关队列的交互式 Java 小程序。如果你将分组到达率设置得足够大，使流量强度超过 1，那么将看到经过一段时间后，队列慢慢地建立起来。

丢包

在上述讨论中，我们已经假设队列能够容纳无穷多的分组。在现实中，一条链路前的队列只有有限的容量，尽管排队容量极大地依赖于路由器设计和成本。因为该排队容量是有限的，随着流量强度接近 1，排队时延并不实际趋向无穷大。相反，到达的分组将发现一个满的队列。由于没有地方存储这个分组，路由器将**丢弃**（drop）该分组，即该分组将会**丢失**（lost）。当流量强度大于 1 时，队列中的这种溢出也能够在用于队列的 Java 小程序中看到。

从端系统的角度看，上述丢包现象看起来是一个分组已经传输到网络核心，但它绝不会从网络发送到目的地。分组丢失的份额随着流量强度增加而增加。因此，一个结点的性能常常不仅根据时延来度量，而且根据分组丢失的概率来度量。正如我们将在后面各章中讨论的那样，丢失的分组可能基于端到端的原则重传，以确保所有的数据最终从源传送到了目的地。

1.4.3 端到端时延

前面的讨论一直集中在结点时延上，即在单台路由器上的时延。我们现在考虑从源到目的地的总时延。为了能够理解这个概念，假定在源主机和目的主机之间有 $N-1$ 台路由器。我们还要假设该网络此时是无拥塞的（因此排队时延是微不足道的），在每台路由器和源主机上的处理时延是 d_{proc}，每台路由器和源主机的输出速率是 R bps，每条链路的传播时延是 d_{prop}。结点时延累加起来，得到端到端时延：

$$d_{\text{end-end}} = N(d_{\text{proc}} + d_{\text{trans}} + d_{\text{prop}}) \tag{1-2}$$

式中再次有 $d_{\text{trans}} = L/R$，其中 L 是分组长度。值得注意的是，式（1-2）是式（1-1）的一般形式，式（1-1）没有考虑处理时延和传播时延。在各结点具有不同的时延和每个结点存在平均排队时延的情况下，需要对式（1-2）进行一般化处理。我们将有关工作留给读者。

1. Traceroute

为了对计算机网络中的时延有一个第一手认识，我们可以利用 Traceroute 程序。Traceroute 是一个简单的程序，它能够在任何因特网主机上运行。当用户指定一个目的主机名字时，源主机中的该程序朝着该目的地发送多个特殊的分组。当这些分组向着目的地传送时，它们通过一系列路由器。当路由器接收到这些特殊分组之一时，它向源回送一个短报文。该报文包括该路由器名字和地址。

更具体的是，假定在源和目的地之间有 $N-1$ 台路由器。则源将向网络发送 N 个特殊的分组，其中每个分组地址指向最终目的地。这 N 个特殊分组标识为从 1 到 N，第一个分组标识为 1，最后的分组标识为 N。当第 n 台路由器接收到标识为 n 的第 n 个分组时，该路由器不是向它的目的地转发该分组，而是向源回送一个报文。当目的主机接收第 N 个分组时，它也会向源返回一个报文。该源记录了从它发送一个分组到它接收到对应返回报文所经受的时间；它也记录了返回该报文的路由器（或目的地主机）的名字和地址。以这种方式，源能够重建分组从源到目的地所采用的路由，并且该源能够决定到所有中间路由器的往返时延。Traceroute 实际上对刚才描述的实验重复了 3 次，因此该源实际上向目的地