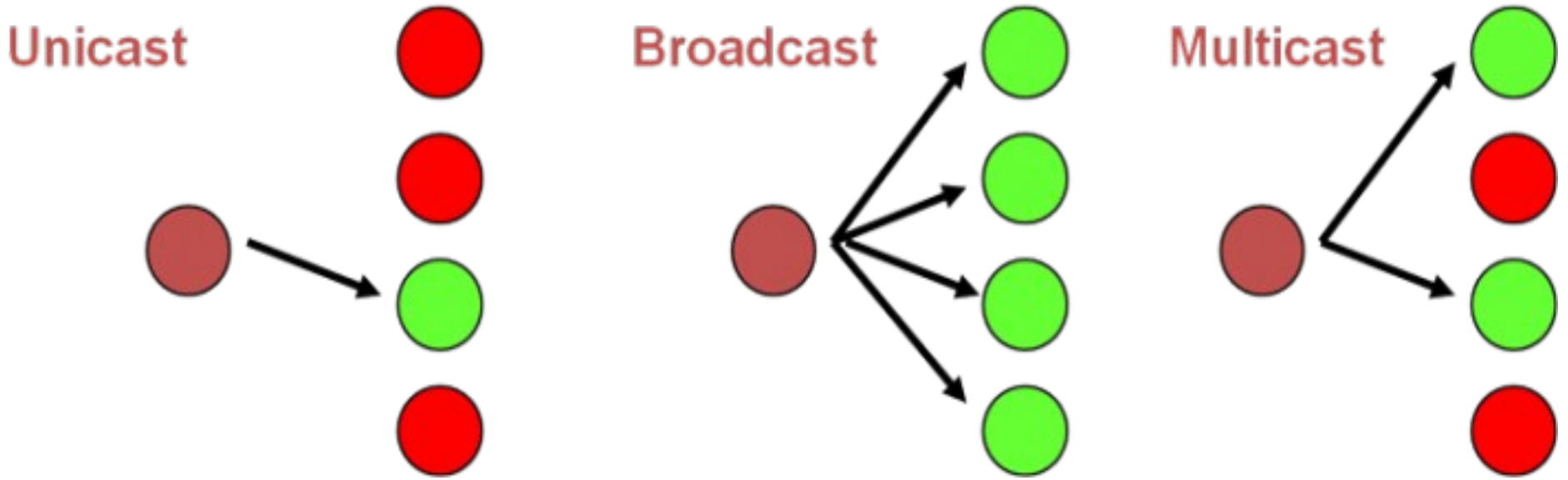
Network Programming

Pemrograman sistem dan jaringan

Henry Saptono
2019

Methods of Communication



UDP multicast communication

- Koneksi point-to-point menangani banyak kebutuhan komunikasi, tetapi memberikan informasi yang sama ke banyak komputer menjadi tantangan karena jumlah koneksi langsung bertambah.
- Mengirim pesan secara terpisah ke setiap penerima menghabiskan waktu pemrosesan dan bandwidth tambahan, yang dapat menjadi masalah untuk aplikasi seperti streaming video atau audio.
- Menggunakan multicast untuk mengirimkan pesan ke lebih dari satu titik akhir sekaligus mencapai efisiensi yang lebih baik karena infrastruktur jaringan memastikan bahwa paket dikirimkan ke semua penerima.

UDP multicast communication

- Pesan multicast selalu dikirim menggunakan UDP, karena TCP memerlukan saluran komunikasi end-to-end.
- Alamat untuk multicast, disebut grup multicast, adalah bagian dari kisaran alamat IPv4 reguler (**224.0.0.0** hingga **230.255.255.255**) yang disediakan untuk lalu lintas multicast.
- Alamat multicast diperlakukan secara khusus oleh router dan switch jaringan, sehingga pesan yang dikirim ke grup dapat didistribusikan melalui Internet ke semua penerima yang telah bergabung dengan grup.

