Chapter 6

Routing Pathologies

We begin our analysis by classifying occurrences of routing pathologies—those routes that exhibited either clear sub-standard performance, or out-and-out broken behavior.

6.1 Unresponsive routers

Some routers do not return the required ICMP messages in response to `traceroute` probes (§ 4.2.2), or do so with insufficient TTL's to make the return trip. We refer to these as *unresponsive* routers. If these routers are prevalent, they will add a great deal of noise to our measurements, making analysis difficult. This is especially the case because an unresponsive router looks identical to a router that had to drop all three probe packets due to congestion, a case we are interested in analyzing.

Fortunately, unresponsive routers are easy to spot. Unlike congested routers, unresponsive routers *consistently* fail to answer any of the `traceroute` probe packets. Because we measured multiple `traceroutes` between sites, we can look for just such consistency.[1]

Upon inspecting the `traceroutes` in $\mathcal{R}_1$, we found 4 unresponsive routers (which between them appeared in a total of 93 `traceroutes`): the last two hops prior to the `ukc` endpoint (repaired on December 8); the last hop prior to the `lbli` endpoint (frequently, but not always); and the 8th hop from `usc` to various destinations for traffic routed between CERFNET (hop 7) and AlterNet or MCINET (hop 9), consistently. This quantity of only 4 unresponsive routers contrasts with the 751 responsive routers in the first measurement set: clearly almost all Internet routers correctly return ICMP messages for expired TTL's. Furthermore, in $\mathcal{R}_2$ we did not identify *any* unresponsive routers, in contrast with 1,095 responsive routers. The previously unresponsive routers found in the first measurement set now were responsive, indicating they had been upgraded (except we were unable to determine if those on the `usc` paths had been upgraded since `usc` did not participate in the second set of measurements).[2]

[1] Recall that we use the term "`traceroute`" to refer to both the utility, and to an instance of a measurement made using the utility.

[2] In doing this analysis for $\mathcal{R}_2$, we encountered a strange anomaly: all of the `traceroutes` from `adv` to `ustutt` were missing the hop between `icm-dc-1-h1/0-t3.icp.net` and `amsterdam1.dante.net`. But this hop consistently appeared in other `traceroutes` to `ustutt`, identifying itself as `icm-dante-e0.icp.net`. It turned out that due to an administrative decision, `icm-dante-e0.icp.net` did not have a route to `adv`'s autonomous

6.2 Rate-limiting routers

Some routers limit the rate at which they generate ICMP messages, to conserve resources (§ 4.2.3). We can partially test for the presence of such routers in our measurements as follows. Recall that, for each hop n, `traceroute` sends three "probes" to elicit ICMP messages in reply. If the hop n router limits its ICMP generation rate, then in general it will reply to the first probe (unless it happens to already have been generating ICMP messages). This reply will lead to `traceroute` rapidly sending another probe, one whose ICMP reply will then be suppressed by the router due to rate-limiting. Since `traceroute` waits up to 5 seconds between probe packets, the third probe will not arrive until 5 seconds after the second, by which time rate-limiting again allows the router to reply. So rate-limiting routers that limit ICMP generation to on the order of one per 1-2 seconds will show up in our measurements as having a high proportion of first and third replies received, but no second reply received. We term such replies "R-*-R," reflecting their pattern.

We analyzed $\mathcal{R}_2$ to determine for each router the proportion ρ of "R-*-R" replies, limiting the analysis to routers for which we had at least 5 measurements. The distribution of ρ was sharply bimodal, with 8 routers exhibiting $\rho \geq 50\%$ and the remaining 701 all having $\rho \leq 20\%$. Of the 8 routers, 7 were endpoints: `inria`, `mid`, `nrao`, `sri`, `ustutt`, `ucl`, and `wustl`. These seven are all running the Solaris operating system, which by default is configured to do rate-limiting. The other router was `cs-gw.colorado.edu`, which, according to its DNS "HINFO" record, is a Cisco 7000. These routers support rate-limiting and apparently this one had the option activated; but we conclude that, in general, routers deployed today do not rate-limit their ICMP generation, at least not on time scales of one per 1-2 seconds.

Because we subsequently only undertake light analysis of dropped `traceroute` probes (and never endpoint drops), for simplicity we assume that all missing ICMP replies correspond to either a dropped `traceroute` probe packet or a dropped reply, and not to the effects of rate-limiting.

6.3 Routing loops

Suppose router R_1's routing tables indicate that, to forward a packet to host H, it should send the packet along a path that eventually includes router R_2. If, due to an inconsistency, R_2's tables indicate it in turn should forward the packet to H via a path that eventually includes R_1, the network contains a loop. The packet will circulate between R_1 and R_2 until either its TTL expires (§ 4.2.1), never reaching H, or the loop is broken by a routing update.

In general, routing algorithms are designed to avoid loops, provided all of the routers in the network share a consistent view of the present connectivity. Thus, loops are apt to form when the network experiences a change in connectivity and that change is not immediately propagated to all of the routers [Hu95]. One hopes that loops resolve themselves quickly, as they represent a complete failure. As long as the loop persists, end-to-end communication involving the path is impossible.

Some researchers have downplayed the significance of temporary routing loops [MRR80], and the ARPANET was subject to transitory looping "at the 1% level" [Co90]. Assuming that this

system, so its replies were always lost.

means that ARPANET paths on average contained a loop 1% of the time, then from the figures presented in this section and the next we will see that loops in the Internet occur much more rarely.

Other researchers have noted that loops can rapidly lead to congestion as a router is flooded with multiple copies of each packet it forwards [ZG-LA92], and minimizing loops is a major Internet design goal [Li89]. To this end, the Border Gateway Protocol (BGP) used between autonomous systems is designed to never allow the creation of inter-AS loops [RL95, Re95], which it accomplishes by tagging all routing information with the AS path it traversed. This technique is based on the observation that routing loops occur only when the propagation of routing *information* itself is subject to loops. The tagging allows a BGP router to determine if a peer is giving it information that the peer directly or indirectly derived from the router itself. If so, the router discards the information.

In this section we analyze our measurements for the prevalence of routing loops. We classify these loops as two types, "persistent" if they lasted longer than the `traceroute` measurement, or "temporary" if they resolved within the span of the `traceroute` observing them. The next two subsections look at these two types, and the final subsection comments on the location of the loops within the network.

6.3.1 Persistent routing loops

A persistent routing loop is easy to detect in a `traceroute`. Here is an example of a loop between `lbl` and `lbli`, ordinarily 6 hops apart:

```
 1  ir6gw.lbl.gov  1.853 ms  1.623 ms  2.358 ms
 2  er1gw.lbl.gov  7.165 ms  2.996 ms  3.098 ms
 3  ir2gw.lbl.gov  4.882 ms  3.516 ms  8.371 ms
 4  isdn1gw.lbl.gov  7.98 ms  4.393 ms  4.311 ms
 5  ascend49.lbl.gov  36.833 ms  32.772 ms  31.428 ms
 6  isdn1gw.lbl.gov  30.428 ms  30.502 ms  33.528 ms
 7  ascend49.lbl.gov  69.006 ms  59.429 ms  58.82 ms
 8  isdn1gw.lbl.gov  59.358 ms  63.734 ms  61.775 ms
 9  ascend49.lbl.gov  85.629 ms  84.168 ms  83.397 ms
10  isdn1gw.lbl.gov  83.374 ms  83.201 ms  83.349 ms
11  ascend49.lbl.gov  110.316 ms  120.243 ms  116.84 ms
12  isdn1gw.lbl.gov  109.221 ms  108.97 ms  109.242 ms
13  ascend49.lbl.gov  135.867 ms  136.797 ms  140.849 ms
14  isdn1gw.lbl.gov  137.359 ms  138.757 ms  137.028 ms
15  ascend49.lbl.gov  171.109 ms  167.197 ms  168.027 ms
16  isdn1gw.lbl.gov  187.18 ms  177.017 ms  165.499 ms
17  ascend49.lbl.gov  199.461 ms  193.441 ms  201.067 ms
18  isdn1gw.lbl.gov  191.205 ms  198.674 ms  192.041 ms
19  ascend49.lbl.gov  228.833 ms  219.05 ms  240.464 ms
20  isdn1gw.lbl.gov  213.537 ms  214.975 ms  220.435 ms
21  ascend49.lbl.gov  249.681 ms  254.247 ms  243.089 ms
22  isdn1gw.lbl.gov  239.341 ms  239.072 ms  243.516 ms
23  ascend49.lbl.gov  268.134 ms  270.585 ms  267.982 ms
24  isdn1gw.lbl.gov  273.742 ms  274.974 ms  265.043 ms
25  ascend49.lbl.gov  297.033 ms  293.392 ms  294.328 ms
26  isdn1gw.lbl.gov  348.844 ms  303.868 ms  291.552 ms
```

Source	Dest.	Loop	Location
`ucol`	`bnl`	`129.19.253.18`, `129.19.253.17`	Col. State Univ.
`austr` `mit`	`umann` `umann`	`mf-0.enss145.t3.ans.net`, `umd-rt1.es.net` same	FIX-East
`lbli`	`xor`	`icm-fix-e-h2/0-t3.icp.net`, `icm-dc-2b-h3/0-t3.icp.net`	FIX-East, Washington D.C.
`lbl`	`lbli`	`isdn1gw.lbl.gov`, `ascend49.lbl.gov` (this loop occurred twice)	LBL
`lbl`	`inria`	`llnl-e-llnl2.es.net`, `llnl2-e-llnl.es.net`	Livermore, California
`sdsc`	`ukc`	`gw.ukc.ac.uk`, `gw.ulcc.ja.net`	London, Canterbury
`sdsc`	`usc`	`mobydick.cerf.net`, `drzog.cerf.net`	SDSC
`harv`	`ucl`	`mf-0.cnss56.washington-dc.t3.ans.net`, `mf-0.cnss58.washington-dc.t3.ans.net`	Washington, D.C.

Table VI: Persistent routing loops in $\mathcal{R}_1$

```
27  ascend49.lbl.gov  335.637 ms  324.15 ms  322.982 ms
28  isdn1gw.lbl.gov  328.654 ms  321.418 ms  316.452 ms
29  ascend49.lbl.gov  344.561 ms  351.843 ms  346.087 ms
30  isdn1gw.lbl.gov  358.938 ms  348.781 ms  355.01 ms
```

`isdn1gw.lbl.gov` is the Laboratory's ISDN gateway, and `ascend49.lbl.gov` is the other end of the ISDN link to `lbli`. Here, `ascend49.lbl.gov` apparently has lost track of the notion that `lbli` resides on its side of the ISDN point-to-point link, so it forwards any packets for `lbli` back to the ISDN gateway.

For our analysis, we considered any `traceroute` showing a loop that was not resolved by the end of the `traceroute` (i.e., after probing 30 hops) as a "persistent loop." Of the 6,204 `traceroutes` in $\mathcal{R}_1$,[3] 10 exhibited persistent routing loops. Table VI summarizes these.

Three of these loops appear to have formed *during* the `traceroute` probe. In the `harv` ⇒ `ucl` loop, for example, the probes made it to London and almost to the `ucl` endpoint before the loop appeared in Washington, D.C., at hop 16:

```
 1  glan-gw.harvard.edu  87 ms  3 ms  2 ms
 2  wjhgw1.harvard.edu  4 ms  2 ms  2 ms
 3  harvard-gw.near.net  8 ms  11 ms  4 ms
 4  prospect-gw.near.net  20 ms  20 ms  12 ms
 5  tang-gw.near.net  32 ms  6 ms  6 ms
 6  enss.near.net  6 ms  6 ms  3 ms
 7  t3-3.cnss48.hartford.t3.ans.net  7 ms  9 ms  11 ms
 8  t3-2.cnss32.new-york.t3.ans.net  9 ms  10 ms  10 ms
 9  t3-1.cnss56.washington-dc.t3.ans.net  18 ms  16 ms  20 ms
```

[3]This number represents the 6,459 total `traceroutes`, minus 255 `traceroutes` originating from `wustl`, which, as explained in § 6.6.2, suffered from a large degree of "fluttering," making it difficult to determine whether true routing loops were also present.

```
10  mf-0.cnss58.washington-dc.t3.ans.net  15 ms  17 ms  16 ms
11  washington2.dante.net  20 ms  15 ms  19 ms
12  icm-dc-1-e4/0.icp.net  75 ms  58 ms  77 ms
13  icm-london-1-s1-1984k.icp.net  144 ms  218 ms  127 ms
14  smds-gw.ulcc.ja.net  230 ms  161 ms  146 ms
15  smds-gw.ucl.ja.net  131 ms  155 ms  138 ms
16  cisco-pb.ucl.ac.uk 1566 ms
      * mf-0.cnss58.washington-dc.t3.ans.net 53 ms
17  mf-0.cnss56.washington-dc.t3.ans.net  58 ms  58 ms  55 ms
18  mf-0.cnss58.washington-dc.t3.ans.net  66 ms  61 ms  60 ms
19  mf-0.cnss56.washington-dc.t3.ans.net  62 ms  68 ms  68 ms
etc.
```

In the `sdsc` ⇒ `usc` loop, the loop formed just one hop from the SDSC source, after the probe had already made it from San Diego to Los Angeles:

```
 1  drzog.cerf.net  163 ms  2 ms  2 ms
 2  134.24.120.102  7 ms  8 ms  7 ms
 3  * ucla-la-smds.cerf.net  66 ms  19 ms
 4  * losnet.ucla.edu  16 ms  16 ms
 5  isi-ucla-gw.ln.net  57 ms  20 ms  18 ms
 6  * * mobydick.cerf.net  9 ms
 7  drzog.cerf.net  13 ms  9 ms  7 ms
 8  mobydick.cerf.net  9 ms  10 ms  9 ms
 9  drzog.cerf.net  10 ms  11 ms  21 ms
10  mobydick.cerf.net  13 ms  32 ms  11 ms
etc.
```

The presence of packet loss (`*`'s) prior to the loop forming at hops 6–7 may indicate connectivity deteriorating prior to a routing change (which led to an inconsistent state). A similar loss can be seen in the `harv` ⇒ `ucl` example above, at hop 16.

