 10 Jun 86 10:26-EDT
NSF-NCAR
SCDSW1
TransLAN
NCAR
University of Illinois
192.17.4.60
TransLAN
TransLAN
University of Michigan
USAN-GW
UMICH1
File: doc/usan-delay

Figure (5)

Delay experiments were performed with the NCAR-SCDSW1 node, as well as with the Illinois node.

The expected length of time for a single satellite hop (up+down link) is about 250 msec, as limited by the speed of light.

IP together with ICMP has an interesting feature that allows measurement of round trip delay times. This is done by using ICMP Echo Requests, which in many implementations get turned around at the remote end while also being transformed into an ICMP Echo Reply. The sending end can now measure the delta time between sending the ICMP Echo Request and the receipt of the ICMP Echo reply.

As a first test I measured the delays between UMICH1 and SCDSW1, which uses a two-hop satellite link: one hop between Michigan and NCAR and one hop for the way back. The results of one thousand samples were logged into a file on the UMICH1 Fuzzball, and after the file was transformed into a more readable format it was fed into the Diamond document preparation and MultiMediaMail system on a Sun workstation. Diamond has spreadsheet support and the result of this is shown in figure (6). This diagram only shows delays in the range between 545 msec and 595 msec, which leaves out a few samples that had higher round-trip delay times. Figure (7) shows the last 50 of the 1000 samples after sorting them for delay times. This diagram also includes delay times higher that 595 msec. If we consider the overhead for the two end systems, UMICH1 and SCDSW1, as well as for the gateway in the middle (USAN-GW), it is seen that the delay is close to an expected double-hop

result.

At the first day where the USAN link between Michigan and NCAR came out I had some conversation with Jon Postel during which he also tested round trip delays from ISI in California via the Arpanet, DCNet, UMICH and USAN. He sent 245 packets of which 225 were received back at ISI as ICMP Echo Replies. This means an 8% loss rate. The minimum round trip delay time here was 1.1 seconds, the maximum was 5.7 seconds, while the average round trip delay was 1.5 seconds. Delays between UMICH1 and NSF-NCAR were also checked and found to be in the same range as in the UMICH1-SCDSW1 case.

