20

Meet the Neighbors

She is a caricature of herself.

— Something my mother used to say when she thought her victim wasn't listening

We started off with a brief overview of the history and socio-political structure of the Browse Service. Basic guidebook stuff. Now we will meet some of the local citizens, learn about their roles in society, and find out how they interact with one another on a personal level. We will introduce them to you one at a time so you can get to know them before you see them in action. Hopefully, that will help you feel more comfortable as you explore the backstreets of the Network Neighborhood.

A quick note before we go visiting... The Browse Service is built on top of other CIFS services and protocols. Layered protocols have a habit of causing terminology confusion, particularly when the talk turns to "clients" and "servers." The thing to keep in mind is that everything is relative. In this case, everything is relative to the service being offered and the client making use of it. An SMB filesharing client may also be a Browse Service server, and a Browse Service server may also be a Browse Service client, and a Browse Service client... the permutations and combinations are practically (though not actually) endless.

Don't let yourself get confused.

To help abate the ensuing chaos a few new, context-specific terms will need to be defined as we go along. You may want to take notes.

20.1 Browse Service Clientele

The Browse Service has two types of clients:

- systems that wish to announce services, and
- systems that wish to find services on the network.

Think of it in terms of the classified advertising section in your local newspaper. Classifieds are available to people who have something to sell as well as to people who are looking to buy. Both of these are clients of the newspaper.

In some of the available documentation, systems that wish to announce services via the Browse List are referred to as "Non-Browser Servers." That's really icky terminology, since those systems *could* be [Potential | Backup | Local Master] Browsers as well. We will refer to nodes that announce services as "Providers," without trying to straighten out what kind of Browser nodes or SMB servers they may or may not be. To add a sense of symmetry, we will use the term "Consumers" to identify the other kind of Browse Service clients — those that want to *find* services on the network.

So, for our purposes:

- A *Provider* is any node that wishes to announce (via the Browse List) that it has services available.
- A *Consumer* is any node that wishes to get hold of a copy of the Browse List so that it can find services.

We promised to introduce the neighbors one at a time. Let's start with the Providers.

20.1.1 *Providers*

Providers announce themselves to the Local Master Browser by periodically broadcasting a message called a `HostAnnouncement Browser Frame`. The message is sent as an IP broadcast so any NBT node could listen in, but the NBT destination given in the message header is the *workgroup*`<1D>` name, so the LMB is obviously the intended recipient.

When a node first starts up, it generally announces itself once per minute. After it has been running for a while it will slow down, typically sending an announcement once every 12 minutes. Different implementations behave

differently, of course, but the suggestion is that the Provider start with a delay of one minute and double the delay until it exceeds 12 minutes, at which point it should settle on 12 minute intervals.

If a Provider stops announcing itself, its entry in the Browse List will (eventually) time out. The time out formula generally given in the documentation is three times the last known announcement period. In testing, however, some systems reported the `Periodicity` value incorrectly so it is probably safer to assume an announcement period of 12 minutes and use a fixed timeout value of 3 × 12 = 36 minutes.

Providers can also remove themselves from the Browse List by sending a `HostAnnouncement` message with an empty list of services. This indicates to the LMB that the host is no longer providing any services. If possible, a Provider should send an empty `HostAnnouncement Browser Frame` when it shuts down.

Some Technologies Shouldn't Mix Alert

*When cable television companies first decided to get into the **I**nternet **S**ervice **P**rovider (ISP) business they ran into an unexpected problem. A lot of PC vendors were installing Windows preconfigured with SMB filesharing turned on and no passwords by default. When these PCs were connected to the cable Internet service, they would start announcing themselves to one another. People up and down the block found that they could both see and access each other's systems, view files, copy software...*

Now that's *a Network Neighborhood.*

20.1.2 *Consumers*

It is important to be polite when dealing with your local government. The LMB is your neighbor, after all, and the time it spends handling the Browse List is volunteer time (unless it is also the appointed DMB). It may have other responsibilities as well — spouse, kids, day job as a word processor or fileserver... If everyone in the neighborhood is constantly asking for help, the LMB may wish that it had never been elected.

The *polite* thing for a Browse Service Consumer to do is to ask the LMB for a list of Backup Browsers. We will call this the "BB List" (short for **B**ackup **B**rowser **L**ist) to distinguish it from the Browse List. The Consumer should keep track of the BB List so that any time it needs an updated copy of the Browse List it can query one of the Browsers on that list. That's how the workload is distributed in the Network Neighborhood.

Keeping in mind that Browse Service duties are cumulative, the LMB will probably include itself in the BB List. On small LANs there may not be any Backup Browsers hanging around, so the LMB may be the *only* Browser listed in the BB List.