The `lbl` ⇒ `inria` loop entailed two separate loops:

```
 1  ir6gw.lbl.gov  1.858 ms  1.66 ms  1.546 ms
 2  er1gw.lbl.gov  3.68 ms  2.423 ms  2.244 ms
 3  lbl-lc2-1.es.net  3.252 ms  2.618 ms  2.645 ms
 4  llnl-lbl-t3.es.net  5.892 ms  4.634 ms  3.985 ms
 5  lanl-llnl-t3.es.net  34.728 ms  29.444 ms  30.195 ms
 6  snla-lanl-t3.es.net  61.712 ms  60.392 ms  60.347 ms
 7  pppl-fnal-t3.es.net  78.807 ms  79.19 ms  77.252 ms
 8  pppl-nis.es.net  79.454 ms  78.5 ms  78.166 ms
 9  umd-pppl.es.net  85.851 ms  105.744 ms  89.141 ms
10  icm-fix-e-f0.icp.net  129.442 ms  86.567 ms  88.157 ms
11  * * *
12  * * llnl-lanl-t3.es.net  321.099 ms
13  lanl-llnl-t3.es.net  577.496 ms  199.259 ms  134.383 ms
14  llnl-lanl-t3.es.net  134.854 ms  135.204 ms  134.909 ms
15  lanl-llnl-t3.es.net  160.895 ms  160.312 ms  162.187 ms
16  llnl-lanl-t3.es.net  161.882 ms  315.869 ms *
17  * * *
18  * * *
```

```
19  * * *
20  * * *
21  * * *
22  * * *
23  * * *
24  llnl2-e-llnl.es.net  17.051 ms  26.225 ms  22.082 ms
25  llnl-e-llnl2.es.net  21.823 ms  15.619 ms  21.804 ms
26  llnl2-e-llnl.es.net  16.693 ms  22.776 ms  26.126 ms
27  llnl-e-llnl2.es.net  23.758 ms  19.809 ms  22.475 ms
etc.
```

The first sign of trouble is at hop 11, where, after having made it to FIX-East in Maryland at hop 10, the network begins dropping probe packets (or their responses). At hop 12, a temporary routing loop forms between the ESNET routers in Livermore, California, and Los Alamos, New Mexico. This loop appears to lead to further problems at the end of hop 16,[4] where subsequent packets are lost for nearly 2 minutes (recall that each `*' represents a lost response, including a 5-second wait). Finally, at hop 24 the network comes back, but in an inconsistent state, with a consequent routing loop. Most likely the routing inconsistency leading to the first loop was propagated through ESNET to form the second loop.

In $\mathcal{R}_2$, 50 `traceroutes` showed persistent loops. Due to $\mathcal{R}_2$'s higher sampling frequency, for some of these loops we can place bounds on how long they persisted, by looking for surrounding measurements between the same hosts that do not show the loop. In addition, sometimes the surrounding measurements *do* show the loop—these allow us to put lower bounds on the loop's duration, too.

Table VII summarizes the loops seen in $\mathcal{R}_2$. The first two columns give the source and destination of the `traceroute`, the next column the date, and the fourth column the number of consecutive `traceroutes` that encountered the loop. The fifth and sixth columns give the routers involved in the loop and the geographic location. Note that only one of the loops spanned multiple cities (and multiple continents!), the last in the table.

The final column gives the bounds we were able to assess for the duration of the loop. Upper bounds indicate the difference in time between the two non-looping `traceroutes` bracketing the loop, if this difference was less than 1 day (otherwise the upper bound is potentially so lax that we omit it). Lower bounds, when present, indicate the difference in time between the first `traceroute` in a sequence observing the loop, and the last. For loops only observed during a single `traceroute`, this bound is omitted. Loops for which we were unable to assign any plausible bounds have their bounds marked as "?".

The loop durations appear to fall into two modes, those definitely under 3 hours (and possibly quite shorter), and those of more than half a day. The presence of persistent loops of durations on the order of hours to tens of hours is quite surprising, and suggests a lack of good tools for diagnosing network problems: neither the NOC's (Network Operation Centers) responsible for the looped routers, nor the customers, apparently discovered and repaired the loops for considerable periods of time, despite the total connectivity outage due to the loop.

We also note a tendency for persistent loops to come in clusters. Geographically, loops occurred much more often in the Washington D.C. area (MAE-East and College Park are only a

[4] So the loop persisted for about 2.5 seconds, as indicated by summing the return times for each of the probe packets.

Source	Dest.	Date	#	Loop	Location	Duration
inria	adv	Nov. 6	1	icm-dc-1-f0/0.icp.net, icm-dc-2b-f2/0.icp.net	Washington	?
inria	near	Nov. 11	1	same as above	Washington	$\leq$ 3 hr
wustl	inria	Nov. 24	1	same as above	Washington	?
inria	pubnix	Nov. 12	1	icm-dc-3-f2/0.icp.net, icm-dc-2b-f2/0.icp.net	Washington	?
inria	austr2	Nov. 15	1	same as above	Washington	?
sintef1	adv	Nov. 12	1	icm-pen-1-h1/0-t3.icp.net, icm-dc-2b-h0/0-t3.icp.net	Washington	?
pubnix	sintef1	Nov. 8	1	sl-ana-1-f0/0.sprintlink.net, sl-ana-2-f0/0.sprintlink.net	Anaheim	?
ustutt	ucl	Nov. 11	16	stuttgart1.belwue.de, stuttgart4.belwue.de	Stuttgart	16–32 hr
connix	bsdi	Nov. 14	1	sl-dc-8-h1/0-t3.sprintlink.net, sl-mae-e-h2/0-t3.sprintlink.net	MAE-East	$\geq$ 10 hr
ustutt	austr	Nov. 14	1	same as above		
pubnix	sintef1	Nov. 14	1	fddi0/0.cr1.dca1.alter.net, cisco1.washington.dc.ms.uu.net	Washington	$\leq$ 5.5 hr
austr	nrao	Nov. 15	1	cpk8-cpk-cf.sura.net, cpk9-cpk-cf.sura.net	College Park	?
many	oce	Nov. 23	12	amsterdam.nl.net, wgm01.nl.net	Amsterdam	14–17 hr
ucol	ustutt	Nov. 24	1	borderx1-hssi3-0.sanfrancisco.mci.net pacbell-nap-atm.sanfrancisco.mci.net	San Francisco	?
ucol	inria	Nov. 27	1	stamand1.renater.ft.net, stamand3.renater.ft.net	Paris	$\leq$ 14 hr
mid	bsdi	Nov. 28	1	sl-dc-6-f0/0.sprintlink.net, sl-dc-8-f0/0.sprintlink.net	Washington	$\leq$ 3 hr
mid	austr	Dec. 6	1	sl-chi-6-h3/0-t3.sprintlink.net, sl-chi-nap-h1/0-t3.sprintlink.net	Chicago	$\leq$ 3 hr
mit	wustl	Dec. 10	1	starnet2.starnet.net, starnet8.starnet.net	St. Louis	?
umann	nrao	Dec. 13	1	heidelberg1.belwue.de, heidelberg2.belwue.de	Heidelberg	?
ucl	mit	Dec. 14	1	mci-its.near.net, w91-rtr-external-fddi.mit.edu	Cambridge	$\leq$ 3 hr
near	ucla	Dec. 16	1	ln-gw.cs.ucla.edu, ucla-isi-gw.ln.net	Los Angeles	?
sri	near	Dec. 17	1*	su-a.bbnplanet.net, su-b.bbnplanet.net	Palo Alto	?
near	sri	same	1*	barrnet.sanfrancisco.mci.net, borderx1-hssi2-0.sanfrancisco.mci.net	San Francisco	?
bsdi	sintef1	Dec. 21	1	icm-pen-2-h2/0-t3.icp.net, icm-uk-1-h0/0-t3.icp.net	Pennsauken, London	$\leq$ 10 hr

Table VII: Persistent routing loops in $\mathcal{R}_2$

few miles away), perhaps because the very high degree of interchange between different network service providers in that area offers ample opportunity for introducing inconsistencies.

Loops involving separate pairs of routers also are clustered in time. The `pubnix` ⇒ `sintef1` loop, involving two AlterNet routers sited in Washington D.C., was measured at the same time between the `connix` ⇒ `bsdi` and `ustutt` ⇒ `austr` observations of a SprintLink loop, at nearby MAE-East. The `sri` ⇒ `near` and `near` ⇒ `sri` loop observations were made back-to-back. They do *not* observe the same loop, but rather two separate loops between closely related routers (the typical path from `near` to `sri` proceeds from MCINET in San Francisco immediately to BARRNET at Stanford (Palo Alto), and then at the next hop to BBN Planet at Stanford). Thus, it appears that the inconsistencies that lead to long-lived routing loops are not confined to a single pair of routers but also affect nearby routers, tending to introduce loops into their tables too. This in turn suggests that any persistent loop encountered in the network is very serious, as it may reflect a substantially larger outage than just the two looped routers initially observed.

6.3.2 Temporary routing loops

Fortunately, routing loops do not always persist for long periods of time. In addition to analyzing the `traceroute` data for persistent loops, we also looked for temporary loops. We define a temporary loop as one during which a router was visited at different hops, yet eventually the `traceroute` probe traveled beyond the loop. This definition requires manual inspection of the candidates, to remove spurious "loops" that are in reality due instead to other factors, such as "fluttering" (rapidly-variable routing; § 6.6.2) or midstream route changes (§ 6.5).

The `lbl` ⇒ `inria` example in the previous section shows both a temporary loop and a permanent loop, both involving ESNET routers. In addition to the `lbl` ⇒ `inria` example above, $\mathcal{R}_1$ exhibited one other case of a temporary routing loop, occurring between `ucl` and `wustl`:

```
 1  cisco.cs.ucl.ac.uk  12 ms  5 ms  5 ms
 2  cisco-pb.ucl.ac.uk  11 ms  4 ms  4 ms
 3  cisco-b.ucl.ac.uk  5 ms  4 ms  5 ms
 4  gw.lon.ja.net  20 ms  22 ms  19 ms
 5  eu-gw.ja.net  60 ms  21 ms  19 ms
 6  icm-lon-1.icp.net  20 ms  25 ms  37 ms
 7  icm-dc-1-s3/2-1984k.icp.net  177 ms  191 ms  168 ms
 8  * sl-dc-7-f0.sprintlink.net  1174 ms  183 ms
 9  sl-starnet-1-s0-t1.sprintlink.net  220 ms  216 ms  233 ms
10  * * *
11  * * *
12  stl2-e0.starnet.net  506 ms  775 ms  262 ms
13  stl3-e0.starnet.net  218 ms * *
14  stl2-e0.starnet.net  919 ms *  237 ms
15  * stl3-e0.starnet.net  193 ms  191 ms
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * tango.cs.wustl.edu  260 ms *
```

Here, at hops 12-15, the STARnet routers engage in a short-term routing loop that evidently is resolved during hops 16-20 (an outage of about 80 seconds).[5]

While in $\mathcal{R}_1$ we only observed two temporary loops, in $\mathcal{R}_2$ we found 23. We confine ourselves here to a look at two of the more seriously pathological, as these illustrate the degree to which routing can degrade.

The first of these was from `rain` to `inria`:

```
 1  r0.pdx.rain.rg.net  3.212 ms  2.903 ms  2.348 ms
 2  border1-serial2-5.seattle.mci.net  8.119 ms  7.509 ms  8.303 ms
 3  core-fddi-0.seattle.mci.net  10.255 ms  11.472 ms  9.087 ms
 4  core2-hssi-3.denver.mci.net  42.005 ms  45.637 ms  41.765 ms
 5  core1-aip-4.denver.mci.net  180.353 ms  210.453 ms  222.771 ms
 6  core2-hssi-2.westorange.mci.net  192.796 ms  224.263 ms  257.99 ms
 7  core2-hssi-2.washington.mci.net  96.183 ms  90.611 ms  90.897 ms
 8  borderx2-fddi-1.washington.mci.net  88.917 ms  98.286 ms  99.512 ms
 9  mae-east-plusplus-two.washington.mci.net
      95.96 ms  111.302 ms  121.937 ms
10  icm-dc-e-f0/0.icp.net  91.077 ms  102.348 ms  95.265 ms
11  * * *
12  * * *
13  * * borderx2-fddi-1.washington.mci.net  269.431 ms
14  mae-east-plusplus-two.washington.mci.net
      440.782 ms  293.266 ms  166.355 ms
15  mae-east-plusplus.washington.mci.net
      89.681 ms  94.609 ms  90.987 ms
16  borderx1-hssi2-0.washington.mci.net  91.661 ms  89.673 ms  96.562 ms
17  core2-fddi-0.washington.mci.net  137.351 ms  174.362 ms  204.639 ms
18  borderx2-fddi-1.washington.mci.net  95.169 ms  90.19 ms  94.371 ms
19  mae-east-plusplus-two.washington.mci.net
      97.839 ms  91.079 ms  97.236 ms
20  mae-east-plusplus.washington.mci.net  92.483 ms  91.213 ms  91.38 ms
21  borderx1-hssi2-0.washington.mci.net  92.318 ms  92.662 ms  95.358 ms
22  * * *
23  r0.pdx.rain.rg.net  3.343 ms !H * *
24  * t8-gw.inria.fr  779.58 ms *
25  tom.inria.fr  657.659 ms * *
```

The `traceroute` begins without any problems, traveling to ICP (the Sprint/NSF International Connectivity Project) in Washington via Seattle, Denver, West Orange (New Jersey), Washington, and MAE-East. At hop 11, however, we observe a 40 second outage. Evidently the outage was due to the loss of the link between `mae-east-plusplus-two.washington.mci.net` and `icm-dc-e-f0/0.icp.net`, because when the outage finished, we find ourselves in a routing loop between five different routers:

```
borderx2-fddi-1.washington.mci.net
mae-east-plusplus-two.washington.mci.net
mae-east-plusplus.washington.mci.net
```

[5] As discussed in § 6.6.2 below, these STARnet routers also suffered from route "fluttering," though that problem was apparently fixed on December 12, and this trace is from December 15, after the repair.

```
borderx1-hssi2-0.washington.mci.net
core2-fddi-0.washington.mci.net
```

This is one of only two times in either $\mathcal{R}_1$ or $\mathcal{R}_2$ that we observed a loop involving more than two routers. (The other is discussed in § 6.4.) The loop persists from hop 13 to hop 21 (at least). At hop 22 we suffer a 15 second outage, and when it resolves we find ourselves all the way back to where we started at hop 1. The router there has returned an "ICMP unreachable" message (the `!H`), indicating it is convinced that it cannot reach `inria`, presumably because it has lost its link to `border1-serial2-5.seattle.mci.net`. After another 15 second outage, however, we suddenly find ourselves in France, at `inria`'s doorstep: either both of the previous problems had resolved themselves, or an alternate path was discovered.

