

Full multicast support in PHP

Obtaining and preparing the source codes

Creating a working directory:

```
mkdir php-mc
cd php-mc/
```

Getting source codes:

```
wget http://cz.php.net/get/php-5.1.4.tar.bz2/from/this/mirror
```

Unpacking source codes:

```
tar xvfj php-5.1.4.tar.bz2
```

Preparing the setup with SOCKETS extension enabled

```
cd php-5.1.4
./configure --enable-sockets
```

The source codes for the sockets extension are here:

```
cd ext/sockets/
```

Back up the original source code for generating the patch:

```
cp sockets.c sockets.c.original
```

We create a “one-click” recompile tool:

```
cat >make.sh
#!/bin/sh
rm sockets.o
rm sockets.lo
cd ../..
make ext/sockets/sockets.lo
cd ext/sockets

chmod +x make.sh
```

Introducing multicast specific constants

The source code which defines constants in PHP will be the following:

```
/* Multicast */
REGISTER_LONG_CONSTANT("IPPROTO_IP",          IPPROTO_IP,
                       CONST_CS | CONST_PERSISTENT);
REGISTER_LONG_CONSTANT("IP_MULTICAST_LOOP",    IP_MULTICAST_LOOP,
                       CONST_CS | CONST_PERSISTENT);
REGISTER_LONG_CONSTANT("IP_MULTICAST_TTL",     IP_MULTICAST_TTL,
                       CONST_CS | CONST_PERSISTENT);
REGISTER_LONG_CONSTANT("IP_MULTICAST_IF",      IP_MULTICAST_IF,
                       CONST_CS | CONST_PERSISTENT);
REGISTER_LONG_CONSTANT("IP_ADD_MEMBERSHIP",    IP_ADD_MEMBERSHIP,
                       CONST_CS | CONST_PERSISTENT);
REGISTER_LONG_CONSTANT("IP_DROP_MEMBERSHIP",   IP_DROP_MEMBERSHIP,
                       CONST_CS | CONST_PERSISTENT);
```

Code which enables the use of multicast

All the operations are done by two functions:

- `socket_set_option`
- `socket_get_option`.

Modifications in `socket_set_option`

A local variable of *unsigned char* type is introduced (`ov_uc` - option value unsigned char) and also an internet address structure and IP multicast request is required by some options:

```
u_char          ov_u_char;
struct          in_addr ov_in_addr;
struct          ip_mreq ov_ip_mreq;
```

The options will use an array which needs to be decoded, into the following variables:

```
zval            **multiaddr, **interface;
```

For the decoding a pair of new key name constants is defined:

```
char            *multiaddr_key = "multiaddr";
char            *interface_key = "interface";
```

IP_MULTICAST_LOOP

The original code from [1]:

```
u_char loop;
setsockopt(socket, IPPROTO_IP, IP_MULTICAST_LOOP, &loop, sizeof(loop));
```

This option has a value 1 or 0, so a comparison is made to convert *int* into *unsigned char*:

```
case IP_MULTICAST_LOOP:
    convert_to_long_ex(&arg4);
    ov_u_char = Z_LVAL_P(arg4) != 0;

    optlen = sizeof(ov_u_char);
    opt_ptr = &ov_u_char;
    break;
```

IP_MULTICAST_TTL

The original code from [1]:

```
u_char ttl;
setsockopt(socket, IPPROTO_IP, IP_MULTICAST_TTL, &ttl, sizeof(ttl));
```

The option has 0 to 255 as the value, so a conversion is made to apply the boundaries:

```
case IP_MULTICAST_TTL:
    convert_to_long_ex(&arg4);
    ov = Z_LVAL_P(arg4);
    if (ov < 0) ov = 0;
    if (ov > 255) ov = 255;
    ov_u_char = ov & 0xFF;

    optlen = sizeof(ov_u_char);
    opt_ptr = &ov_u_char;
    break;
```

IP_MULTICAST_IF

The original code from [1]:

```
struct in_addr interface_addr;
setsockopt (socket, IPPROTO_IP, IP_MULTICAST_IF, &interface_addr,
            sizeof(interface_addr));
```

In PHP, we need to convert the argument to a string, and then to *inet_addr* structure:

```
case IP_MULTICAST_IF:
    convert_to_string_ex(&arg4);
    if (!inet_aton(Z_STRVAL_PP(&arg4), &ov_in_addr)) {
        php_error_docref(NULL TSRMLS_CC, E_WARNING,
            "could not convert optval to inet_addr" );
        RETURN_FALSE;
    }

    optlen = sizeof(ov_in_addr);
    opt_ptr = &ov_in_addr;
    break;
```

IP_ADD_MEMBERSHIP

IP_DROP_MEMBERSHIP

The original code from [1]:

```
struct ip_mreq mreq;
setsockopt (socket, IPPROTO_IP, IP_DROP_MEMBERSHIP, &mreq, sizeof(mreq));
```