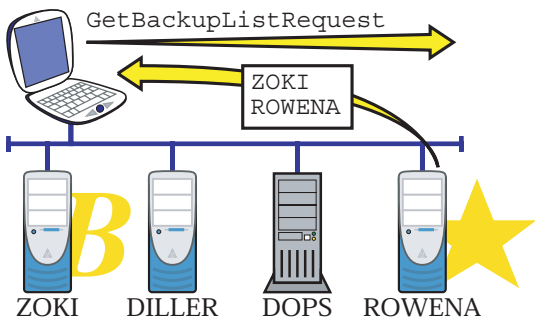

Figure 20.1: *Requesting the Backup Browser list*

Node ZOKI is a Backup Browser, and node ROWENA is the LMB. A Browse Service client broadcasts a request for the Backup Browser List (BB List), and the LMB responds (unicast) with a list of active browsers.

The request for the BB List is sent as a broadcast NBT datagram. The request message, as indicated in Figure 20.1, is known as a `GetBackupListRequest Browser Frame`. If the Consumer does not receive a response to the initial request, it should try again a couple of times. If no reply is received at all, the client *may* call for a new election — once only — and then try again to get a BB list. It may be the case that there are no Potential Browsers on the LAN at all, resulting in no LMB and no local Browse List. Continually calling for new elections in this situation would be futile (and rude).

Let's hope, however, that there is an LMB and that it does respond. The reply from an LMB is known as a `GetBackupListResponse Browser Frame`. It is *not* sent as a broadcast. Instead, the response is sent back to the requester in a unicast datagram (in NBT terminology, a `DIRECT UNIQUE DATAGRAM`).

...and that's what it takes to find out where copies of the Browse List are kept.

At this point in the proceedings the Consumer has obtained the NBT name of a Browser (either a Backup Browser or the LMB) and is ready to send a query to obtain the Browse List (see Figure 20.2).

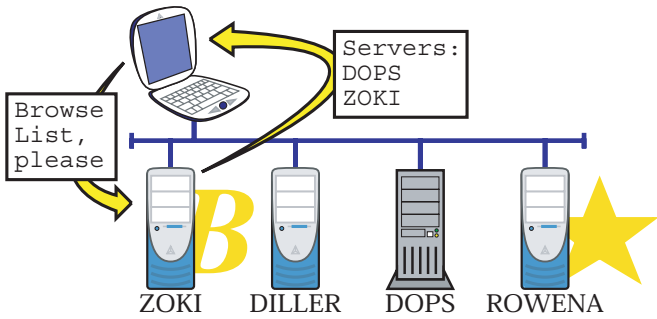

Figure 20.2: *Requesting the Browse List*

Earlier, the Consumer node learned that `ZOKI` had a copy of the Browse List. The Consumer uses the **R**emote **A**dministration **P**rotocol to request a copy of the Browse List.

In this example, nodes `ZOKI` and `DOPS` are Providers, advertising services via the Browse List.

This step is a little more complex than the previous ones. The Browse List might be very large, in which case (recalling the limitations of the NBT Datagram Service) an NBT datagram might not be big enough to hold it all. So instead of using the Datagram Service the Browse List query is sent using the **R**emote **A**dministration **P**rotocol (RAP) which rides on top of the `SMB_COM_TRANSACTION` message (aka SMBtrans). SMBtrans, in turn, allows for data transfers of up to 64K.

20.2 The Local Master Browser

It's time to meet your elected officials.

All Browser nodes register the *workgroup*`<1E>` NetBIOS name. The Local Master Browser, as you already know, identifies itself by registering two additional NetBIOS names: *workgroup*`<1D>` and `MSBROWSE<01>`. NetBIOS names represent communications end-points — services or applications that are using the NetBIOS API to listen for connections or messages. On the other side of these particular names there is software waiting to hear the chatter of the Browse Service.

The LMB has the following duties.

Maintaining the master copy of the workgroup Browse List for the local IP subnet

The LMB listens on the *workgroup*<1D> name for `HostAnnouncement` messages from members of its own workgroup. It also listens on the `MSBROWSE<01>` name so that it can hear `DomainAnnouncement Browser Frame` messages from other LMBs serving other workgroups on the same LAN (see Figure 20.3). The information collected from these announcements is used to build the Browse List.

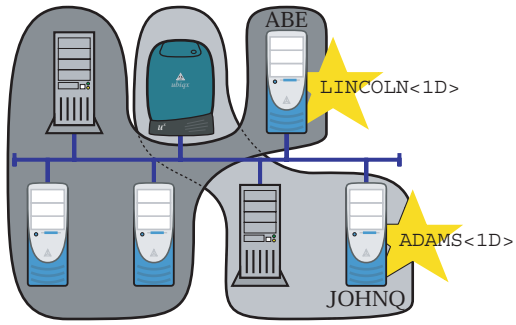

Figure 20.3: *Multiple workgroups on the same LAN*

Two workgroups, each with their own Local Master Browser, share the same IP subnet. Node ABE will collect the local Browse List for the `LINCOLN` workgroup, and will also listen for `DomainAnnouncement` messages from node `JOHNQ`.