The second seriously pathological `traceroute` was from `ucol` to `umann`:

```
 1  cs-gw-srl.cs.colorado.edu  3 ms  3 ms  2 ms
 2  cu-gw-fddi.colorado.edu  5 ms  2 ms  4 ms
 3  ncar-cu.co.westnet.net  13 ms  4 ms  8 ms
 4  ml-t3-gw.ucar.edu  11 ms  24 ms  34 ms
 5  border2-hssi1-0.denver.mci.net  73 ms  141 ms  87 ms
 6  core-fddi-1.denver.mci.net  80 ms  22 ms  24 ms
 7  * core2-hssi-2.westorange.mci.net  47 ms  64 ms
 8  core2-hssi-2.washington.mci.net  58 ms  63 ms  59 ms
 9  borderx2-fddi-1.washington.mci.net  73 ms  98 ms  111 ms
10  mae-east-plusplus-two.washington.mci.net  60 ms  64 ms  60 ms
11  icm-dc-e-f0/0.icp.net  112 ms  99 ms  91 ms
12  icm-dc-1-h1/0-t3.icp.net  81 ms  94 ms  105 ms
13  icm-dante-e0.icp.net  115 ms  150 ms *
14  * amsterdam1.dante.net  205 ms *
15  nl-s1.dante.bt.net  177 ms  166 ms  151 ms
16  nl-f0-0.eurocore.bt.net  172 ms  190 ms  176 ms
17  de-s1-1.eurocore.bt.net  206 ms  247 ms  227 ms
18  de-f0.dante.bt.net  251 ms  181 ms  227 ms
19  * * *
20  * * *
21  * icm-dc-2b-f2/0.icp.net  151 ms  138 ms
22  icm-dc-1-f0/0.icp.net  97 ms  86 ms  64 ms
23  icm-dc-2b-f2/0.icp.net  98 ms  85 ms  107 ms
24  icm-dc-1-f0/0.icp.net  109 ms  92 ms umd2-pppl2.es.net  251 ms
25  * mae-east-plusplus-two.washington.mci.net  178 ms  251 ms
26  pppl2-umd2.es.net  702 ms * *
27  core-hssi-3.sanfrancisco.mci.net  158 ms !H *
    core-fddi-1.denver.mci.net  34 ms !H
```

Everything is fine up until hop 18, with the path traversing from Boulder to Denver, in Colorado; then over MCINET to West Orange and down to MAE-East, then across to Amsterdam and over to Duesseldorf—almost there! But a 35 second outage at hops 19–21 is the beginning of trouble. When the network begins responding again, we have fallen back to a temporary loop between `icm-dc-1-f0/0.icp.net` and `icm-dc-2b-f2/0.icp.net` in Washington, D.C., a position similar to that we had achieved at hops 11-12 earlier. At hop 25 we again visit `mae-east-plusplus-two.washington.mci.net`, already visited at hop 10. Note two things about this hop. First, we have now backtracked twice, once to `icm-dc-2b-f2/0.icp.net`, and then again to MAE-East, which is an earlier hop than ICM in Washington. Second, we have acquired an additional *15 hops* to our route *upstream* of MAE-East, so along with the routing loop in Washington, there is also a major change closer to `ucol`. At hop 26 we find ourselves on ESNET, but at hop 27 we initially are rerouted to San Francisco on MCINET, indicating *another* upstream change (since ESNET does not have a link from Princeton to MCI in San Francisco). This router indicates that it knows of an immediate outage by flagging the hop using `!H`. But only five seconds later we lose connectivity even to San Francisco—we are back in Denver again, as we were at hop 6, and unable to make any further progress (the router flags `!H`).

Clearly at least two different major failures occurred in this example, one the routing loop at `icm-dc-2b-f2/0.icp.net`, and the other the rapidly changing (and lengthening) path upstream from MAE-East. In the previous example, the same applies: we observed both a routing loop in Washington, and a connectivity outage between Portland and Seattle. A very interesting question is whether these failures were actually reflections of a single underlying catastrophe that propagated through the network at large.

All in all we observed 20 instances of multiple large-scale changes such as illustrated in this example, suggesting that either the propagation of a single fault's effects through the network sometimes leads to widespread, temporary instability, or that a mechanism separate from the exchange of routing information is producing widespread faults. Determining which of these is the case and how the fault propagates would make for interesting future work.

6.3.3 Location of routing loops

We analyzed the routers involved in temporary and persistent loops to see whether any of the loops involved more than one AS. As mentioned above, the design of BGP in theory prevents any inter-AS loops, by preventing any looping of routing information. We found that only three of the $\mathcal{R}_1$ loops spanned more than one AS, and only two of those in $\mathcal{R}_2$. We also learned that at least one of the inter-AS loops in $\mathcal{R}_2$ occurred due to the presence of a static route, and thus clearly was not the fault of BGP. It may be that the others have similar explanations. In any event, it appears clear from our data that BGP loop suppression virtually eliminates inter-AS looping.

6.4 Erroneous routing

A final example of a routing loop occurred during a `connix` $\Rightarrow$ `ucl traceroute`, which also exhibits *erroneous* routing, where the packets clearly took the wrong path:

```
 1  mfd-01.rt.connix.net  8 ms  4 ms  3 ms
 2  sl-dc-5-s2/0-512k.sprintlink.net  39 ms  39 ms  39 ms
 3  sl-dc-6-f0/0.sprintlink.net  39 ms  38 ms  50 ms
 4  psi-mae-east-1.psi.net  48 ms  66 ms *
 5  * * core.net218.psi.net  90 ms
 6  192.91.187.2  1139 ms  1188 ms *
 7  * * *
 8  biu-tau.ac.il  1389 ms * *
 9  tau.man.ac.il  1019 ms * *
10  * * *
```

```
11  * cisco301s1.huji.ac.il  1976 ms *
12  * * *
13  * * *
14  * * cisco101e5.huji.ac.il  1974 ms
15  * * *
16  * cisco103e2.gr.huji.ac.il  1010 ms  1069 ms
17  cisco101e01.cc.huji.ac.il  2132 ms * *
18  cisco102e13.huji.ac.il  888 ms  976 ms  2005 ms
19  cisco103e2.gr.huji.ac.il  1657 ms * *
20  * * cisco101e01.cc.huji.ac.il  1349 ms
21  * * *
etc.
```

Recall that `connix` is sited in Middlefield, Connecticut, and `ucl` in London, England. Yet at hop 6, instead of routing towards London, the route winds up visiting `192.91.187.2` as the next hop—`192.91.187.2` is sited at the Weizmann Institute in Rehovot, Israel! (As can be seen by the long latency to hop 6, a satellite link is involved here.) Not surprisingly, the bewildered Israeli routers do not really know what to make of the London-bound packet: it enters a routing loop between `cisco101e01.cc.huji.ac.il`, `cisco102e13.huji.ac.il`, and `cisco103e2.gr.huji.ac.il` prior to being discarded. The lack of any response to `traceroute` probes beyond hop 20 may be due to the route being terminated further upstream, or because growing congestion on the US–Israel link led to subsequent probes getting dropped.

There is a security lesson to be considered here, too: one really cannot make any safe assumptions about where one's packets might travel on the Internet. If the Israeli routers had an alternate path to London available to them, it is possible that this highly circuitous route would have succeeded (cf. § 6.9).

6.5 Connectivity altered mid-stream

In 10 of the $\mathcal{R}_1$ traces we observed routing connectivity reported earlier in the `traceroute` later lost or altered, indicating we were catching a routing failure as it happened:

```
 1  netlab1-gw.usc.edu  3 ms  3 ms  3 ms
 2  rtr1.usc.edu  3 ms  2 ms  2 ms
 3  isi-usc-gw.ln.net  5 ms  4 ms  5 ms
 4  ucla-isi-gw.ln.net  121 ms  230 ms *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * rtr1.usc.edu  2 ms !H *
10  * * *
11  rtr1.usc.edu  2 ms !H * *
12  * * *
13  rtr1.usc.edu  2 ms !H *  2 ms !H
```

In this trace from `usc` to `ucol`, by hop 4 the packets have made it from `usc` out to the UCLA/ISI Los Nettos gateway. The large round-trip times reported at hop 4 indicate trouble, however,[6] and after the second hop 4 reply, connectivity is lost for about 70 seconds. When it returns, connectivity is only present to the hop 2 router, which reports that the destination host is unreachable (the "`!H`" flag). Because the recovery only extends to the 2nd hop, we infer that the problem occurred not at the hop 4 router but rather at hop 3, the gateway between USC and ISI.

In the other traces, a connectivity loss was followed by a recovery, as shown in this `traceroute` between `bnl` and `usc`:

```
 1  cerberus.90.bnl.gov  2 ms  2 ms  2 ms
 2  nioh.bnl.gov  3 ms  2 ms  4 ms
 3  192.12.15.224  3 ms  2 ms  2 ms
 4  pppl-bnl.es.net  11 ms  11 ms  14 ms
 5  * * *
 6  * 192.12.15.224  4 ms !H *
 7  * 192.12.15.224  3 ms !H *
 8  * 192.12.15.224  5 ms !H *
 9  * * *
10  * * *
11  * 192.12.15.224  4 ms !H *
12  * 192.12.15.224  84 ms !H *
13  * * *
14  * usc-cit-gw.ln.net  563 ms  257 ms
15  rtr5.usc.edu  283 ms  317 ms  242 ms
16  catarina.usc.edu  282 ms  102 ms  211 ms
17  escondido.usc.edu  199 ms  306 ms  392 ms
```

Router `192.12.15.224` is located at the `bnl` site. At hop 5, it clearly loses its link to `pppl-bnl.es.net`, and the link does not return for two minutes. Once it does, the `traceroute` probes are able to continue all the way to `usc`.

Three additional $\mathcal{R}_1$ traces revealed similar high-delay recoveries, incurring outages ranging from about 1 minute to almost 5 minutes. One striking example is from `wustl` to `ucol`:

```
 1  jcr-166.cs.wustl.edu  5 ms  2 ms  2 ms
 2  ncrc-eng.wustl.edu  3 ms  2 ms  2 ms
 3  128.252.5.120  3 ms  3 ms  2 ms
 4  128.252.1.2  4 ms  4 ms  3 ms
 5  sl-dc-7-s7-t1.sprintlink.net  30 ms  28 ms  28 ms
 6  sl-dc-6-f0/0.sprintlink.net  81 ms  27 ms  33 ms
 7  sl-dc-8-f0/0.sprintlink.net  106 ms  37 ms  30 ms
 8  * * *
 9  * * sl-dc-8-f0/0.sprintlink.net  32 ms !H
10  * * *
11  * * *
12  * * *
13  * * *
```

[6] Between `usc` and `ucol` this hop usually had a latency of 5-10 msec. We did not, however, undertake any rigorous evaluation of hop latencies, because of the potentially large noise associated with these times, as discussed in § 4.2.2, and as illustrated above.

```
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  * * *
24  * * *
25  clark.cs.colorado.edu  128 ms  106 ms  105 ms
```

Here, connectivity was lost for between 15-17 hops. At first it might appear from this `traceroute` that the route upon recovery consisted of 25 hops, but that is instead a measurement artifact: by the time the network had recovered, the `traceroute` hop-count had ratcheted so high that the first successful probes following the outage made it all the way to the `ucol` endpoint. They no doubt would also have done so if they had been transmitted with somewhat lower TTL's.

Two other traces revealed different, quite quick recovery behavior:

```
 1  netlab1-gw.usc.edu  3 ms  3 ms  3 ms
 2  rtr1.usc.edu  4 ms  3 ms  3 ms
 3  cit-usc-gw.ln.net  8 ms  3 ms  4 ms
 4  cerfnet-cit-gw.ln.net  17 ms  23 ms  6 ms
 5  sdsc-cit.cerf.net  84 ms  39 ms  21 ms
 6  mobydick.cerf.net  30 ms  37 ms  35 ms
 7  ucop-sdsc-2.cerf.net  85 ms  43 ms  50 ms
 8  sl-ana-3-s2/6-t1.sprintlink.net  68 ms  86 ms  84 ms
 9  sl-ana-1-f0/0.sprintlink.net  94 ms  72 ms  53 ms
10  sl-fw-6-h2/0-t3.sprintlink.net  100 ms  99 ms  62 ms
11  sl-fw-2-f0.sprintlink.net  120 ms  130 ms  132 ms
12  sl-colorado-1-s0-t1.sprintlink.net  146 ms  151 ms  172 ms
13  * t3-0.cnss56.washington-dc.t3.ans.net  121 ms  140 ms
14  t3-0.enss145.t3.ans.net  132 ms  127 ms  120 ms
15  icm-fix-e-f0.icp.net  155 ms  129 ms  306 ms
16  icm-dc-2b-h3/0-t3.icp.net  370 ms  137 ms  148 ms
17  sl-dc-8-f0/0.sprintlink.net  127 ms  144 ms  145 ms
18  * sl-fw-5-h4/0-t3.sprintlink.net  334 ms  211 ms
19  sl-fw-2-f0.sprintlink.net  156 ms  183 ms  157 ms
20  sl-colorado-1-s0-t1.sprintlink.net  202 ms *  199 ms
21  gw2.boulder.co.coop.net  179 ms  193 ms  189 ms
22  bandicoot.xor.com  237 ms  199 ms  210 ms
```

The path here is from `usc` to `xor`. It looks fairly straight-forward, suffering only three isolated losses, but observe that hop 11 and hop 19 are identical! (As are hops 12 and 20.) The `sl-colorado-1-s0-t1.sprintlink.net` router is only two hops from the destination, `bandicoot.xor.com`, so apparently this `traceroute` was on the verge of reaching its destination at hop 14 (and indeed two of the other `usc` ⇒ `xor` `traceroutes` took only 14 hops) when a routing change occurred upstream, forcing the packets to detour all the way to the East coast of the U.S. on their trip from California to Colorado. In contrast to the examples in the previous section, in this case the routing change occurred quite smoothly, with only a single packet loss at hop 13 indicating a 5-second outage during the switch-over.

