Fakultas Informatika
School of Computing
Telkom University

CSG3L3
SISTEM TERDISTRIBUSI

Topik 6 : *Indirect Communication*

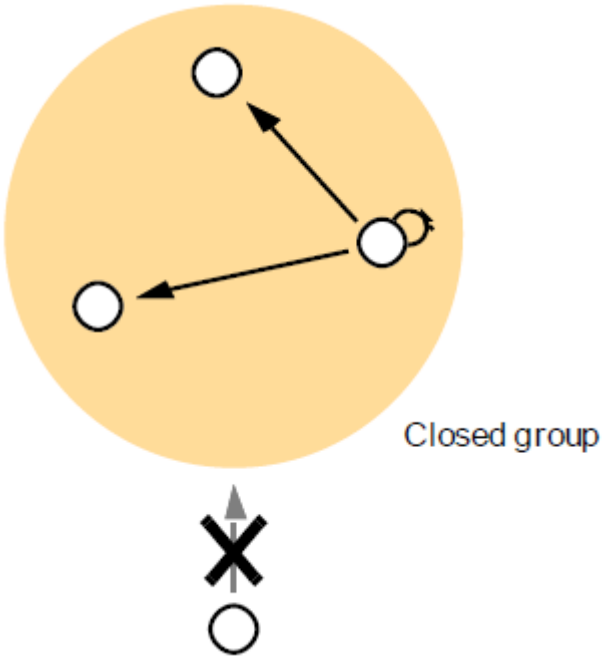
Matriks *Space* dan *Time Coupling* pada Sistem Terdistribusi

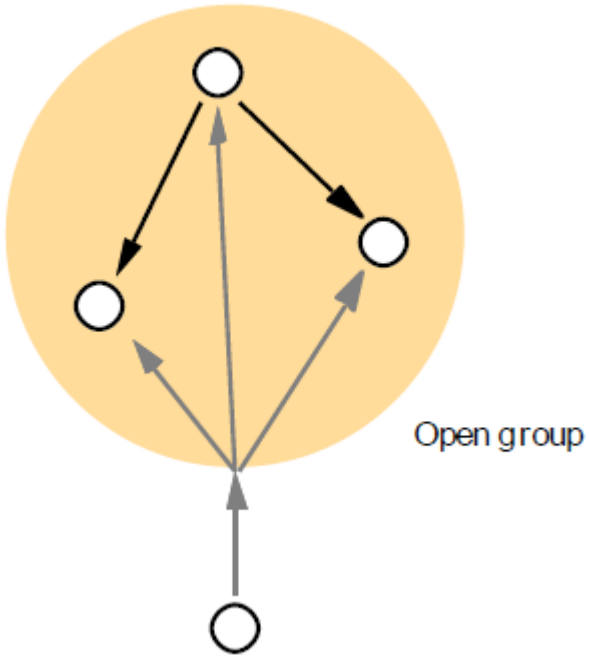
	Time-coupled	***Time-uncoupled***
Space-coupled	**Karaktersitik:** Komunikasi secara langsung kepada satu atau lebih penerima (*receiver*). *Receiver* harus ada (secara fisik) pada saat melakukan komunikasi (contoh: *message passing* & *remote invocation*)	**Karaktersitik:** Komunikasi langsung tertuju kepada satu atau lebih *receiver* namun komunikasi dapat dilakukan pada saat yang berbeda.

