NAME

Array::IntSpan::IP – a Module for arrays using IP addresses as indices

SYNOPSIS

```
use Array::IntSpan::IP;

my $foo = Array::IntSpan::IP->new(['123.45.67.0',   '123.45.67.255', 'Network 1'],
                                  ['123.45.68.0',   '123.45.68.127', 'Network 2'],
                                  ['123.45.68.128', '123.45.68.255', 'Network 3'])

print "The address 123.45.68.37 is on network ".$foo->lookup("\173\105\150\45").".
unless (defined($foo->lookup(((123*256+45)*256+65)*256+67))) {
  print "The address 123.45.65.67 is not on a known network.\n";
}

print "The address 123.45.68.177 is on network ".$foo->lookup("123.45.68.177").".\

$foo->set_range('123.45.68.128', '123.45.68.255', 'Network 4');
print "The address 123.45.68.177 is now on network ".$foo->lookup("123.45.68.177")
```

DESCRIPTION

`Array::IntSpan::IP` brings the advantages of `Array::IntSpan` to IP address indices. Anywhere you use an index in `Array::IntSpan`, you can use an IP address in one of three forms in `Array::IntSpan::IP`. The three accepted forms are:

Dotted decimal
: This is the standard human-readable format for IP addresses. The conversion checks that the octets are in the range 0–255. Example: `'123.45.67.89'`.

Network string
: A four character string representing the octets in network order. Example: `"\173\105\150\131"`.

Integer
: A integer value representing the IP address. Example: `((123*256+45)*256+67)*256+89` or `2066563929`.

Note that the algorithm has no way of distinguishing between the integer values 1000 through 9999 and the network string format. It will presume network string format in these instances. For instance, the integer `1234` (representing the address `'0.0.4.210'`) will be interpreted as `"\61\62\63\64"`, or the IP address `'49.50.51.52'`. This is unavoidable since Perl does not strongly type integers and strings separately and there is no other information available to distinguish between the two in this situation. I do not expect that this will be a problem in most situations. Most users will probably use dotted decimal or network string notations, and even if they do use the integer notation the likelyhood that they will be using the addresses `'0.0.3.232'` through `'0.0.39.15'` as indices is relatively low.

METHODS

ip_as_int

The class method `Array::IntSpan::IP::ip_as_int` takes as its one parameter the IP address in one of the three formats mentioned above and returns the integer notation.

AUTHOR

Toby Everett, teverett@alascom.att.com