By inspecting other `usc` routes involving `t3-0.cnss56.washington-dc.t3.ans.net` at hop 13, we conclude that the change occurred at hop 10, where instead of routing from Anaheim, California to Fort Worth, Texas, as shown above, and staying inside Sprintlink, the switch was made to route to Houston, Texas, using ANS.

Another example, a `ucl` ⇒ `wustl` `traceroute`, is even more striking:

```
 1  cisco.cs.ucl.ac.uk  13 ms  5 ms  5 ms
 2  cisco-pb.ucl.ac.uk  14 ms  4 ms  4 ms
 3  cisco-b.ucl.ac.uk  5 ms  4 ms  4 ms
 4  gw.lon.ja.net  48 ms  36 ms  81 ms
 5  eu-gw.ja.net  71 ms  58 ms  72 ms
 6  icm-lon-1.icp.net  56 ms  120 ms  119 ms
 7  icm-dc-1-s3/2-1984k.icp.net  162 ms  137 ms  175 ms
 8  sl-dc-7-f0.sprintlink.net  160 ms  197 ms  189 ms
 9  sl-starnet-1-s0-t1.sprintlink.net  166 ms  122 ms  634 ms
10  ncrc-acn.wustl.edu  457 ms  127 ms  119 ms
11  ncrc-eng.wustl.edu  140 ms  237 ms  174 ms
12  cisco-b.ucl.ac.uk  488 ms !H jcr.ecl.wustl.edu  244 ms  232 ms
13  tango.cs.wustl.edu  228 ms *  151 ms
```

Note that the first hop 12 router, `cisco-b.ucl.ac.uk`, is the same as the hop 3 router! This router also reports "`!H`", indicating it could not forward the packet, and yet the second and third `traceroute` probe packets for that hop make it all the way to `wustl`. This `traceroute` appears to reflect a 500 msec outage, quickly repaired.

We thus see that the distribution of recovery times from routing problems is at least bimodal—some recoveries occur quite quickly, on the time scale of congestion delays, while others take on the order of a minute to resolve. The latter type of recovery presents significant difficulties to time-sensitive applications that assume outages are short-lived.

Sometimes the presence of a connectivity change is more subtle, such as in this $\mathcal{R}_1$ `traceroute` from `korea` to `ucol`:

```
 1  fpls.postech.ac.kr  2 ms  1 ms  1 ms
 2  fddicc.postech.ac.kr  3 ms  2 ms  2 ms
 3  ktrc-postech.hana.nm.kr  30 ms  30 ms  51 ms
 4  gateway.hana.nm.kr  31 ms  31 ms  31 ms
 5  hana.hana.nm.kr  33 ms  44 ms  32 ms
 6  bloodyrouter.hawaii.net  1152 ms  1275 ms  968 ms
 7  bloodyrouter.hawaii.net  744 ms  336 ms  325 ms
 8  arc1.nsn.nasa.gov  384 ms  491 ms  691 ms
 9  jpl6.nsn.nasa.gov  791 ms  772 ms  1082 ms
10  jpl3.nsn.nasa.gov  876 ms *  1641 ms
11  ncar1.nsn.nasa.gov  1117 ms  1225 ms  848 ms
12  * cu-gw.ucar.edu  1280 ms  805 ms
13  cu-ncar.co.westnet.net  774 ms  884 ms *
14  cs-gw.colorado.edu  1079 ms  897 ms  603 ms
15  lewis.cs.colorado.edu  283 ms  383 ms  899 ms
```

In this example, hop 6 and hop 7 were both to `bloodyrouter.hawaii.net`.[7] The subsequent route shown above is exactly the route taken by every other `korea` $\Rightarrow$ `ucol traceroute`, except each hop is delayed by one (e.g., `jpl6.nsn.nasa.gov` is hop 9 here instead of hop 8 as usual).

Duplicate hops such as this one are most likely due to upstream route changes (§ 4.2.3) which, in this example, added an extra hop upstream to `bloodyrouter.hawaii.net`. The change would have had to occur just between the end of the probes for hop 6 and the beginning of those for hop 7. We considered all such duplicated hops to be midstream route changes.

In contrast with the rarity of connectivity changes in $\mathcal{R}_1$ (10 total), in $\mathcal{R}_2$ we observed 155 instances of a change, a fact we comment upon further in § 6.10.

6.6 Fluttering

We use the term "fluttering" to refer to rapidly-variable routing. On the time scale of a single `traceroute` (seconds to minutes) we would expect the path we are measuring to remain stable, yet surprisingly often our data showed that the packets belonging to a single `traceroute` took multiple paths through the Internet.

6.6.1 A simple example

Route fluttering can be detected from `traceroute` output by the presence of more than one host listed for a single hop, as in this example of a $\mathcal{R}_1$ `traceroute` between `korea` and `austr`.

```
 1  fpls.postech.ac.kr  2 ms  2 ms  2 ms
 2  fddicc.postech.ac.kr  3 ms  2 ms  2 ms
 3  ktrc-postech.hana.nm.kr  57 ms  123 ms  30 ms
 4  gateway.hana.nm.kr  31 ms  31 ms  31 ms
 5  hana.hana.nm.kr  33 ms  140 ms  32 ms
 6  bloodyrouter.hawaii.net  825 ms  722 ms  805 ms
 7  usa-serial.gw.au  960 ms  922 ms  893 ms
 8  national-aix-us.gw.au  1039 ms * *
 9  * rb1.rtr.unimelb.edu.au  903 ms rb2.rtr.unimelb.edu.au  1279 ms
10  itee.rtr.unimelb.edu.au  1067 ms  1097 ms  872 ms
11  * * mulkirri.cs.mu.oz.au  1468 ms
12  mullala.cs.mu.oz.au  1042 ms  1140 ms  1262 ms
```

Here, the 9th hop shows two different hosts (as well as no reply for the first `traceroute` packet), `rb1.rtr.unimelb.edu.au` and `rb2.rtr.unimelb.edu.au`. Thus, it appears that for the second packet `national-aix-us.gw.au` routed the packet to `rb1.rtr.unimelb.edu.au`, and for the third packet to `rb2.rtr.unimelb.edu.au`. (This change occurred most likely for purposes of load-balancing—see § 6.6.2 and § 7.4.)

It is important to keep in mind, though, that the actual route flutter could have occurred *upstream* from `national-aix-us.gw.au`, and that for the hop 9 `traceroute` packets, the 8th hop was actually a different router altogether (§ 4.2.3).

[7] In the example we have shown hostnames rather than IP addresses, as this aids in placing the router's location and service provider. It is possible for two different IP addresses to translate to the same hostname (indeed this is very common for routers). But inspecting the raw `traceroute` reveals the same IP address for both hop 6 and hop 7.

For subsequent hops, we cannot tell which of `rb1.rtr.unimelb.edu.au` or `rb2.rtr.unimelb.edu.au` was used (indeed, it could have been all of one or the other, or a continuation of switching between the two, or still a third router; the path was consistent with others we observed from the two routers).

6.6.2 A more dramatic example

The preceding example is straight-forward and demonstrates only minor fluttering, which presumably has no significant effect on the characteristics of the Internet path between `korea` and `austr`. A more dramatic example comes from a $\mathcal{R}_1$ `traceroute` between `wustl` and `umann`:

```
 1  128.252.166.249  11 ms  29 ms  8 ms
 2  128.252.123.254  3 ms  2 ms  2 ms
 3  128.252.5.120  3 ms  3 ms  14 ms
 4  128.252.1.135  6 ms  3 ms  3 ms
 5  199.217.253.1  19 ms  35 ms 199.217.253.3  64 ms
 6  144.228.73.17  56 ms 144.228.27.5  26 ms  28 ms
 7  144.228.20.101  29 ms  38 ms 144.228.70.2  55 ms
 8  144.228.10.25  69 ms  65 ms 192.157.65.74  57 ms
 9  144.228.8.233  217 ms  117 ms 194.41.0.17  118 ms
10  144.228.10.22  107 ms 193.172.4.8  122 ms  114 ms
11  192.203.230.253  68 ms 193.172.4.12  130 ms 192.203.230.253  70 ms
12  193.174.74.94  194 ms 140.222.8.4  72 ms 193.174.74.94  192 ms
13  193.174.74.29  192 ms  189 ms  192 ms
14  140.222.112.2  108 ms 129.143.6.16  222 ms  216 ms
15  140.222.64.1  128 ms 153.17.62.105  236 ms 140.222.64.1  141 ms
16  129.143.61.2  238 ms  284 ms 140.222.104.2  162 ms
17  134.155.48.125  242 ms 140.222.72.1  164 ms 134.155.48.125  263 ms
```

Here we show the route using untranslated IP addresses, since showing the names of all of the various routers would make for messy reading. However, consider hop 10:

```
10  icm-fix-w-h2/0-t3.icp.net 107 ms amsterdam6.empb.net 122 ms 114 ms
```

The first packet visited FIX-West at NASA AMES Research Center (Moffett Field, San Francisco Bay Area), while the second and third made it to Amsterdam!

The divergence begins at hops 4-5:

```
 4  128.252.1.135  6 ms  3 ms  3 ms
 5  stl1-e0.starnet.net  19 ms  35 ms stl3-e0.starnet.net  64 ms
```

The WUSTL border router (`128.252.1.135`) picks two different STARnet routers for the next hop, each of which presumably has a different notion of the best path to Europe. The confused `traceroute` shown above can be reduced to two separate `traceroutes` at this split. First, the "successful" path—the one that first reaches `umann`:

```
 5  ?
 6  sl-dc-7-s7-t1.sprintlink.net
 7  icm-dc-1-f0/0.icp.net
 8  icm-dante-e0.icp.net
```

```
 9  amsterdam1.dante.net
10  amsterdam6.empb.net
11  duesseldorf2.empb.net
12  ipgate2.win-ip.dfn.de
13  duesseldorf2.win-ip.dfn.de
14  heidelberg1.belwue.de
15  mannheim.belwue.de
16  belwue-gw.uni-mannheim.de
17  eratosthenes.informatik.uni-mannheim.de
```

Geographically, this route traverses: St. Louis, Missouri; Washington, D.C.; Amsterdam, the Netherlands; and Duesseldorf, Heidelberg, and Mannheim, in Germany.[8]

The second route instead criss-crosses the United States:

```
 5  ?
 6  sl-ana-3-s3/1-t1.sprintlink.net
 7  sl-ana-2-f0/0.sprintlink.net
 8  sl-stk-6-h2/0-t3.sprintlink.net
 9  144.228.8.233
10  icm-fix-w-h2/0-t3.icp.net
11  t3-0.enss144.t3.nsf.net
12  t3-3.cnss8.san-francisco.t3.ans.net
13  ?
14  t3-1.cnss112.albuquerque.t3.ans.net
15  t3-0.cnss64.houston.t3.ans.net
16  t3-1.cnss104.atlanta.t3.ans.net
17  t3-0.cnss72.greensboro.t3.ans.net
```

Geographically, this route traverses: St. Louis, Missouri; Anaheim, Stockton, FIX-West, and San Francisco, California; Albuquerque, New Mexico; Houston, Texas; Atlanta, Georgia; and Greensboro, North Carolina.[9] From other `traceroutes` that included `t3-0.cnss72.greensboro.t3.ans.net`, we can determine that eventually this route would also have made it to the destination, albeit with many more hops. For example, from a trace from `sri` to `umann`, we have:

```
12  t3-0.cnss72.greensboro.t3.ans.net
13  t3-0.cnss56.washington-dc.t3.ans.net
14  t3-0.enss145.t3.ans.net
15  umd-rt1.es.net
16  umd2-e-stub.es.net
17  pppl2-umd2.es.net
18  ipgate2.win-ip.dfn.de
```

[8]Hop 5 is marked as "?" because from the trace it is not clear which of the two STARnet routers picks this route (by forwarding to `sl-dc-7-s7-t1.sprintlink.net`), and which picks the longer route.

[9]Hop 13 is missing because, in the raw trace, all three replies to the hop 13 `traceroute` probe were returned by `duesseldorf2.win-ip.dfn.de`, which clearly is not the next hop following `t3-3.cnss8.san-francisco.t3.ans.net`, but rather represents hop 13 from the first route.

By inspecting other `traceroutes` from `wustl` to `umann`, it is evident that hop 13 for the second route is `t3-0.cnss16.los-angeles.t3.ans.net`, so we can add Los Angeles to the list of California cities traversed by the route.