Sending Multicast Messages

- Program echo-client yang dimodifikasi ini akan mengirim pesan ke grup multicast, lalu melaporkan semua respons yang diterimanya. Karena tidak mengetahui berapa banyak respons yang diharapkan, ia menggunakan nilai batas waktu pada soket untuk menghindari kondisi blocking (menunggu jawaban) tanpa batas

Echo-client UDP Multicast

```
import socket
import struct
import sys
message = 'very important data'
multicast_group = ('224.3.29.71', 10000)


# Create the datagram socket
sock = socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)
# Set a timeout so the socket does not block
indefinitely when trying
# to receive data.
sock.settimeout(0.2)
```

Echo-client UDP Multicast

```
#Set the time-to-live for messages to 1
#so they do not go past the
#local network segment.
ttl = struct.pack('b', 1)
sock.setsockopt(socket.IPPROTO_IP,
socket.IP_MULTICAST_TTL, ttl)
```

Echo-client UDP Multicast

```
try:
    # Send data to the multicast group
    print('sending "%s"' % message)
    sent = sock.sendto(message.encode(), multicast_group)

    # Look for responses from all recipients
    while True:
        print('waiting to receive')
        try:
            data, server = sock.recvfrom(16)
        except socket.timeout:
            print('timed out, no more responses')
            break
        else:
            print('received "%s" from %s' % (data, server))
```

Echo-client UDP Multicast

```
finally:
    print('closing socket')
    sock.close()
```

Receiving Multicast Messages

- Langkah pertama untuk membuat penerima multicast adalah membuat soket UDP.

```
import socket

import struct

import sys

multicast_group = '224.3.29.71'

server_address = ('', 10000)

# Create the socket

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Bind to the server address

sock.bind(server_address)
```

Receiving Multicast Messages

- Setelah soket biasa dibuat dan terikat ke port, itu dapat ditambahkan ke grup multicast dengan menggunakan **setsockopt ()** untuk mengubah opsi **IP_ADD_MEMBERSHIP**.
- Nilai opsi adalah representasi **8 byte** yang dikemas dari alamat grup multicast diikuti oleh antarmuka jaringan tempat server harus mendengarkan lalu lintas, yang diidentifikasi oleh alamat **IP**-nya. Dalam hal ini, penerima mendengarkan semua antarmuka menggunakan **INADDR_ANY**.

```
# Tell the operating system to add the socket to the
multicast group

# on all interfaces.

group = socket.inet_aton(multicast_group)

mreq = struct.pack('4sL', group, socket.INADDR_ANY)

sock.setsockopt(socket.IPPROTO_IP,
socket.IP_ADD_MEMBERSHIP, mreq)
```

Receiving Multicast Messages

- Loop utama untuk server penerima

```
# Receive/respond loop
while True:
    print('\nwaiting to receive message')
    data, address = sock.recvfrom(1024)

    print('received %s bytes from %s' % (len(data),
address))
    print(data)

    print('sending acknowledgement to', address)
    sock.sendto(b'ack', address)
```

Contoh Output

- Contoh ini menunjukkan penerima multicast yang berjalan pada dua host yang berbeda, A memiliki alamat 192.168.1.17 dan B memiliki alamat 192.168.1.8

```
[A]$ python ./socket_multicast_receiver.py
waiting to receive message
received 19 bytes from ('192.168.1.17', 51382)
very important data
sending acknowledgement to ('192.168.1.17', 51382)

[B]$ python ./socket_multicast_receiver.py
waiting to receive message
received 19 bytes from ('192.168.1.17', 51382)
very important data
sending acknowledgement to ('192.168.1.17', 51382)
```

Contoh output

- Pengirim berjalan pada host A. Pesan dikirim satu kali, dan dua ucapan terima kasih dari pesan yang telah terkirim diterima, masing-masing dari host A dan B.

```
[A]$ python ./socket_multicast_sender.py


sending "very important data"
waiting to receive
received "ack" from ('192.168.1.17', 10000)
waiting to receive
received "ack" from ('192.168.1.8', 10000)
waiting to receive
timed out, no more responses
closing socket
```