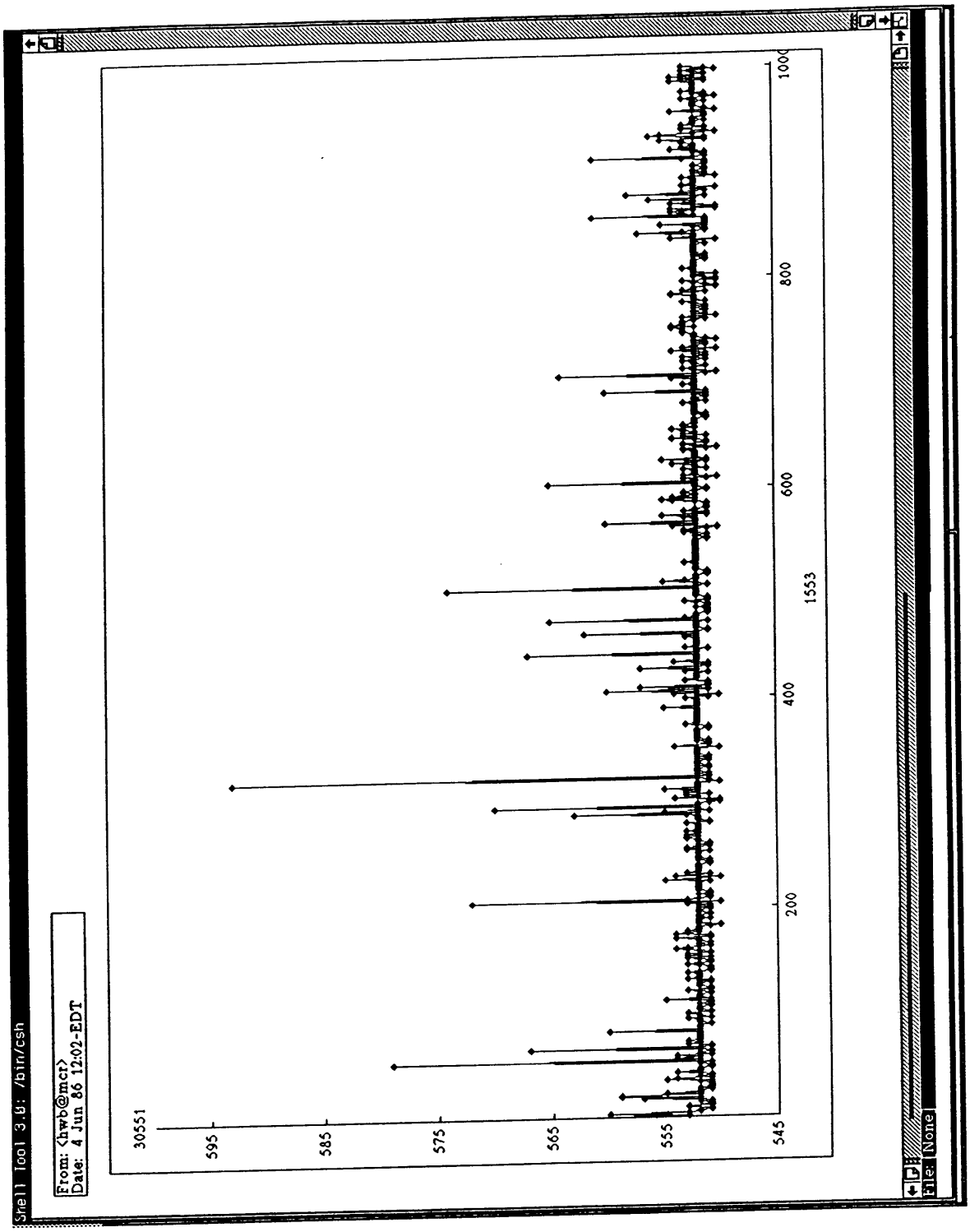

Shell Tool 3.0: /bin/csh
From: <hwb@mcr>
Date: 4 Jun 86 12:02-EDT
30551
595
585
575
565
555
545
200
400
600
800
1553
File: None

Figure (6)