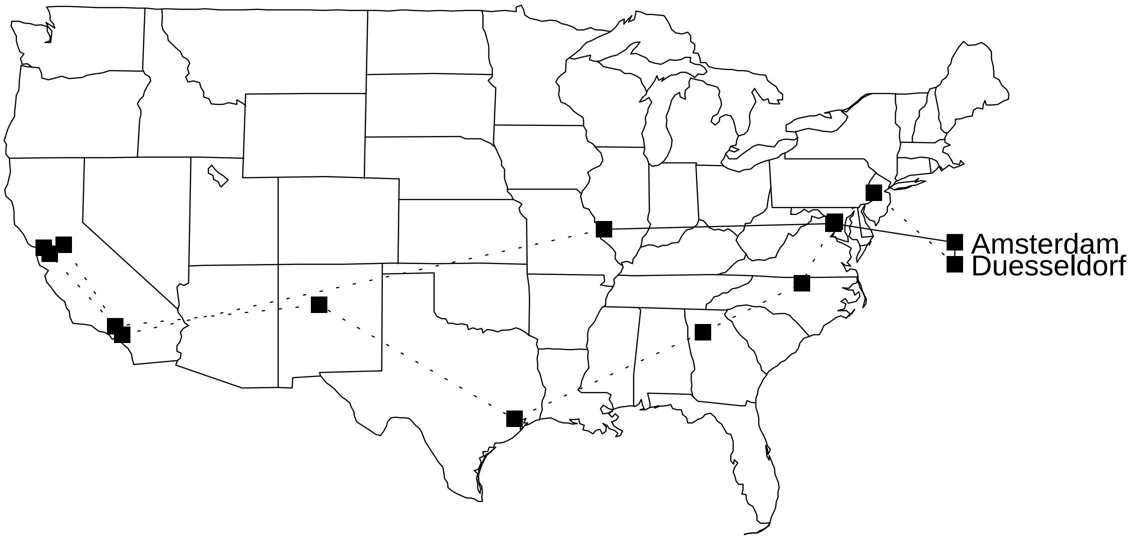

Figure 6.1: Routes taken by alternating packets from `wustl` (St. Louis, Missouri) to `umann` (Mannheim, Germany), due to fluttering

```
19  ipgate2.win-ip.dfn.de
20  duesseldorf2.win-ip.dfn.de
21  heidelberg1.belwue.de
22  mannheim.belwue.de
23  belwue-gw.uni-mannheim.de
24  eratosthenes.informatik.uni-mannheim.de
```

Thus, it appears that the second `wustl` ⇒ `umann` route would also succeed in delivering packets, though using 29 hops instead of 17.

The `wustl` fluttering occurs over very small timescales, essentially the time between successive `traceroute` probes, which are spaced out by the amount of time it takes for each reply to the previous probe packet (§ 4.2.2). One routing mechanism that can lead to such small-scale fluttering occurs when a router alternates between multiple next-hop routers in order to split load among the links to those routers. Such behavior is explicitly allowed in [Ba95, p.79], though that document also cautions that there are situations for which it is inappropriate, and so it should at most be a configurable option for a router. It turns out that the `wustl` fluttering was indeed due to load-splitting: STARnet had two T1 links for its access to Sprintlink, one to Anaheim and the other to Washington, D.C. (as shown above), and would alternate packets "round-robin" between them in order to balance load [My95].

Figure 6.1 shows the two routes that packets can take from `wustl` to `umann`. The dramatic difference in the lengths of the two routes highlights the great impact an early routing discrepancy can make.

Of the 380 `traceroutes` initiated by `wustl`, 255 exhibited fluttering, all but one occurring before 12PM PST, December 13. After this point, the Anaheim link apparently became unavailable, and the routing was no longer split. This change however was not due to a decision to eliminate fluttering, but, apparently, simply due to an outage along the Anaheim link. On December 20 the Anaheim link again became operational, and led to an interesting pathology:

```
 1  128.252.166.249  4 ms  2 ms  3 ms
 2  128.252.123.254  3 ms  2 ms  4 ms
 3  128.252.5.120  3 ms  2 ms  2 ms
 4  128.252.1.2  5 ms  3 ms  3 ms
 5  199.217.253.2  4 ms  3 ms  4 ms
 6  199.217.253.1  4 ms  11 ms 199.217.253.3  6 ms
 7  199.217.253.2  4 ms 144.228.73.17  58 ms  56 ms
 8  144.228.70.1  56 ms 199.217.253.3  4 ms  5 ms
 9  144.228.10.29  85 ms 144.228.73.17  74 ms  63 ms
10  144.228.30.5  102 ms  217 ms  218 ms
11  144.228.10.29  81 ms 144.228.10.17  93 ms  92 ms
12  144.228.20.6  84 ms  131 ms  125 ms
13  192.157.65.227  85 ms 144.228.10.29  80 ms 192.157.65.227  81 ms
14  144.228.20.6  137 ms 144.228.30.5  264 ms 144.228.20.6  165 ms
15  144.228.10.17  70 ms * *
16  144.228.30.5  90 ms * 144.228.20.6  74 ms
17  * 192.157.65.227  105 ms *
18  137.39.128.7  120 ms * *
19  * 192.157.65.227  84 ms *
20  * * *
21  * * *
22  * * 137.39.128.7  202 ms
23  * * *
24  * * *
25  * * *
26  * * *
27  * * *
28  * * *
29  * * *
```

The fluttering begins at hop 6:

```
 5  stl2-e0.starnet.net  4 ms  3 ms  4 ms
 6  stl1-e0.starnet.net  4 ms  11 ms stl3-e0.starnet.net  6 ms
```

Here, packets again alternate between `stl1-e0.starnet.net` and `stl3-e0.starnet.net`. Hop 7, though, shows that the routing is further confused:

```
 7  stl2-e0.starnet.net 4 ms sl-ana-3-s3/1-t1.sprintlink.net 58 ms 56 ms
```

It appears that either `stl1-e0.starnet.net` or `stl3-e0.starnet.net` forwarded the packet *back* to `stl2-e0.starnet.net`, while the other forwarded the packet to `sl-ana-3-s3/1 t1.sprintlink.net` in Anaheim, California. In the next hop:

```
 8  sl-ana-1-f0/0.sprintlink.net  56 ms stl3-e0.starnet.net  4 ms  5 ms
```

one of the packets makes it to the next Anaheim hop, while the other is forwarded (apparently from `stl2-e0.starnet.net`) to `stl3-e0.starnet.net`.

At this point, the packets proceed to `bsdi` but with some making one (or even more!) visits from `stl2-e0.starnet.net` to the non-forwarding STARnet router (it is difficult to determine whether this is `stl1-e0.starnet.net` or `stl3-e0.starnet.net`). Viewed geographically:

```
 9  Fort-Worth-6  85 ms Anaheim  74 ms  63 ms
10  Fort-Worth-5  102 ms  217 ms  218 ms
11  Fort-Worth-6  81 ms Washington-DC-8  93 ms  92 ms
12  Washington-DC-6  84 ms  131 ms  125 ms
13  Boone-VA  85 ms Fort-Worth-6  80 ms Boone-VA  81 ms
14  Washington-DC-6  137 ms Fort-Worth-5  264 ms Washington-DC-6  165 ms
15  Washington-DC-8  70 ms * *
16  Fort-Worth-5  90 ms * Washington-DC-6  74 ms
17  * Boone-VA  105 ms *
18  Dallas  120 ms * *
19  * Boone-VA  84 ms *
```

The `Fort-Worth-5` router at both hop 10 and hop 16 indicates that one of the hop 16 packets made *three* trips to the non-forwarding STARnet router prior to getting forwarded to the working router. Most likely this pathology occurred due to a set of inconsistent routing tables introduced by the reactivation of the Anaheim link.

For reference, a flutter-free route from `wustl` to `bsdi` is:

```
 1  jcr-166.cs.wustl.edu  5 ms  2 ms  2 ms
 2  ncrc-eng.wustl.edu  3 ms  2 ms  2 ms
 3  128.252.5.120  4 ms  3 ms  2 ms
 4  128.252.1.2  6 ms  6 ms  3 ms
 5  sl-dc-7-s7-t1.sprintlink.net  29 ms  28 ms  25 ms
 6  sl-dc-6-f0/0.sprintlink.net  156 ms  26 ms  64 ms
 7  boone1.va.alter.net  30 ms  35 ms  28 ms
 8  dallas1.tx.alter.net  80 ms  67 ms  69 ms
```

where the `128.252.x.y` routers are local to WUSTL (`traceroutes` to `bsdi` stop in Dallas, as explained in § 6.7.4).

The STARnet routing remained split for many more months. Here is a traceroute from `wustl` to `umann`, taken on July 2, 1995:

```
 1  128.252.166.249  3 ms  2 ms  2 ms
 2  128.252.123.254  4 ms  2 ms  2 ms
 3  128.252.5.120  4 ms  2 ms  2 ms
 4  128.252.41.2  4 ms  3 ms  3 ms
 5  199.217.253.1  4 ms  6 ms  11 ms
 6  144.228.73.17  71 ms 144.228.27.5  41 ms 144.228.73.17  166 ms
 7  144.228.20.8  30 ms 144.228.70.1  151 ms  56 ms
 8  144.228.10.29  87 ms 144.228.10.42  61 ms 144.228.10.29  90 ms
 9  144.228.30.5  143 ms  258 ms 192.41.177.252  35 ms
10  144.228.10.17  91 ms 134.55.12.161  81 ms  67 ms
11  192.188.33.10  138 ms  159 ms 144.228.10.42  74 ms
12  192.41.177.252  79 ms  73 ms  74 ms
13  153.17.200.105  198 ms *  220 ms
14  192.188.33.10  202 ms * *
15  193.174.74.141  224 ms 134.155.48.125  245 ms  214 ms
```

Fluttering occurs downstream of the hop 5 router:

```
 5  stl1-e0.starnet.net  4 ms  6 ms  11 ms
 6  Anaheim-3  71 ms Washington-DC-7  41 ms Anaheim-3  166 ms
```

and continues from there. This example is slightly different from the previous ones we looked at, in that the STARnet routers `stl2-e0.starnet.net` and `stl3-e0.starnet.net` no longer appear. Instead, it looks like `stl1-e0.starnet.net` is doing its own load-splitting between `sl-ana-3-s3/1-t1.sprintlink.net` and `sl-dc-7-s7-t1.sprintlink.net`, on opposite sides of the country.

STARnet has since switched to a single connection (via MCI), so this pathology no longer occurs [My95].

In § 13.1.3 we analyze the effects that the split-routing had upon TCP performance. Surprisingly, it was generally quite minor. While `wustl` packets very often arrived out of order, they only very rarely arrived so far out of order as to trigger a spurious fast retransmission, as discussed in § 6.6.5 below.

6.6.3 Fluttering at another site

Putting aside `traceroute` probes initiated at `wustl`, of the remaining 6,079 $\mathcal{R}_1$ probes, 295 (about 5%) exhibited fluttering. None of these sites suffered such extreme fluttering as `wustl`; all of the flutters affected either a single hop or at most two hops. Here is an example of a two-hop flutter, between `ncar` and `ucol`, both sited in Boulder, Colorado:

```
1  north-gw.scd.ucar.edu  3 ms  2 ms  2 ms
2  server-gw.ucar.edu  3 ms  2 ms  2 ms
3  cu-gw.ucar.edu  4 ms  3 ms  3 ms
4  129.19.248.62  5 ms cu-ncar.co.westnet.net  5 ms 129.19.248.62  6 ms
5  cs-gw.colorado.edu  6 ms  6 ms  5 ms
6  lewis.cs.colorado.edu  8 ms  19 ms  9 ms
```

The 4th hop shows a flutter from `129.19.248.62` (at Colorado State University) to `cu-ncar .co.westnet.net` and back again. We note that the problem occurred during a hop to Colorado State University, which suggests that those routers may be prone to fluttering. Indeed, of the 295 remaining flutters, 277 involved `ucol`. For all but 6 of these, the fluttering occurred immediately downstream from either the `cu-gw.colorado.edu` router (for traffic outbound from `ucol`) or the `cu-gw.ucar.edu` (traffic inbound to `ucol`). It appears that these routers were splitting load just as did the STARnet router in the previous section, but both downstream routers they alternated between had the same view of subsequent wide-area routing, so the effect remained localized.

Neither the `ucol` nor the `wustl` fluttering was present in $\mathcal{R}_2$. The only repeated pattern we found was that every route originating at `sdsc` that passed through `nynap-sdsc-atm-ds3.cerf.net` suffered from downstream fluttering. Here is an example, from a `traceroute` to `adv`:

```
1  tigerfish.sdsc.edu  8 ms  8 ms  8 ms
2  mobydick.cerf.net  85 ms  246 ms  18 ms
3  nynap-sdsc-atm-ds3.cerf.net  475 ms  380 ms  71 ms
4  sprintnap.ans.net 73 ms t3-3.cnss32.new-york.t3.ans.net 75 ms 77 ms
5  cnss33.new-york.t3.ans.net  76 ms  77 ms  76 ms
6  enss240.t3.ans.net  80 ms  80 ms  79 ms
7  enss240.t3.ans.net  173 ms betelgeuse.advanced.org  81 ms  87 ms
```

There were only 7 of these, however, so their overall impact on routing performance in $\mathcal{R}_2$ was insignificant.

6.6.4 Skipping

When analyzing the traces for fluttering, we notice an interesting anomaly in which routers were visited "prematurely." Here is an example, taken from an xor ⇒ ucl traceroute:

```
 1 xor-gw.xor.com 0 ms 0 ms 10 ms
 2 gw1.boulder.co.coop.net 0 ms 0 ms 0 ms
 3 sl-fw-2-s9-t1.sprintlink.net 30 ms 30 ms 30 ms
 4 sl-fw-5-f1/0.sprintlink.net 30 ms 20 ms 40 ms
 5 sl-dc-8-h3/0-t3.sprintlink.net 60 ms 60 ms 60 ms
 6 icm-dc-1-f0/0.icp.net 1520 ms
   icm-london-1-s1-1984k.icp.net 160 ms
   icm-dc-1-f0/0.icp.net 60 ms
 7 icm-london-1-s1-1984k.icp.net 150 ms 140 ms 150 ms
 8 smds-gw.ulcc.ja.net 140 ms 150 ms 140 ms
 9 smds-gw.ucl.ja.net 150 ms 150 ms 140 ms
10 cisco-pb.ucl.ac.uk 160 ms 160 ms 160 ms
11 cisco.cs.ucl.ac.uk 150 ms 160 ms 160 ms
12 neptune.cs.ucl.ac.uk 160 ms 160 ms 170 ms
```

At hop 6, we see flutter between icm-dc-1-f0/0.icp.net and icm-london-1-s1-1984k .icp.net. But hop 7 then reveals that icm-london-1-s1-1984k.icp.net is actually the next hop!