Announcing its workgroup to other LMBs representing other workgroups

The LMB broadcasts `DomainAnnouncement` messages to the `MSBROWSE<01>` name to announce itself (and its workgroup) to the LMBs of other workgroups.

Announcing itself to its own workgroup

The LMB periodically sends a `LocalMasterAnnouncement Browser Frame` to the *workgroup*<1E> name. The Backup Browsers use this announcement to keep track of the LMB so that they can update their copies of the Browse List.

Delegating responsibility to Backup Browsers

The LMB is expected to promote Potential Browsers to Backup Browsers as the need arises. This is done by sending a `BecomeBackup Browser Frame` to the selected Browser node. The LMB should also provide the BB List in response to a `GetBackupListRequest`.

Coordinating with the Domain Master Browser

The LMB should query the NBNS for the *workgroup*`<1B>` name (which is registered by the DMB). There are two exceptions to this rule:

- B-mode nodes will not query the NBNS because remote LANs are outside of their scope and B nodes shouldn't talk to strangers.
- If the LMB is also the workgroup DMB then it doesn't need to query the NBNS to find itself.

Otherwise, the LMB will periodically announce its existence to the DMB by sending a `MasterAnnouncement Browser Frame`. Once the LMB and the DMB know about one another, they will periodically request a copy of each other's Browse Lists. That's how the lists get merged across subnets.

The LMB should also remember which Browse List entries were received from nodes on the local LAN, and which came from the DMB. The LMB is *authoritative* with regard to local nodes, but it must rely upon the DMB for everything else. If there are ever any conflicts, the LMB will favor its own local constituents.

Responding to requests for a copy of the Browse List

The LMB may receive queries for a copy of the Browse List from local Backup Browsers, the Domain Master Browser, or from everyday Consumers who simply want to display the Network Neighborhood.

Participating in Local Elections

Like all Browser nodes, the LMB listens on the *workgroup*`<1E>` name so that it can participate in local Browser Elections.

Quite the busy civil servant.

20.3 Becoming a Backup Browser

A Potential Browser becomes a Backup Browser in one of three ways.

1. If the number of Providers on the local LAN exceeds a predefined limit, the Local Master Browser will select an available Potential Browser and promote it to Backup Browser.

   In theory, the LMB will appoint new Backup Browsers as needed to prevent itself from getting overloaded. Some documentation says that there should be one Backup Browser for every 32 Providers on the local LAN. Other documentation says other things. In any case, the LMB can promote a Potential Browser by sending a `BecomeBackup Browser Frame`.

   The `BecomeBackup Browser Frame` is another NBT broadcast datagram. In this case, it is sent to the group name *workgroup*`<1E>`, which means that all of the Potential Browsers will receive the message. The NetBIOS machine name of the node to be promoted is carried within the message.

2. A Potential Browser may decide on its own to become a Backup Browser. It simply announces its new status to the LMB by sending out a `HostAnnouncement Browser Frame` with a specific flag set. (The flags will be described when we go into detail about the `NetServerEnum2` RAP call.)

   Note that the Backup Browser uses a `HostAnnouncement` to make itself known. That's the same thing a Provider does to announce services. In fact, the Backup Browser *is* announcing itself as a Provider: it provides access to the Browse List. This stuff gets mighty recursive at times.

   According to the Leach/Naik Browser Internet Draft, an LMB that loses a new election should demote itself to Backup Browser status instead of dropping all the way down to a Potential Browser. The theory is that it is the most likely to be promoted should something bad happen to the new LMB, so it should maintain a fairly up-to-date copy of the Browse List to ensure a smooth transition.

3. Browser roles are cumulative so, to be pedantic, the LMB is also a Backup Browser.

At the time that the Browse Service was created it may have been reasonable to be concerned about one computer bearing the brunt of all of the client requests, particularly on a LAN with a very large number of nodes. Today's computers are capable of handling the load *and* their own work without breaking a sweat. It would take an effort (a purposeful Denial of Service attack, for instance) to cause the LMB any real trouble.

20.4 Crossing the Street with the DMB

Browser roles are cumulative, as we keep saying, so the Domain Master Browser is also the Local Master Browser for its subnet and it must handle all of the duties of an LMB. One such duty is participation in local Browser Elections. Of course, since the DMB is the *appointed* workgroup president it is expected to win the election — which it will do because the election is rigged. More on that when we go into detail regarding the election process.