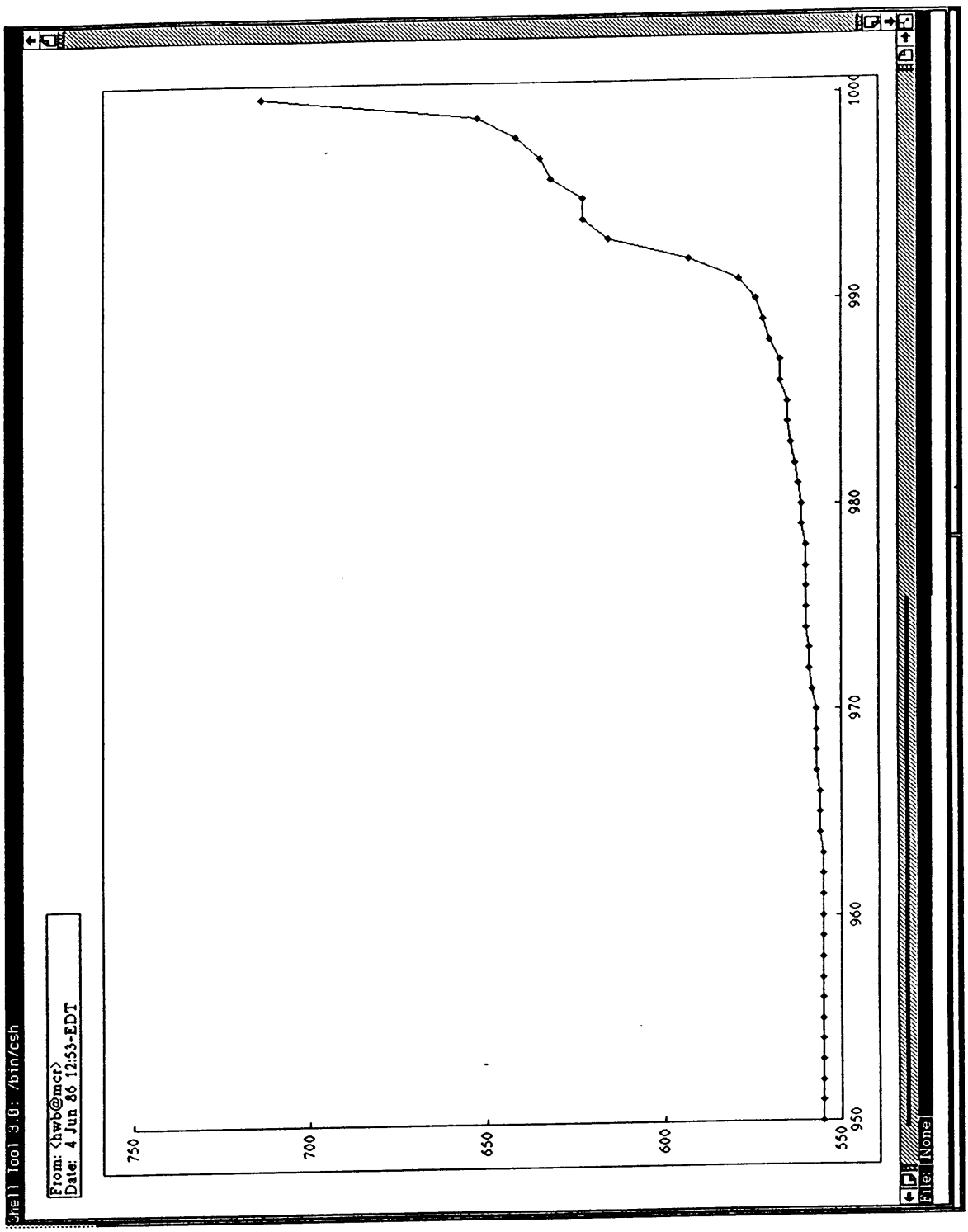

Shell tool 3.0: /bin/csh
From: <hwb@mcr>
Date: 4 Jun 86 12:53-EDT
750
700
650
600
550
950
960
970
980
990
1000
File: (None)

Figure (7)