All told, 11 traceroutes in $\mathcal{R}_1$ and 22 in $\mathcal{R}_2$ (at a number of different routers) showed this "skipping" effect. Furthermore, very often the packet return time just prior to the skip was unusually high (note in the example above the return time of 1,520 msec, much larger than any other in the traceroute). It appears that the router was under a period of stress during the time of the skip, and (perhaps due to a forwarding bug only exhibited under high load) a packet was erroneously forwarded without decrementing and checking its TTL. The downstream router then decremented the TTL, noted it had expired, and returned an ICMP message. The upstream router subsequently recovered from the error condition and continued to correctly forward packets, as is shown for the third probe of hop 6 above.

If the source of the router load were network traffic, then the response from the downstream router should have been heavily delayed too, but, as shown above, it was not. Another explanation is that the load was instead due to the upstream router processing a routing update. This agrees with the fact that the router recovered quickly from the load condition: all that was needed was a single packet's worth of time (about 160 msec above) for the load to disappear.

That a router might, under stress, forward a packet without decrementing its TTL raises a possibility of network instability. If the router stress was due to a routing loop, packets might circulate around the loop indefinitely because their TTL's would not correctly expire, which might in turn maintain the router stress.

We considered traceroutes exhibiting "skipping" as reflecting a pathology separate from "fluttering," since the underlying mechanisms (load-balancing vs. an apparent packet forwarding error) are quite different.

6.6.5 Significance of fluttering

While fluttering can provide benefits as a way to balance load in a network, it also creates a number of problems for different networking applications:

1. A fluttering network path presents the difficulties that arise from *unstable* network paths, as discussed in § 7.1: difficult-to-predict behavior, potential inconsistencies in state information created in the routers on behalf of connections, and problems with constructing consistent measurements of the network's condition. However, if fluttering occurs only at a larger granularity than individual packets—for example, per connection or per end-to-end "flow"—then these problems are ameliorated.

2. If the fluttering only occurs in one direction (as it does for `wustl`, but not for `ucol`), then the path is necessarily *partially asymmetric*, too, suffering from the problems discussed in § 8.1: difficulties in computing unidirectional latencies for protocols such as NTP, difficulties in using "sender-only" measurement techniques, and inefficiencies in keeping state for bidirectional flows.

3. Constructing reliable estimates of the path characteristics, such as round-trip time and available bandwidth, becomes potentially very difficult, since in fact there may be *two* different sets of values to estimate.

4. When the two routes have different propagation times, such as many of those from the `wustl` site, then packets will often arrive at the destination out-of-order, depending on whether they took the shorter route or the longer route. At a minimum, this can lead to extra processing at the receiver to reassemble the out-of-order data stream.

   It can lead to a more serious problem for TCP connections, however. Whenever a TCP endpoint receives an out-of-order packet, the receipt triggers the sending of a redundant acknowledgement in reply, as a mechanism for informing the sender that the receiver has a hole in its sequence space. If three out-of-order packets arrive in a row, then the receiver will generate three redundant acknowledgements. These are enough in turn to trigger "fast retransmission" by the sender (§ 9.2.7), leading it to needlessly retransmit data. Thus, out-of-order delivery can result in redundant network traffic, both due to the extra acknowledgements, and due to possible data retransmissions. We explore this phenomenon further in § 13.1.3.

These problems all argue for eliminating large-scale fluttering whenever possible, where we define fluttering as large-scale if it leads to significantly different routes (as it does for `wustl`). On the other hand, when the effects of the flutter are confined, as for `ucol`, or invisible at the network layer (such as split-routing used at the link layer, which would not show up at all in our study), then these problems are all ameliorated.

Finally, we note that "deflection" and "dispersion" routing schemes that forward packets along varying or multiple paths have many of the characteristics of fluttering paths [BDG95, GK97]. While these schemes can offer benefits in terms of simplified routing decisions, enhanced throughput, and resilience, they bring with them the difficulties discussed above. From the discussion of dispersion routing in [GK97], it appears that the literature in that area to date has only considered the problem of out-of-order delivery, which is addressed simply by noting that the schemes require a resequencing buffer.

Failure mode	# Failures	Notes
Host down	81 (65 %)	umann, sdsc, and inria accounted for 93%
Stub network outage	31 (25 %)	ustutt accounted for 74% of these
Infrastructure failure	13 (10 %)	no dominant pattern

Table VIII: Failure modes for unreachable hosts in $\mathcal{R}_1$

Failure mode	# Failures	Notes
Host down	277 (45 %)	panix accounted for 61% of these
Stub network outage	170 (27.5 %)	nrao accounted for 57% of these
Infrastructure failure	170 (27.5 %)	no dominant pattern

Table IX: Failure modes for unreachable hosts in $\mathcal{R}_2$

6.7 Unreachability

In addition to traceroute failures due to persistent routing loops and erroneous routing, 125 of the $\mathcal{R}_1$ traceroutes and 617 of the $\mathcal{R}_2$ traceroutes failed to reach the destination host for other reasons. We analyzed these failures to determine the corresponding failure modes, summarized in Tables VIII and IX.

6.7.1 Host down

We concluded that a host was down (first row) if the traceroute to it terminated at one of the routers which in another traceroute proved to be the penultimate hop to that host. In $\mathcal{R}_1$, this occurred 81 times out of a total of 6,459 traceroutes, giving us an unconditional probability that a site participating in our study was down during an experiment of $p \approx 1.25\%$. This probability corresponds to an availability of $\approx 98.75\%$, Similarly, for $\mathcal{R}_2$ we get an availability of $\approx 99.2\%$. These values are a bit higher than the median availability of 97.2% reported in [LMG95], though our "polling" frequency is lower than theirs (a mean of 10 minutes), which could explain the discrepancy. Also, as noted in § 4.4, our sites do *not* plausibly constitute a random sample of Internet hosts (while [LMG95]'s sites are much closer to such), so disagreement between the two figures is not particularly significant. Finally, note that most of the failures were due to just a few of the sites, as indicated in the tables.

6.7.2 Stub network outage

We classified an Unreachability failure as a "stub network outage" (second row) if the final router reached during the traceroute was sited inside the same institute as the endpoint (but not a penultimate hop), or at the border between the institute and the remainder of the Internet.[10]

[10] Such a failure could also occur at the traceroute source's institute. One might think we would never observe this in our traces because, in order to generate a traceroute, the npd_control site had to be able to connect to

The numbers of observations of such failures correspond to availabilities of 99.5% for both $\mathcal{R}_1$ and $\mathcal{R}_2$, though again we cannot draw a general conclusion about connectivity to Internet sites because our collection of participating sites might not be representative. We also need to be wary about generality given the strong dominance of this type of failure by routes to the `ustutt` and `nrao`[11] sites.

On the other hand, the prevalence of network outages to `ustutt` gives us an opportunity to assess how quickly a router learns that the next-hop router has crashed. If a router does not have a route to a packet's destination, the router is required to generate some form of ICMP "Destination Unreachable" message [Ba95]. However, a router *may not know* that it has no route to the packet's destination, because it is unaware that the next-hop router has crashed. These two cases result in different `traceroute` behavior: the first elicits a "`!H`" (or "`!N`") response in the `traceroute` output, while the second will simply show a dropped packet. Consider the following `traceroute` from `ukc` to `ustutt`:

```
 1  rtcomp.ukc.ac.uk  2 ms  2 ms  2 ms
 2  brtcomp.ukc.ac.uk  2 ms  2 ms  2 ms
 3  brtsj.ukc.ac.uk  3 ms  3 ms  3 ms
 4  smds-gw.ulcc.ja.net  7 ms  7 ms  6 ms
 5  eu-gw.ja.net  8 ms  8 ms  6 ms
 6  london4.empb.net  12 ms  11 ms  8 ms
 7  duesseldorf2.empb.net  33 ms  31 ms  38 ms
 8  ipgate2.win-ip.dfn.de  91 ms  52 ms  46 ms
 9  duesseldorf4.win-ip.dfn.de  70 ms  44 ms  32 ms
10  stuttgart4.belwue.de  67 ms  68 ms  56 ms
11  stuttgart1.belwue.de  84 ms  85 ms  74 ms
12  belwue-gw.uni-stuttgart.de  63 ms  57 ms  69 ms
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  * * belwue-gw.uni-stuttgart.de  68 ms !H
24  * * *
25  * * *
26  * * *
27  * * *
28  * * *
29  * * *
30  * * belwue-gw.uni-stuttgart.de  64 ms !H
```

the source in the first place. However, some sources have multiple connections to the Internet, and we did observe several instances where we were able to connect to a source but it was unable to advance packets to any routers outside of its site. We include these instances in the tables as stub network outages.

[11] It turns out that the entire `nrao` site was intentionally disconnected from the Internet from November 28 through December 6, 1995, following a serious break-in by a network cracker.

Hop 12 makes it to `belwue-gw.uni-stuttgart.de`, `ustutt`'s border router. Normally the next hop would be to `cisco1.rus.uni-stuttgart.de`, inside the `ustutt` site, and hop 12 gives no indication of an impending problem here. But the next 36 packets are dropped, reflecting an outage of 3.5 minutes. At hop 23, `belwue-gw.uni-stuttgart.de` again responds, but this time includes an ICMP unreachable message. Thus, it appears that it took `belwue-gw.uni-stuttgart.de` at least 3.5 minutes to learn that the next hop had crashed.

What follows, from hops 24-30, remains a puzzle: `belwue-gw.uni-stuttgart.de` apparently forgets that the next-hop router has crashed and only relearns the fact after another 100 seconds. At this point the `traceroute` terminates because it has reached the 30-hop limit.

Of the 23 $\mathcal{R}_1$ stub network outages involving `ustutt`, 19 exhibited this pattern.[12] For those 19, the learning periods range from 0 seconds (the router immediately knew that the next hop was unavailable) to 170 seconds, with a median of 30 seconds and a mean of 50 seconds (distributed roughly exponentially–see § 6.8 for the significance of this). For the other four `ustutt` outages, the router failed to learn the unavailability of the downstream hop before the `traceroute` terminated due to the 30-hop limit. These failures spanned between 105 and 225 seconds, so those give lower bounds on the learning time.

Clearly, for `belwue-gw.uni-stuttgart.de`, the router does not quickly learn about a next-hop crash. If this slow response is typical (we lack enough data to know if it is), then Internet traffic is subject to outages on the order of a minute whenever a router crashes. This finding is consistent with the BGP specification, which recommends that routers wait for 90 seconds' worth of unanswered polls before deciding that a peer is unreachable [RL95]. The higher this figure is, the less prone a network is to routing oscillations; but high delays in detecting unreachable peers also present serious difficulties for real-time protocols that need to quickly adapt to such faults [GR95].

6.7.3 Infrastructure failure

The final type of failure (third row in each table) reflects a problem inside the Internet infrastructure: the terminating router in the `traceroute` was in the middle of the network, not at the source or destination.[13] In this case, we *can* make a general statement about availability, since the basis for our study is the assumption that the collection of routes between our sites *is* representative of Internet connectivity as a whole (§ 4.4). A total of 13 failures out of 6,459 $\mathcal{R}_1$ observations corresponds to an Internet infrastructure availability of 99.8%, while for $\mathcal{R}_2$ this percentage drops to 99.5%. The difference is significant using the methodology discussed in § 4.5. If we add to these failures the instances of persistent routing loops (§ 6.3.1) and erroneous routing (§ 6.4), then the $\mathcal{R}_1$

[12] All of the `ustutt` outages occurred between the early morning of Saturday, December 10th and the early morning of Monday, December 12th (Stuttgart time), indicating that the crashed router was down for the weekend.

[13] In some cases, such a termination can still reflect an unreachable host or a stub network outage, if the unreachability information has been propagated into the interior of the network. However, in these cases we would expect that the information is not propagated *deeply* into the network, since the need to "aggregate" routing information means that information pertaining to individual host or stub network outages cannot be propagated beyond the point at which it is aggregated with information for other, reachable hosts or networks.

We inspected the points in the terminating routers for the infrastructure failures and found that in the vast majority of cases, the router was sited far from the unreachable destination. For example, we observed several infrastructure failures for `traceroutes` going from `bnl` to European sites, each of which terminated at `ames-llnl.es.net` in California. Such a termination is much more likely to reflect loss of general connectivity to Europe, than an outage of a single European site being propagated all the way to a router in California.

availability falls to 99.6%, and that for $\mathcal{R}_2$ to 99.35%. We must bear in mind, however, that these numbers will be skewed by the fairly large proportion of our attempted measurements that failed due to an inability to contact the remote `npd` site (§ 5.2); some of these failures could be due to infrastructure problems, making these availability numbers overestimates.

A solid figure for Internet infrastructure availability is important for network service providers wishing to provide a form of *guaranteed service* in which the guarantees carry legal (contractual) obligations [Fe90, PaFe94]. We do not claim that the availabilities given in the preceding paragraph are such solid figures, but they are a step in that direction.[14]

6.7.4 Consistently unreachable hosts

Several hosts in our study were either always or frequently unreachable. Those always unreachable—`bsdi` in $\mathcal{R}_1$, and `oce` and `ucol` in $\mathcal{R}_2$—all reside behind firewalls that drop incoming, unidentified UDP packets (such as used by `traceroute`; § 4.2.3), so `traceroutes` to it always showed connectivity lost after the hop prior to the firewall. We adjusted for this behavior by considering any `traceroutes` that made it to that hop as making it all the way to the host.

The other frequently unreachable host, `lbli`, is connected to the Internet via an ISDN circuit. This circuit disconnects after any idle period during which `lbli` did not use the circuit for a configurable amount of time (typically 10-20 minutes). Thus, many `traceroutes` to `lbli` found the circuit down, and terminated at the Internet side of the ISDN link. As with the firewall hosts, we considered these `traceroutes` as having successfully reach the `lbli` host.

The net effect of these adjustments is to introduce possible underestimation into our assessment of the prevalence of stub network outages and hosts being down. Most likely, this introduced bias is quite slight, given how our stub network outages and downed hosts statistics were dominated by just a few sites anyway.