Code in the PHP expects an array option value, which has at least the *multiaddr* key. If a second key is defined (as *interface*), it is used in the multicast request, otherwise a wildcard address INADDR_ANY is used.

```
case IP_DROP_MEMBERSHIP:
    convert_to_array_ex(&arg4);
    opt_ht = HASH_OF(arg4);

    if (zend_hash_find(opt_ht, multiaddr_key, strlen(multiaddr_key) + 1,
        (void **)&multiaddr) == FAILURE) {
        php_error_docref(NULL TSRMLS_CC, E_WARNING,
            "no key \"%s\" passed in optval", multiaddr_key);
        RETURN_FALSE;
    }

    convert_to_string_ex(multiaddr);

    if (!inet_aton(Z_STRVAL_PP(multiaddr), &(ov_ip_mreq.imr_multiaddr))) {
        php_error_docref(NULL TSRMLS_CC, E_WARNING,
            "could not convert optval[multiaddr] to inet_addr" );
        RETURN_FALSE;
    }

    if (zend_hash_find(opt_ht, interface_key, strlen(interface_key) + 1,
        (void **)&interface) == FAILURE) {
        ov_ip_mreq.imr_interface.s_addr = htonl(INADDR_ANY);
    } else {
        convert_to_string_ex(interface);

        if (!inet_aton(Z_STRVAL_PP(interface), &(ov_ip_mreq.imr_interface))) {
            php_error_docref(NULL TSRMLS_CC, E_WARNING,
                "could not convert optval[interface] to inet_addr" );
            RETURN_FALSE;
        }
    }

    optlen = sizeof(ov_ip_mreq);
    opt_ptr = &ov_ip_mreq;
    break;
```

Modifications in `socket_get_option`

Some local variables were introduced:

```
u_char          other_val_u_char;
struct in_addr  other_val_in_addr;
char*           other_val_string;
```

IP_MULTICAST_LOOP

IP_MULTICAST_TTL

The original code from [1]:

```
u_char loop;
int size;

getsockopt(socket, IPPROTO_IP, IP_MULTICAST_LOOP, &loop, &size)
```

These options return both a byte, so the code to handle them is only one:

```
case IP_MULTICAST_LOOP:
case IP_MULTICAST_TTL:
    optlen = sizeof(other_val_u_char);

    if (getsockopt/php_sock->bsd_socket, level, optname,
        (char*)&other_val_u_char, &optlen) != 0) {
        PHP_SOCKET_ERROR/php_sock, "unable to retrieve socket option", errno);
        RETURN_FALSE;
    }

    RETURN_LONG(other_val_u_char);
```

IP_MULTICAST_IF

The code gets the internet address, converts it to a string and returns it:

```
case IP_MULTICAST_IF:
    optlen = sizeof(other_val_in_addr);

    if (getsockopt/php_sock->bsd_socket, level, optname,
        (char*)&other_val_in_addr, &optlen) != 0) {
        PHP_SOCKET_ERROR/php_sock, "unable to retrieve socket option", errno);
        RETURN_FALSE;
    }

    other_val_string = inet_ntoa(other_val_in_addr);

    RETURN_STRING(other_val_string, 1);
```

Finishing

Now, when the modifications are made, we run our `make.sh` script:

```
./make.sh
/bin/sh /home/daniel/php-mc/php-5.1.4/libtool --silent --preserve-dup-deps --
mode=compile gcc -Iext/sockets/ -I/home/daniel/php-mc/php-5.1.4/ext/sockets/
-DPHP_ATOM_INC -I/home/daniel/php-mc/php-5.1.4/include -I/home/daniel/php-
mc/php-5.1.4/main -I/home/daniel/php-mc/php-5.1.4 -I/usr/include/libxml2 -
I/home/daniel/php-mc/php-5.1.4/ext/date/lib -I/home/daniel/php-mc/php-
5.1.4/TSRM -I/home/daniel/php-mc/php-5.1.4/Zend -I/usr/include -g -O2 -c
/home/daniel/php-mc/php-5.1.4/ext/sockets/sockets.c -o ext/sockets/sockets.lo
```

If the compilation doesn't show any error, the extension will compile without problems.

Creating a patch

To simplify the use of the modifications, we create a patch. This is done by moving the edited file to another directory:

```
cd ~/php-mc

mkdir -p php-5.1.4-mc/ext/sockets
mv php-5.1.4/ext/sockets/sockets.c php-5.1.4-mc/ext/sockets/sockets.c
```

We delete the old sources, which are already compiled and unpack them again:

```
rm -rf php-5.1.4
tar xvfj php-5.1.4.tar.bz2
```

To create a patch, a *diff* command has to be used (-u means unified format):

```
diff -u php-5.1.4/ext/sockets/sockets.c php-5.1.4-mc/ext/sockets/sockets.c
>php5-ext-sockets-multicast.patch
```

We can publish the patch on the web:

```
mkdir -p ~/www/diary.rozsnyo.com/2006/06/16/
mv php5-ext-sockets-multicast.patch ~/www/diary.rozsnyo.com/2006/06/16/
```

The patch is now located under the following URL:

```
http://diary.rozsnyo.com/2006/06/16/php5-ext-sockets-multicast.patch
```

Installing modified PHP in Gentoo Linux

We will manually patch the unpacked sources, but firstly download the patch:

```
su -
wget http://diary.rozsnyo.com/2006/06/16/php5-ext-sockets-multicast.patch
```

Start the installation until unpacking:

```
ebuild /usr/portage/dev-lang/php/php-5.1.4-r1.ebuild fetch unpack
```

Patch the modified file:

```
cd /var/tmp/portage/php-5.1.4-r1/work/php-5.1.4/
patch -p1 </root/php5-ext-sockets-multicast.patch
```

Finish the installation:

```
cd
ebuild /usr/portage/dev-lang/php/php-5.1.4-r1.ebuild compile install qmerge
```

References

- [1] Multicast programming
<http://www.tldp.org/HOWTO/Multicast-HOWTO-6.html>
- [2] Multicast Example Programs
<http://ntrg.cs.tcd.ie/undergrad/4ba2/multicast/antony/example.html>