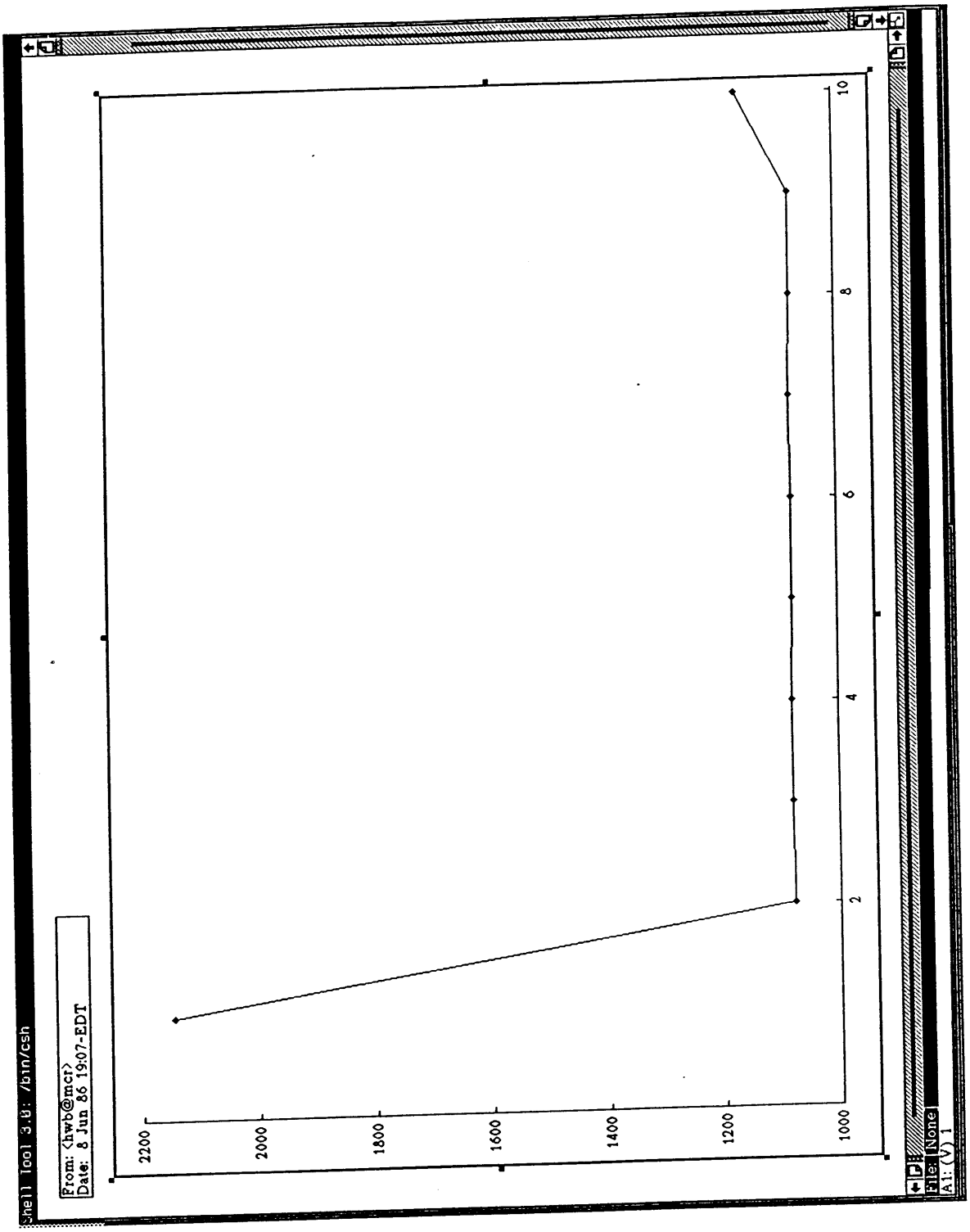

Shell tool 3.0: /bin/csh
From: <hwb@mcr>
Date: 8 Jun 86 19:07-EDT
2200
2000
1800
1600
1400
1200
1000
2
4
6
8
10
File: None
A1: (V) 1

Figure (8)

The following three diagrams (8, 9, and 10) show connectivity to the Illinois node with the Internet address 192.17.4.60, (which is logically on the same net as USAN). Connectivity from UMICH to that node requies four satellite hops for a round trip path: from Michigan to NCAR, from NCAR to Illinois, from Illinois back to NCAR, and from there back to Michigan. Here the results also were close to the expected delay times. Figure (8) shows the first ten samples. Since an Ethernet/Internet address correlation protocol gets used to determine the remote Ethernet hardware address, and since Fuzzballs drop the original buffer if these ARP protocol replies are not received within 100 msec, the first ICMP Echo request gets lost. This makes it appear as if it takes twice the time to traverse the four satellite hops. In reality the ARP reply arrives later then expected, gets latched into the ARP cache, and the higher layers in the originating machine send a new packet about a second later. The subsequent 999 packets lie in the roughly one second range, as shown in Figure (9). Figure (10) again has these 999 packets sorted by delay while showing the last 50 packets of the one thousand packet test. The reason that this chart stops around 990 is because 9 packets got lost somewhere in this setup.