6.7.5 Unreachable due to too many hops

As noted in § 4.2.1, `traceroute` by default probes up to 30 hops of the route between two hosts. This length sufficed for all of the $\mathcal{R}_1$ measurements, and all but 6 of the $\mathcal{R}_2$ measurements.[15] The fact that it failed occasionally in $\mathcal{R}_2$, however, indicates that the operational diameter of the Internet has grown beyond 30 hops, and argues for using large initial TTL values when a host originates an IP datagram. In informal studies of the link connecting the Lawrence Berkeley National Laboratory to the rest of Internet, we have found that most hosts send IP datagrams with TTL's well above 30, but a non-negligible proportion of the datagrams (10% in one dataset) appear to have been sent with TTL's of around 30.

While routes of more than 30 hops were not correctly measured by `traceroute` in our experiment, they were so rare as to not present any significant source of error.

A final note concerning large hop counts: it is sometimes assumed that the hop count of a route equates to its geographical distance. While from our data this appears roughly the case, we

[14] Naturally, a network service provider will keep detailed statistics on their own network, and not need a figure such as that we have computed. But if they must deal with other providers for portions of the end-to-end route, such a figure as a rule-of-thumb will prove useful.

[15] 5 of the 6 were to or from `inria`. Routing within France (and international routing in general) often has many hops. The other was between `umont` and `umann`, also international in scope.

noticed some remarkable disagreements, both in terms of a few hops corresponding to large distances, and many hops corresponding to little distance. For example, the shortest route we observed from `ncar`, in Colorado, to `sdsc`, in southern California (about 1,500 km distant), was three hops:

```
cs-vbns.ucar.edu
cs-atm0-0-3.sdsc.vbns.net
rintrah.sdsc.edu
```

This route traveled over the VBNS ATM backbone (recall from § 4.2.3 that `traceroute` elicits paths at the *network layer*, and does not measure any "hops" made at the link layer). We also observed in $\mathcal{R}_1$ a 5 hop route from `pubnix` to `bsdi`, about 2,000 km distant.

On the other hand, all of the routes we observed between `mit` and `harv` (in either direction), sited about 3 km apart, were 11 hops, and we observed 14 and 17 hop routes between `sri` and `lbl`, about 50 km apart.

6.8 Temporary outages

The final pathology we studied was temporary network outages. When a sequence of consecutive `traceroute` probes are lost, the most likely cause is either a temporary loss of network connectivity, or very heavy congestion lasting 10's of seconds. For each `traceroute`, we examined its longest period of consecutive probe losses (other than consecutive losses at the end of a `traceroute` when, for example, the endpoint was unreachable).

The resulting distribution of the number of probes lost appears trimodal. In $\mathcal{R}_1$ ($\mathcal{R}_2$), about 55% (43%) of the `traceroutes` had no losses, 44% (55%) had between 1 and 5 losses, and 0.96% (2.2%) had 6 or more losses[16]

Of these latter, after eliminating those to `ukc` in $\mathcal{R}_1$ (because these "outages" are actually unresponsive routers; see § 6.1), the distribution of the number of probes lost in the $\mathcal{R}_1$ data is quite close to geometric. Figure 6.2 plots the outage duration on the x-axis vs. the probability of observing that duration or larger on the y-axis (logarithmically scaled). The outage duration is determined by the number of probe losses multiplied by 5 seconds per lost probe. The line added to the plot corresponds to what would be expected for a geometric distribution with probability $p = 0.92$ that a probe beyond the 5th is dropped. (The line appears straight due to the logarithmic y-axis scale and the fact that the geometric distribution is the discrete counterpart to the exponential distribution.) As can be seen, the fit is fairly good, especially in the tail.

From the above evidence it is reasonable to argue that long outages are well-modeled as persisting for 30 seconds plus an exponentially distributed random variable with mean equal to about 40 seconds. This finding would be convenient, since the exponential distribution often makes for tractable analysis.

If we turn to the $\mathcal{R}_2$ data, however, we find that the geometric tail with $p = 0.92$ is still present, but only for outages more than 75 seconds long, as illustrated in Figure 6.3. For outages between 30 and 70 seconds, the duration still exhibits a geometric distribution, but with $p = 0.62$, suggesting two different recovery mechanisms, one operating on time scales of 30 seconds to a minute or so and the other on significantly longer time scales.

[16] Recall from § 4.2.3 that probe "losses" can also be due to ICMP rate-limiting, which we do not differentiate. We analyze true packet losses in much greater detail in Chapter 15.

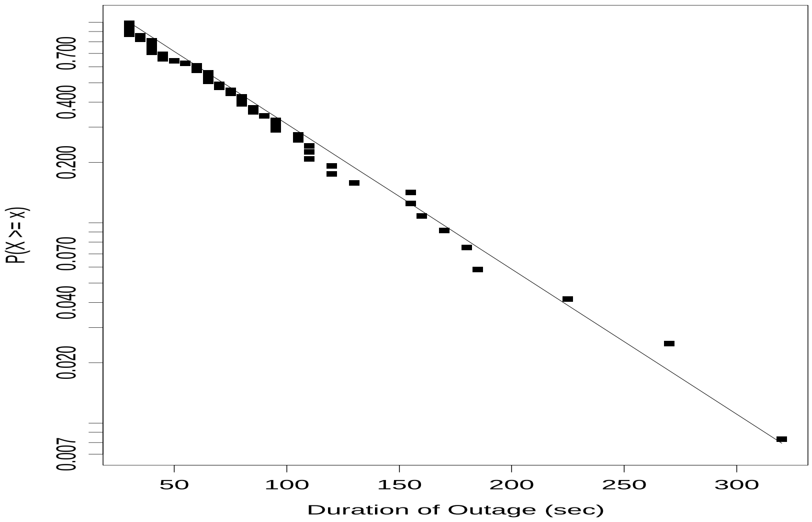

Figure 6.2: Distribution of long $\mathcal{R}_1$ outages

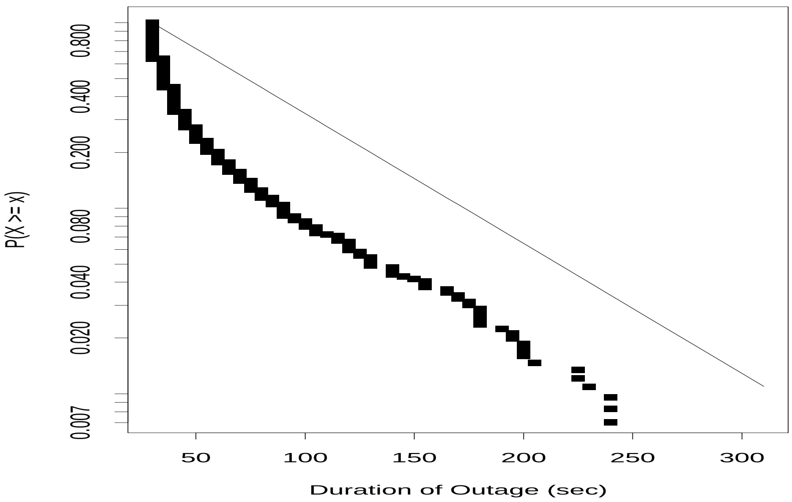

Figure 6.3: Distribution of long $\mathcal{R}_2$ outages

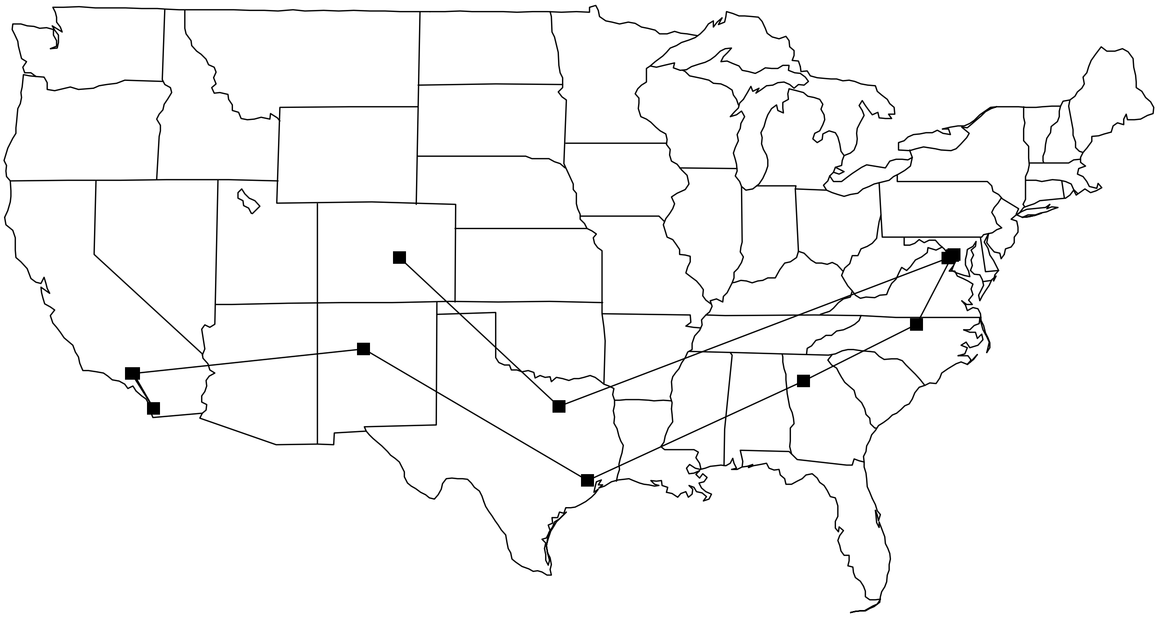

Figure 6.4: Circuitous route from `bsdi` to `usc`

Note that § 6.7.2 provides separate evidence that the time taken for routers to recover from the loss of a next-hop router is exponentially distributed, with a mean of 50 seconds (shorter than the $\mathcal{R}_1$ fit, but in agreement with the $\mathcal{R}_2$ data).

6.9 Circuitous routing

Since the inception of the Internet Protocol, one of its main goals has been resilience in the presence of network failures [Cl88]. In this section we document some of the more circuitous routes the network found in order to maintain connectivity in the presence of failures. These routes do not represent pathologies *per se* but rather triumphs of robust routing, or, sometimes, simply the lack of the necessary infrastructure to take advantage of more direct routes.

Figure 6.4 shows a route used from `bsdi`, in Colorado Springs, Colorado, to `usc`, in Los Angeles, California. The route is perhaps three times longer than the `bsdi` route to `sri` (located in Northern California), which also makes a first hop to Dallas, Texas, but from there travels to San Jose, California, rather than to the East coast.

Figure 6.5 shows one of the routes used from `lbli`, in Berkeley, California, to `ucol`, in Boulder, Colorado. Here the packets travel all the way to the East coast, then back to the West coast, and finally over to Colorado. A more direct path, also present in our data, travels straight from New Mexico to Colorado. Presumably this link was unavailable during the time of the longer route.

Figure 6.6 shows a route from `nrao`, in Charlottesville, Virginia, to `wustl`, in St. Louis, Missouri. This route increases the distance of the more direct route we also observed (via Washington, D.C., and then straight to St. Louis) by roughly a factor of five.

Figure 6.7 shows an even more tortuous route to `wustl`, this time from `lbl`. The packets