Matriks *Space* dan *Time Coupling* pada Sistem Terdistribusi (cont'd)

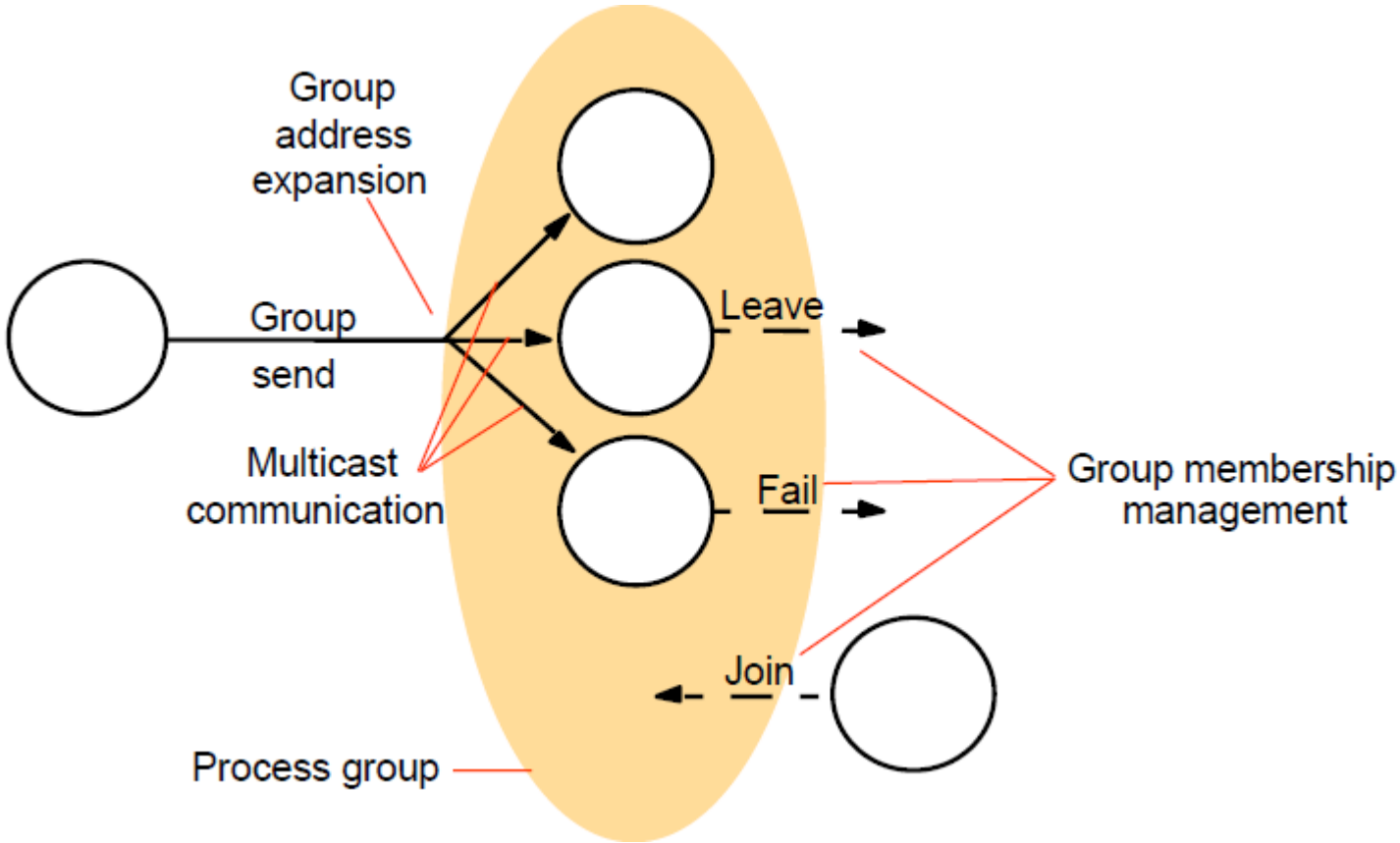
	Time-coupled	***Time-uncoupled***
Space-uncoupled	**Karaktersitik:** Pengirim (*sender*) tidak perlu mengetahui identitas penerima (*receiver*), penerima harus ada (secara fisik) saat berkomunikasi. (Contoh: Multicast)	**Karaktersitik:** Pengirim (*sender*) tidak perlu mengetahui identitas penerima (*receiver*). Pengirim dan penerima berkomunikasi di waktu yang berbeda