The DMB listens on the *workgroup*<1B> name for (unicast) `MasterAnnouncement` messages from Local Master Browsers on remote subnets. It keeps track of these announcements and, periodically, contacts the LMBs and asks for a new copy of their local Browse List. The DMB merges the local Browse Lists collected from the various LMBs (including its own) into a master Browse List for the entire workgroup. The LMBs, in turn, will periodically query the DMB and add the remote entries collected in the workgroup master list to their own local Browse Lists. That's how local LANs get a complete copy of the combined workgroup Browse List.

The key to making this all work is the NBT Name Service, the **N**et**B**IOS **N**ame **S**erver (NBNS) in particular. The scattered LMBs use the NBNS (aka WINS server) to find the *workgroup*<1B> name, which is registered by the DMB. Without that, cross-subnet browsing would not work because the LMBs would be unable to announce themselves to the DMB, and would also be unable to request copies of the DMB's master list.

Note that B mode NBT nodes do not talk to the NBNS and, therefore, cannot find a remote Domain Master Browser. That's okay, though, because the scope of a B mode NBT LAN is limited to the local IP subnet anyway. Even if a B node could do cross-subnet browsing, it wouldn't (shouldn't) be able to connect to a server on a remote subnet.

In contrast, P nodes *must* transact *all* of their Browse Service business directly with the Domain Master Browser. The NBT Scope available to a P node is the set of names it can resolve via the NBNS. It doesn't do broadcasts, so the only Browser node that it can find is the DMB because the DMB is the only Browser node with a name that can be properly resolved via the NBNS. M and H nodes have the best of both worlds. They can send broadcasts *and* use the NBNS to resolve names.

Now that you have a basic idea of how this stuff works, think about what might have happened if Microsoft had correctly implemented group name handling in their WINS implementation and had also provided a working **N**et**B**IOS **D**atagram **D**istribution Server (NBDD). If they had done that, the broadcast datagrams used by the Browse Service — the announcements and election requests and such — would have reached the entire extent of the virtual NetBIOS LAN even if it spanned multiple subnets, even across WAN links and such. For better or worse, that would have changed the design and workings of the Browse Service entirely.

20.5 Elections

Browser elections are fun to watch. They have more in common with the noise and chaos of a party convention than with an actual election. The process is something of a shouting match, and the winner is the last one left shouting at the end.

Starting an election is a lot like picking a fight. Some punk computer somewhere on the LAN sends out a challenge, called a `RequestElection Browser Frame`. The message lets all of the Potential Browsers on the LAN know how tough the sender thinks it is (see Figure 20.4). The Potential Browsers respond by broadcasting their own `RequestElection` messages, also declaring how tough they think they are. The Potential Browsers don't really want to fight about it, though, so when they hear a `RequestElection` message from a node that is tougher than they are, they shut up. The specifics of the election criteria will be covered when the we study the browser frames in detail.

Just to complete the fighting analogy, each transmission of a `Request-Election` message during a browser election is called a "round." There are typically four rounds because the eventual winner of the election will repeat

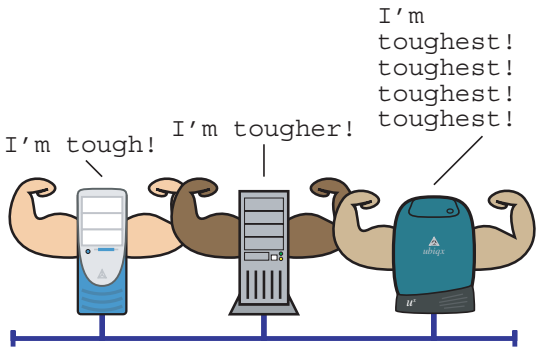

Figure 20.4: *Browser elections*

The Potential Browsers announce their strengths by sending a `RequestElection` message. The strongest candidate must repeat its announcement several times (typically four) to ensure that all challengers have acquiesced.

its `RequestElection` message four times to ensure that all of its challengers have given up. Once the winner is confident in its victory it sends a `LocalMasterAnnouncement Browser Frame`, which has two purposes. First, it lets all of the Backup Browsers know where to find the LMB. Second, the `LocalMasterAnnouncement` message announces the end of the election. Any further `RequestElection` messages heard on the wire will signal a new Browser Election.

Elections can be forced by sending an empty `RequestElection Browser Frame` — that is, one that announces the sender as being a complete wimp. All of the Potential Browsers on the LAN will have better credentials, so they will try to respond. Elections may be called when a Consumer can no longer find the LMB, or when a new node joins the workgroup and thinks that it has what it takes to be the LMB. When a Domain Master Browser starts up, for instance, it will always call for elections (since it *must* take over the role of LMB).

The `RequestElection` message is another NBT broadcast datagram. It is meant to be sent to the *workgroup*`<1E>` name but it turns out that many clients will accept this message if it is sent to the `MSBROWSE<01>` name as well, so you can actually cause all of the workgroups on a single subnet to hold elections at the same time.