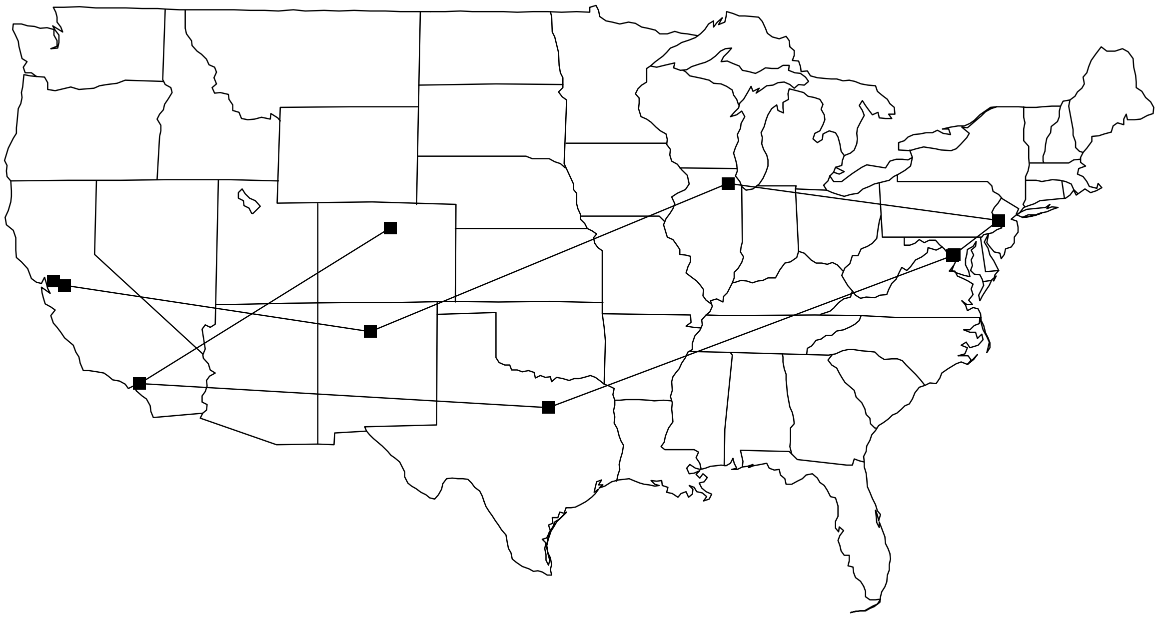

Figure 6.5: Circuitous route from `lbli` to `ucol`

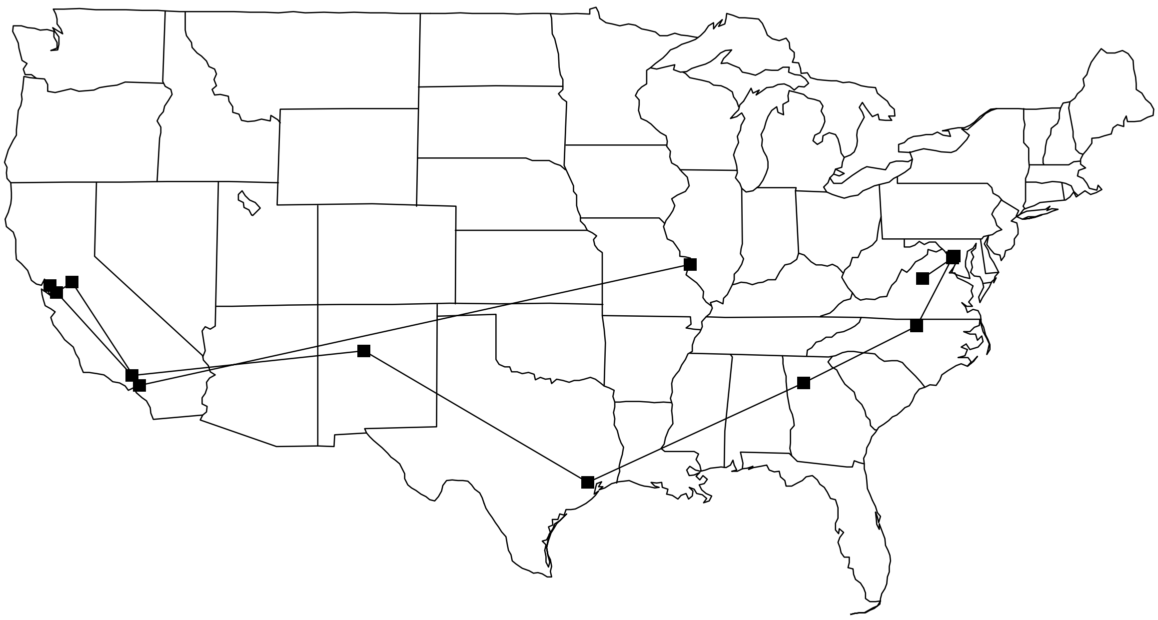

Figure 6.6: Circuitous route from `nrao` to `wustl`

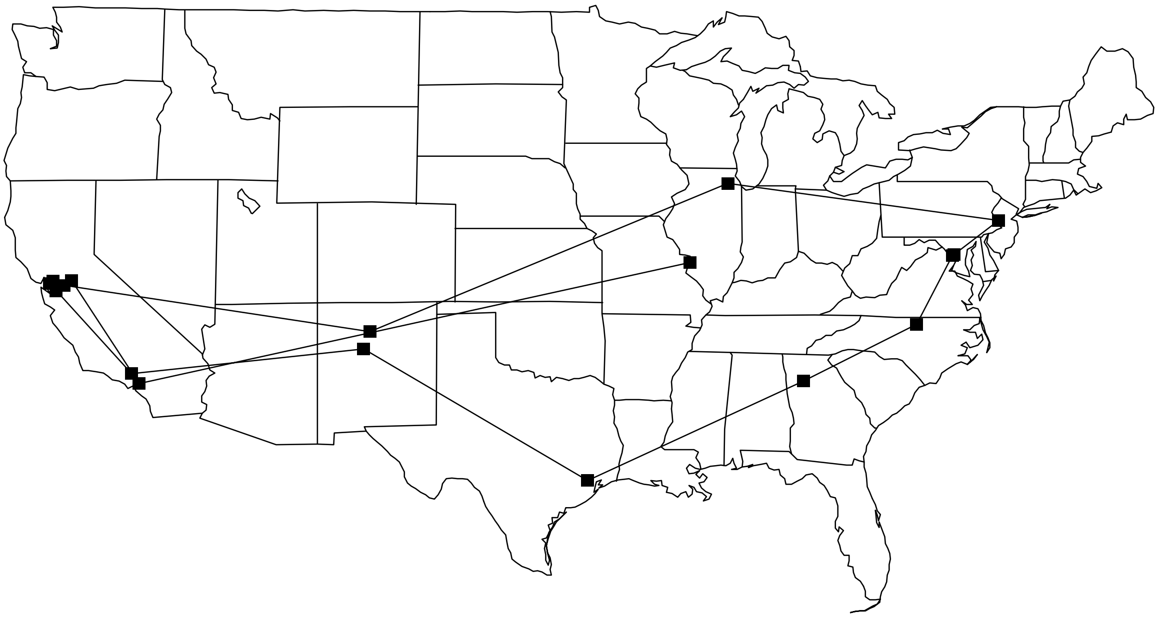

Figure 6.7: Circuitous route from `lbl` to `wustl`

first travel to Livermore, California, and then Los Alamos, New Mexico, via ESNET. They continue up to Illinois and across to Washington, D.C., via Princeton, New Jersey, and College Park, Maryland. They next take a southern route all the way back to northern California (!), back to southern California, and finally across to St. Louis. Figure 6.8 shows the 29 hops making up this path. One might be tempted to conclude that the path must have been the product of some sort of one-time glitch, but it showed up 5 different times in the $\mathcal{R}_1$ data.

In Figure 6.9 we see an illustration of the difficulties sometimes encountered even when going a very short distance. This route was the only one we observed from `ncar` to `xor` (8 observations total). `ncar` is located in Boulder, Colorado, and `xor` in East Boulder, Colorado, a few miles to the east. Yet the route between them visits the Gulf of Mexico and the East coast before crossing those few miles.

Circuitous routing is not limited to the United States. Figure 6.10 shows the route from `inria`, located in Southern France, to `oce`, located in the Netherlands, a few hundred kilometers to the North. The routing takes the packets across the Atlantic ocean to Vienna, Virginia (and nearby Falls Church), before crossing the Atlantic again to Amsterdam. The return path from `oce` to `inria` also follows this path, except in one instance the routing went from Amsterdam to Paris via Vienna, Austria (shown with a dotted line), rather than Vienna, Virginia. We speculated that perhaps the trans-Atlantic routing was due simply to accidental misconfiguration based on the similarities of the names; but we learned from EUnet personnel that much more likely the trans-Atlantic routing was intentional, due to its low-cost and higher available capacity compared to the underprovisioned intra-European links [Bi95].

Persistent circuitous routing might strike us as pathological, and unexpected in a well-run network. Because we do not know the underlying reasons for the routing configurations, we are unable from our data to answer why circuitous routing exists. We speculate, however, that

```
ir6gw.lbl.gov                          (Berkeley, CA)
er1gw.lbl.gov
lbl-lc2-1.es.net
llnl-lbl-t3.es.net                     (Livermore, CA)
lanl-llnl-t3.es.net                    (Los Alamos, NM)
fnal-lanl-t3.es.net                    (Batavia, IL)
pppl-fnal-t3.es.net                    (Princeton, NJ)
pppl-nis.es.net
umd-pppl.es.net                        (College Park, MD)
mf-0.enss145.t3.ans.net
t3-2.cnss56.washington-dc.t3.ans.net  (Washington, DC)
t3-1.cnss72.greensboro.t3.ans.net     (Greensboro, NC)
t3-0.cnss104.atlanta.t3.ans.net       (Atlanta, GA)
t3-2.cnss64.houston.t3.ans.net        (Houston, TX)
t3-0.cnss112.albuquerque.t3.ans.net   (Albuquerque, NM)
t3-1.cnss16.los-angeles.t3.ans.net    (Los Angeles, CA)
t3-2.cnss8.san-francisco.t3.ans.net   (San Francisco, CA)
t3-0.enss144.t3.ans.net               (Moffett Field, CA)
fix-w.icm.net
sl-stk-5-h2/0-t3.sprintlink.net       (Stockton, CA)
sl-stk-6-f0/0.sprintlink.net
sl-ana-2-h4/0-t3.sprintlink.net       (Anaheim, CA)
sl-ana-3-f0/0.sprintlink.net
sl-starnet2-1-s0-t1.sprintlink.net    (St. Louis, MO)
stl2-e0.starnet.net
ncrc-acn.wustl.edu
ncrc-eng.wustl.edu
jcr.ecl.wustl.edu
tango.cs.wustl.edu
```

Figure 6.8: Individual routers comprising circuitous path from `lbl` to `wustl`

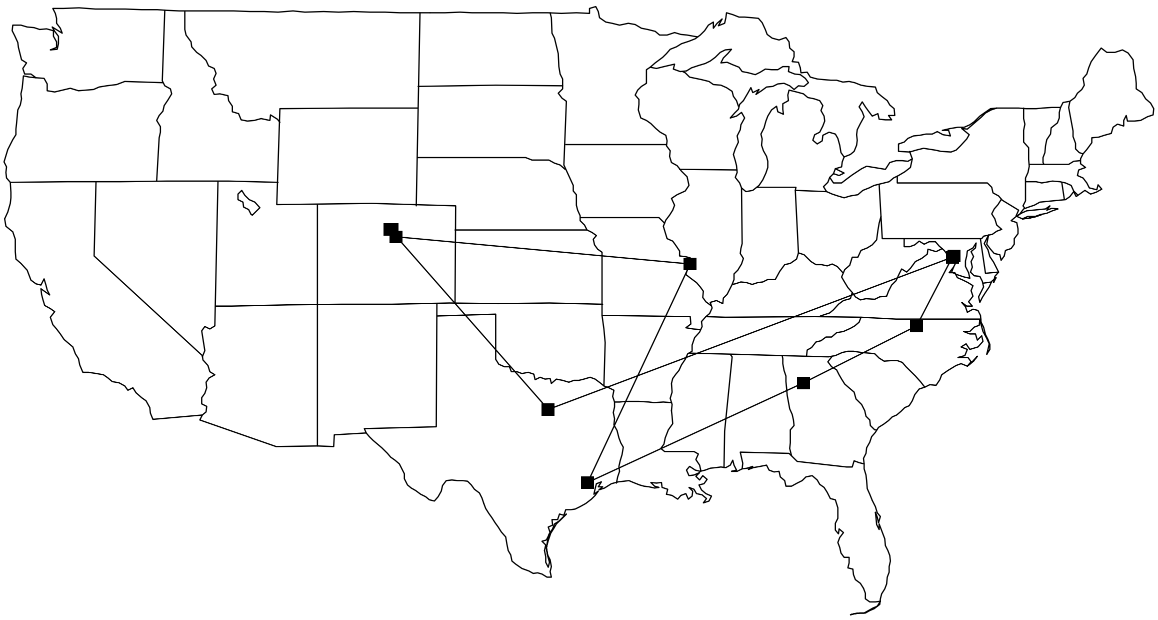

Figure 6.9: Circuitous route from `ncar` to `xor`

Figure 6.10: Circuitous route from `inria` to `oce`

Pathology	Probability	Trend	Notes
Unresponsive routers	0.00–0.53%		Rare enough to not present a measurement problem.
Failure to decrement TTL	0.18% $\perp$ 0.06%	better	Downstream router visited prematurely.
Persistent routing loops	0.13–0.16%		Some lasted for hours.
Temporary routing loops	0.055–0.078%		
Erroneous routing	0.004–0.004%		Packets in $\mathcal{R}_1$ visited Israel! No instances in $\mathcal{R}_2$.
Connectivity altered mid-stream	0.16% $\perp$ 0.44%	worse	Suggests rapidly varying routes.
Infrastructure failure	0.21% $\perp$ 0.48%	worse	No dominant link.
Temporary outage $\geq$ 30 secs	0.96% $\perp$ 2.2%	worse	Outage duration distributed as constant plus exponential. This distribution in $\mathcal{R}_2$ is bimodal.
Total user-visible pathologies	1.5% $\perp$ 3.4%	worse	

Table X: Summary of representative routing pathologies

it may be an inevitable consequence of the structure of today's Internet: the network is so vast and heterogeneous, and so under-instrumented for purposes of diagnosing end-to-end ailments, that errors inexorably arise and persist for long periods of time.

6.10 Summary

Table X summarizes the routing pathologies we studied in this section. The table is confined to those pathologies for which we claim our samples are representative (§ 4.4). (So, for example, we omit the "fluttering" pathology, which was heavily dominated by a pair of sites in our study; and also "host down," and "stub network outage.") The first part of the table reflects pathologies that are *not* in general visible to an end-to-end user of the network; that is, their presence does not significantly impact most network users. The second part of the table summarizes pathologies that *are* user-visible.

The second column gives the probability of observing the pathology, in two forms. When the probability is given as a range, such as for "persistent routing loops," then the proportion of observations of the pathology in $\mathcal{R}_1$ was consistent with the proportion in $\mathcal{R}_2$ (using the methodology in § 4.5). The range reflects the values spanned by the two datasets.

When the table lists two probabilities separated by "$\perp$," then the proportion of $\mathcal{R}_1$ observations was *inconsistent*, with 95% confidence, with the proportion of $\mathcal{R}_2$ observations. The first probability applies to the $\mathcal{R}_1$ measurements, and reflects the state of the Internet at the end of 1994; and the second to the $\mathcal{R}_2$ measurements, reflecting the state at the end of 1995.

For those pathologies with inconsistent probabilities, the third column assesses the trend during the year separating the $\mathcal{R}_1$ and $\mathcal{R}_2$ measurements. A trend of "better" indicates that the situation improved, and "worse" that it degraded. One pathology improved significantly: the likelihood of a router failing to decrement the TTL decreased. This change likely reflects upgraded and more stable router software.

Note though that this pathology is of no interest to end-to-end users of the network—improvements in the pathology do not reflect any significant gains in network service for the user. On the other hand, of the pathologies given in the second part of the table, which *are* of interest to users, *none of them improved!*, and *a number became significantly worse*.

The final row summarizes the total probability of observing a user-visible pathology. We note that: *During 1995, the likelihood of a user encountering a serious end-to-end routing problem more than doubled, to 1 in 30.* The most prevalent of these problems was an outage lasting more than 30 seconds.

This finding raises concerns regarding the long-term stability of the Internet. Clearly, if the trend continues, then network service will degrade to unacceptable levels. Unfortunately, from only two points in time it is impossible to assess the actual likelihood of the trend continuing.

Finally, we note that, for reasons given in § 5.2, our estimates of the prevalence of pathologies are biased towards underestimation; the true figures are most likely somewhat higher.