Perbedaan Antara *Close Group* dan *Open Group*

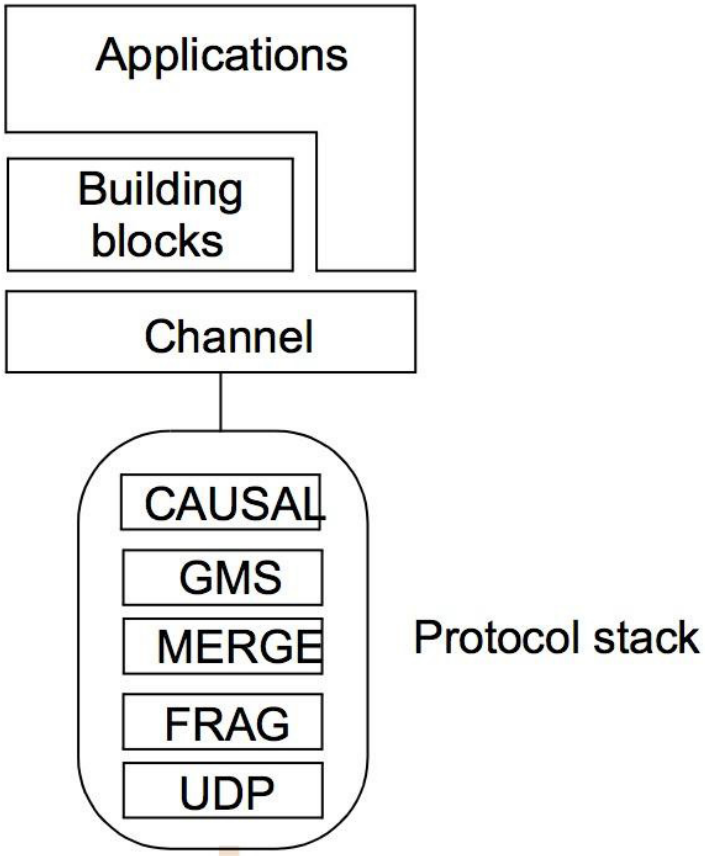




Status / Peran pada *Group Membership Management*



Arsitektur dari *JGroup*



Contoh Implementasi Grup di Java Menggunakan JGroups

```
import org.jgroups.JChannel;

public class FireAlarm {

    public void raise() {

        try {

            JChannel ch = new JChannel();

            ch.connect("AlarmChannel");

            Message m = new Message(null, null, "Fire!");
```

Contoh Implementasi Grup di Java Menggunakan Jgroups (cont'd)

