1. Both Link-State and Distance-Vector algorithms are concerned with finding efficient ways to route traffic around the network. Link-state uses a centralized, full knowledge approach to finding the best system. This approach, using an application of Dijkstra's algorithm, means that after information on each router's state is known by every router, they will be able to compute the best paths from any given point to any other point; however, doing this algorithm comes at the trade off of being costly in terms of performance if it is run on well-connected networks, and this expense is avoided by running it on sparse ones. Distance-Vector algorithms, unlike link-state, doesn't require full knowledge of the network. This fact makes it much more resilient to shifts in the network topology. Since it is decentralized, it is slower than link-state; however, it requires no synchronization. The usefulness of both depends entirely on how sparse the network is and how much it could change after initial routes are established.

2. NAT information.
    a. Since the numbers are sequential, it could be possible to determine the number of devices behind the NAT. By tracking the different sequences of numbers, and slotting each packet into a sequence as it came by, you could determine the number of devices by checking how many sequences were tracked. If some device were to randomly choose the next number in a separate device's sequence, we would begin to see multiple packets with the same sequence number, which indicates multiple devices. So we could determine the number of unique hosts behind a NAT in this situation.
    b. The technique outlined above would fall apart as soon as all sequence numbers were randomly assigned. At that point, since above we tracked the order of numbers to count the number of devices, and now there is no order, we would be unable to determine the number of devices.

3. Without modifying the NAT protocol or setting up application specific NAT rules, this becomes nearly impossible. Since each NAT device could be responsible for some large number of other device's traffic, it is not feasible to provide each of those devices with an IP address in order to establish a direct connections. Furthermore, since we aren't allowed to set up application specific NAT configuration due to the constraints of the problem, a solution similar to hole punching as is discussed in RFC 3027 Chapter 5 is not feasible. By the nature of the NAT protocol, computers are not able to establish a direct connection without going through the NAT device.

4. If I were a large corporation with networks spanning the continent, it makes sense to use a combination of RIP and OSPF. One of the main differences between the two protocols is that RIP looks at routing in terms of hop count and OSPF looks at it in terms of path cost. So, for a smaller LAN, such as within a specific branch of the corporation, it would make more sense to use RIP. This protocol would account for the possibilities of devices joining and leaving the network regularly, such as employees connecting throughout the work day and then disconnecting. RIP will handle these situations well while still

providing reliable routing. On the larger, inter-continental scale it makes sense to use OSPF since OSPF will look to minimize path cost instead of hops. When you cross the continent, you would want your communications to take the shortest path so that they can get to their destination promptly, and so OSPF would be the correct choice.