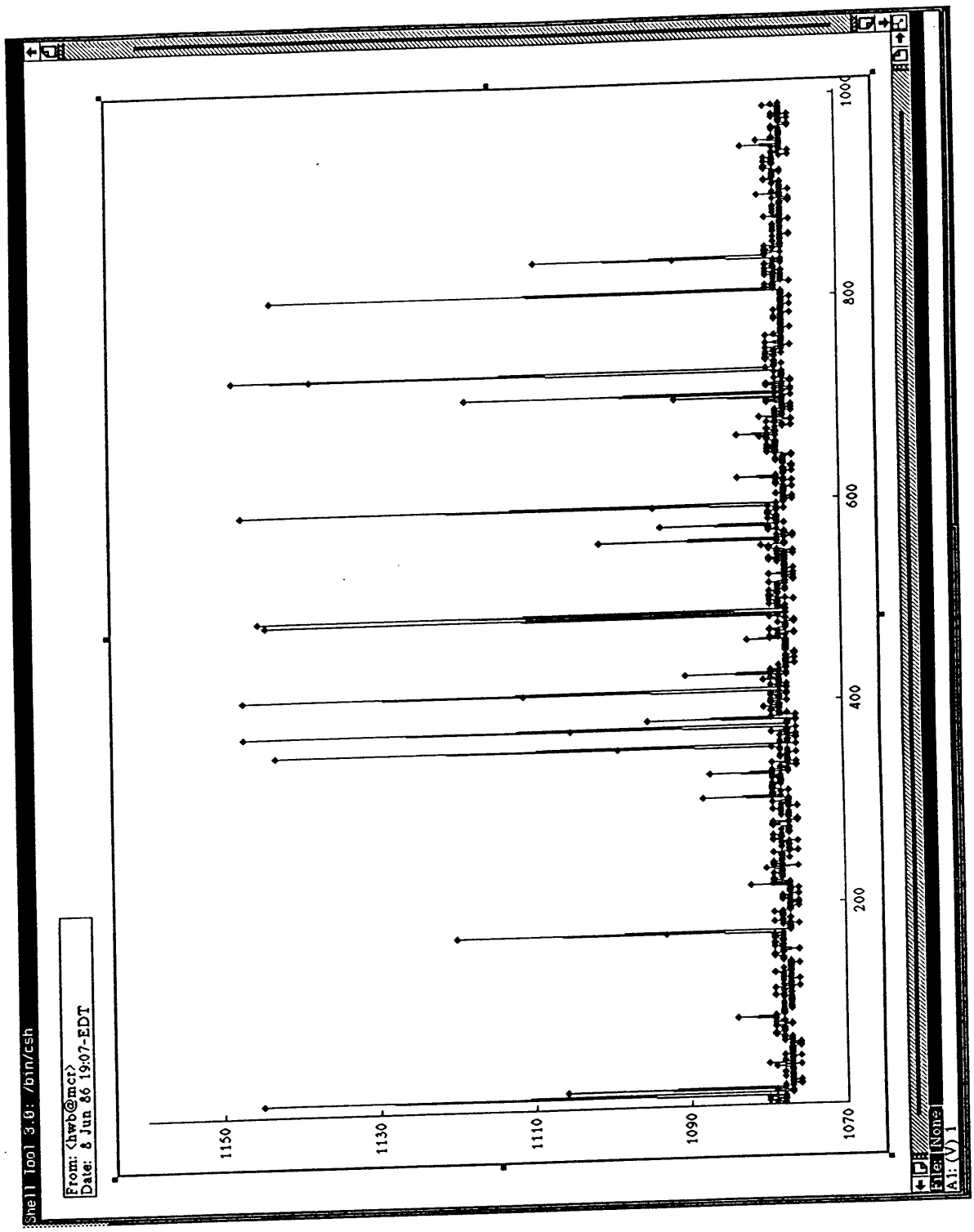

Shell Tool 3.0: /bin/csh
From: <hwb@mcr>
Date: 8 Jun 86 19:07-EDT
1150
1130
1110
1090
1070
200
400
600
800
1000
File: None
A1: (V) 1

Figure (9)