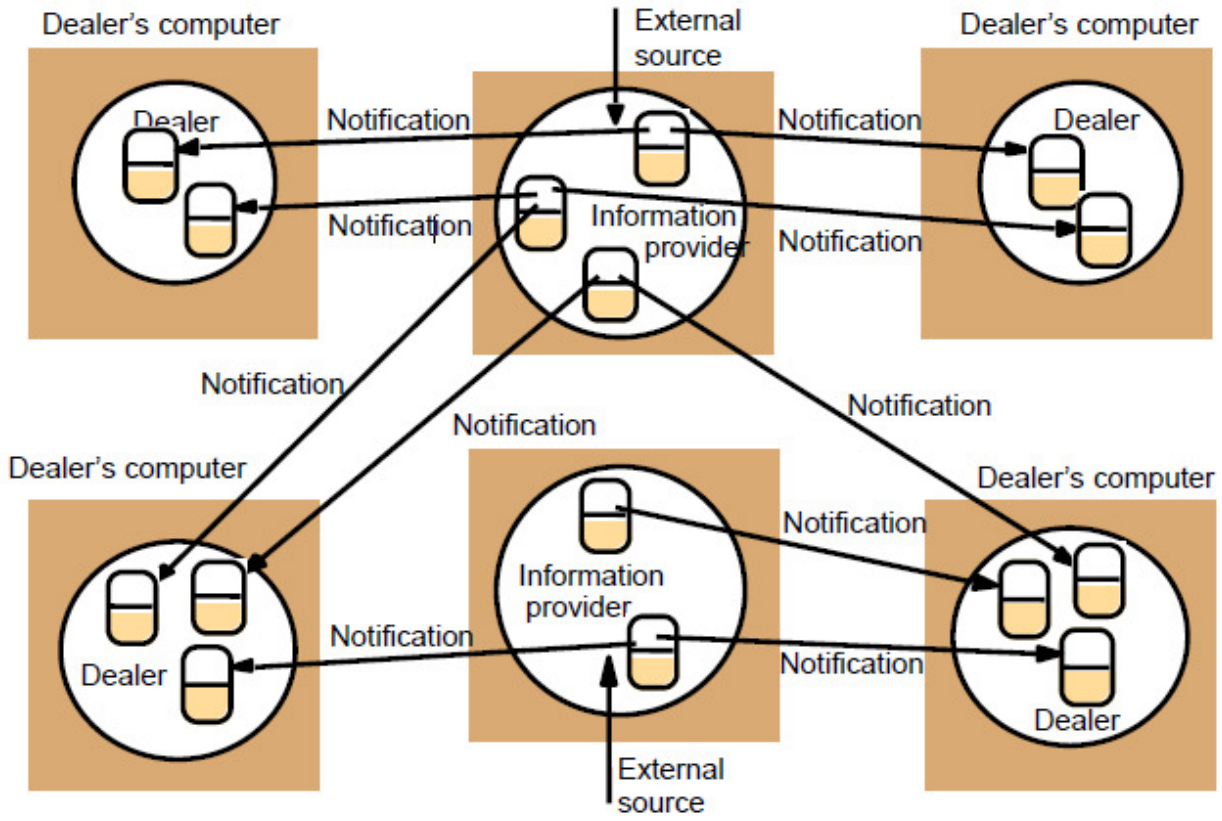
```
            ch.send(m);
        }
        catch(Exception e) {}
    }
}
```

Class FireAlarmConsumer

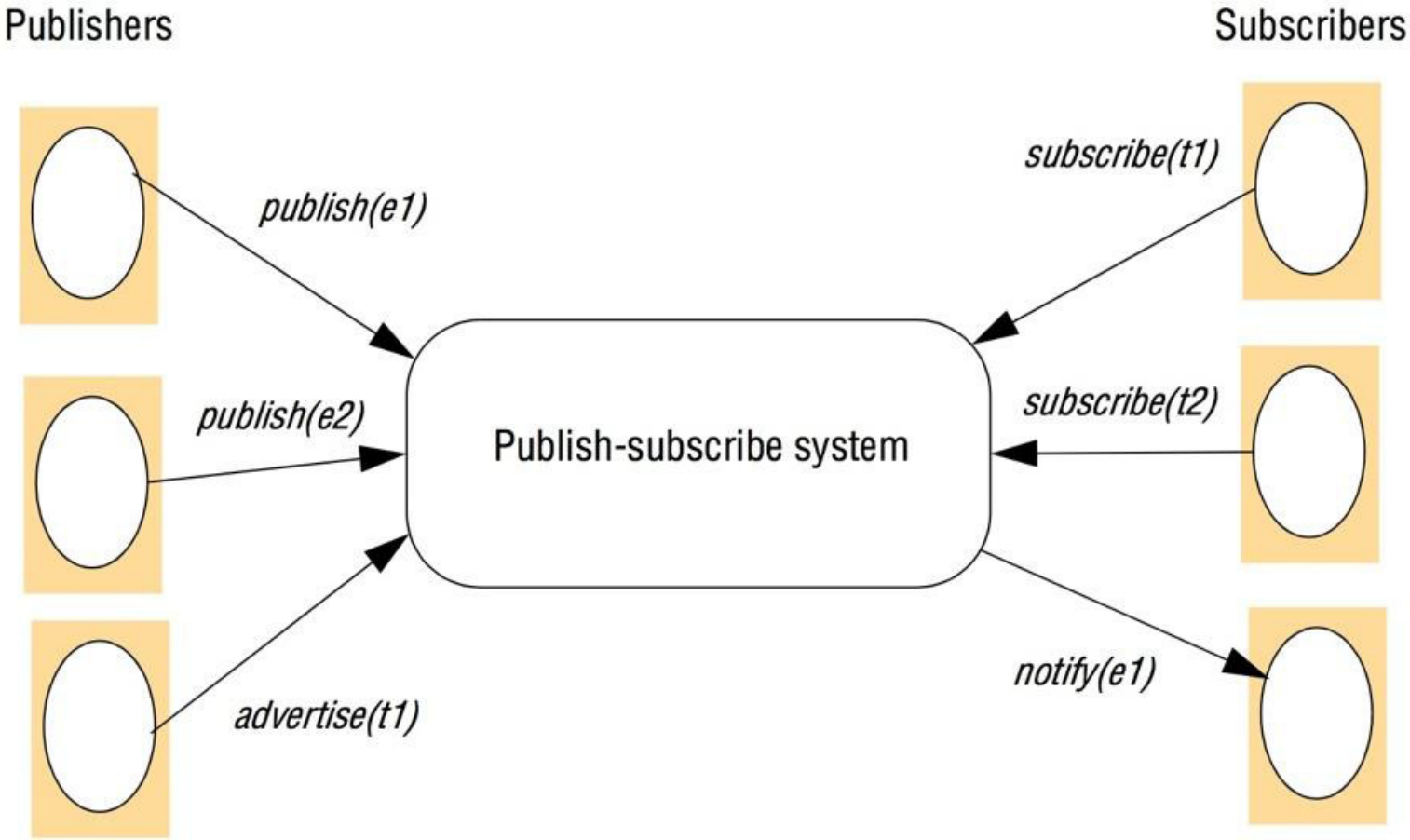
```
import org.jgroups.JChannel;

public class FireAlarmConsumerJG {
    public String await() {
    try {
            JChannel channel = new JChannel();
            channel.connect("AlarmChannel");
            Message msg = (Message) channel.receive(0);
            return (String) msg.GetObject();
    } catch(Exception e) {
            return null;
    }
  }
}
```