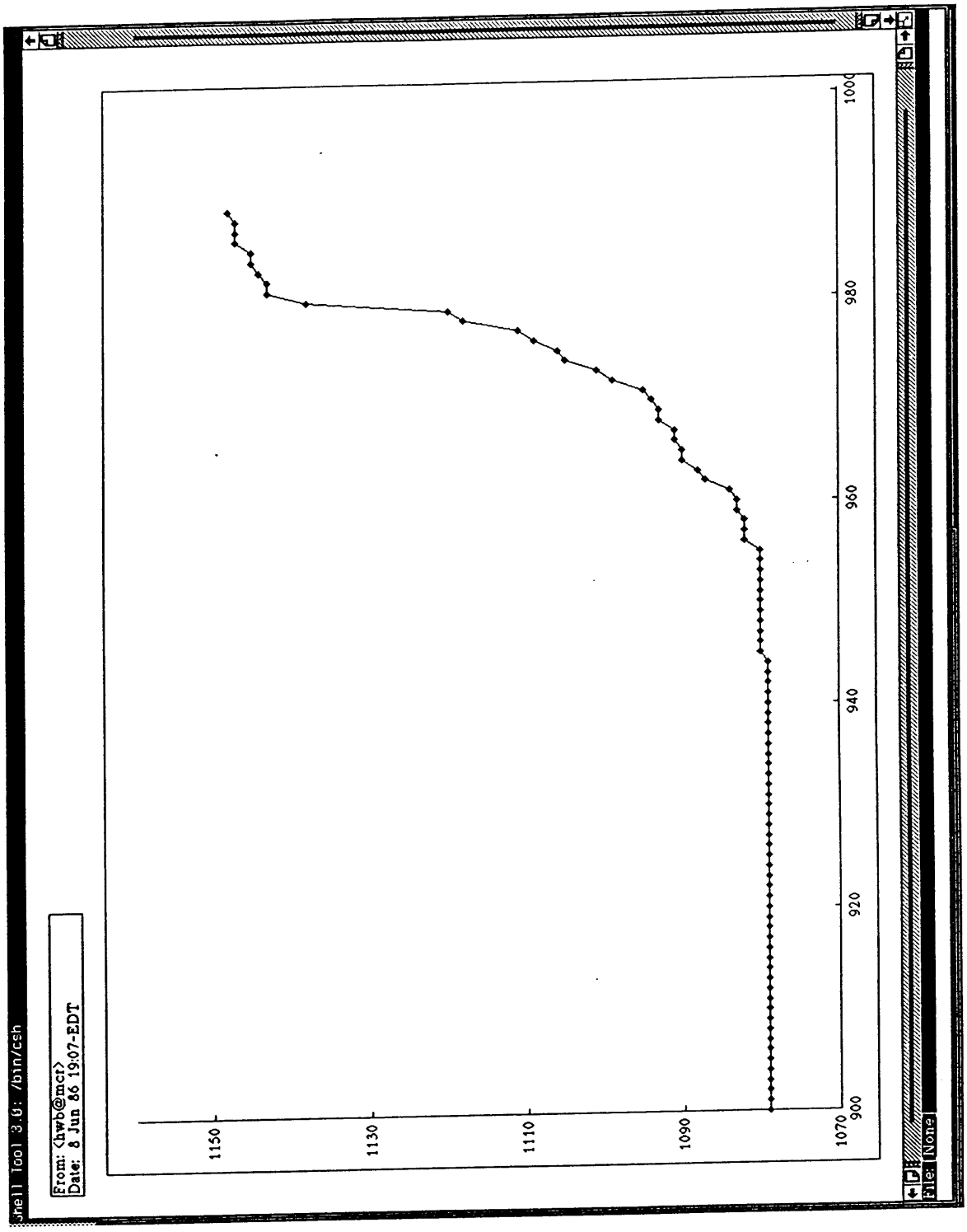

From: <hwb@mcr>
Date: 8 Jun 86 19:07-EDT
1150
1130
1110
1090
1070
900
920
940
960
980
1000
File: None

Figure (10)