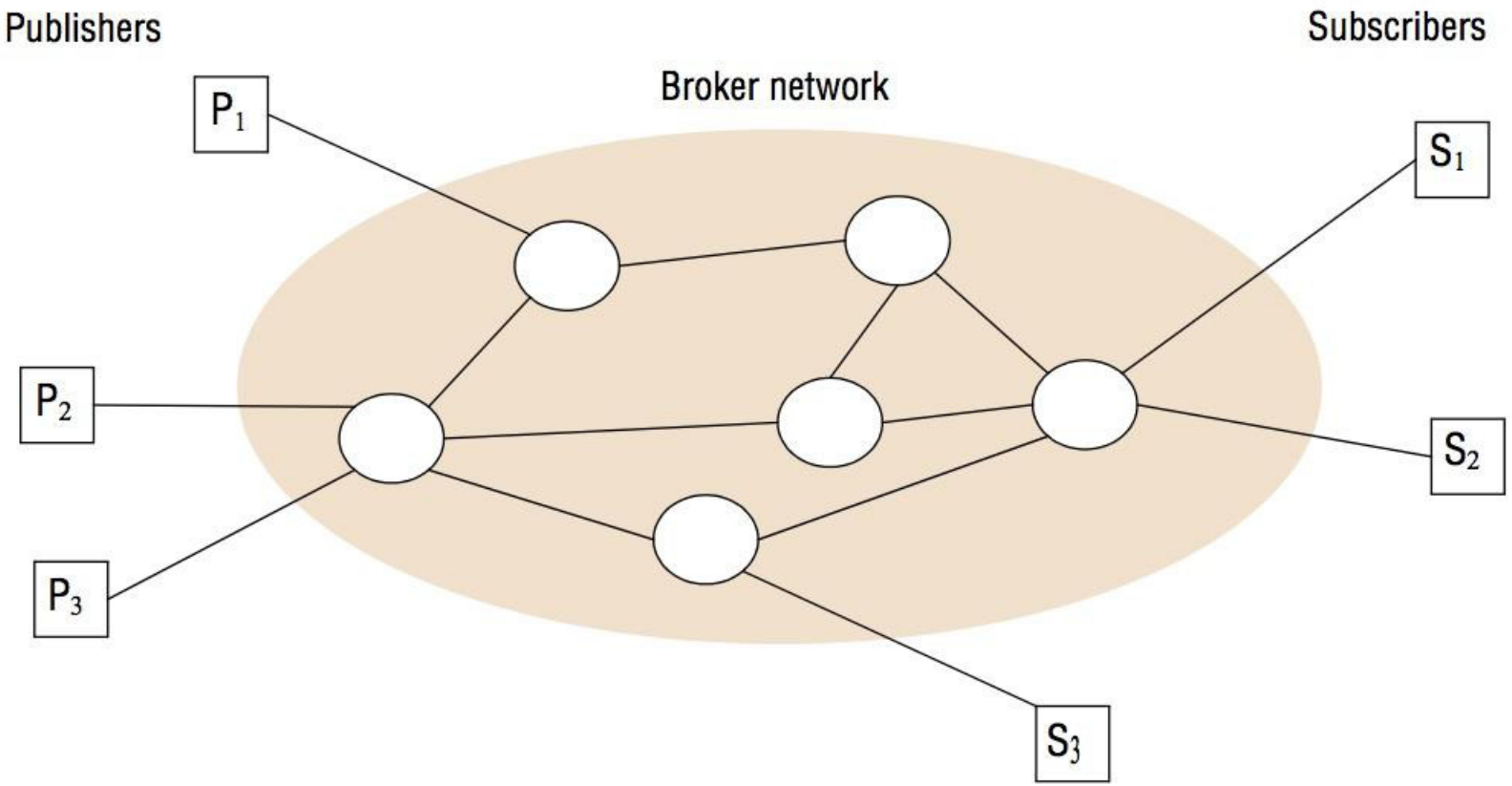
Bisakah Anda Menjelaskan Proses yang Terjadi pada Ilustrasi di Bawah Ini?

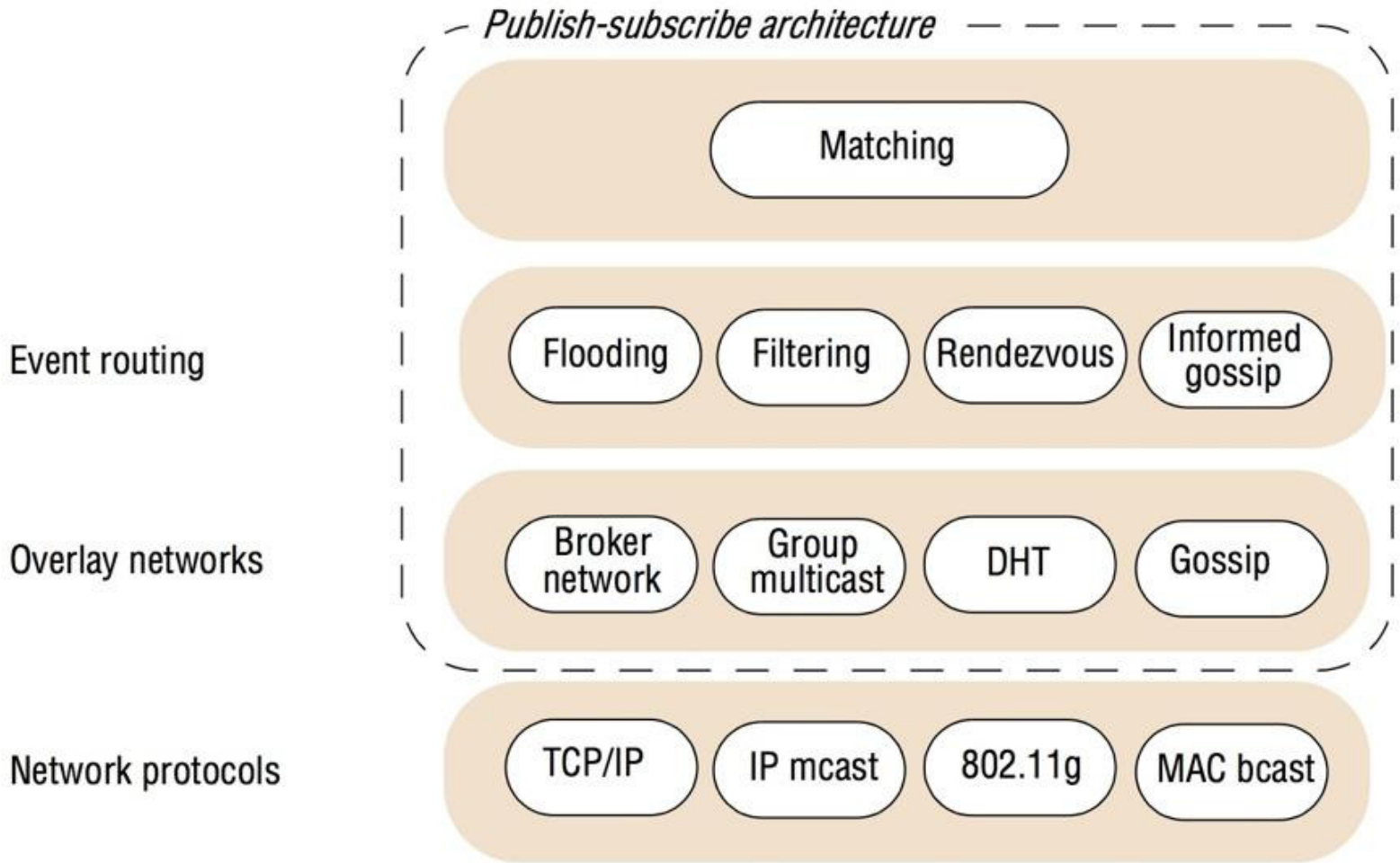


Paradigma *Publish-Subscribe*



Jaringan *Broker* pada Sistem *Publish-Subscribe*




Publish-subscribe architecture
Matching
Event routing
Flooding
Filtering
Rendezvous
Informed gossip
Overlay networks
Broker network
Group multicast
DHT
Gossip
Network protocols
TCP/IP
IP mcast
802.11g
MAC bcast

Routing Berbasis *Filtering*

```
upon receive publish(event e) from node x                    1
    matchlist := match(e, subscriptions)                     2
    send notify(e) to matchlist;                             3
    fwdlist := match(e, routing);   4
    send publish(e) to fwdlist - x;  5
upon receive subscribe(subscription s) from node x           6
    if x is client then   7
        add x to subscriptions;   8
    else add(x, s) to routing; 9
    send subscribe(s) to neighbours - x;        10
```

Routing berbasis *Rendezvous*

```
upon receive publish(event e) from node x at node i
    rvlist := EN(e);
    if i in rvlist then begin
        matchlist :=match(e, subscriptions);
        send notify(e) to matchlist;
    end
    send publish(e) to rvlist - i;
upon receive subscribe(subscription s) from node x at node i
    rvlist := SN(s);
    if i in rvlist then
        add s to subscriptions;
    else
        send subscribe(s) to rvlist - i;
```

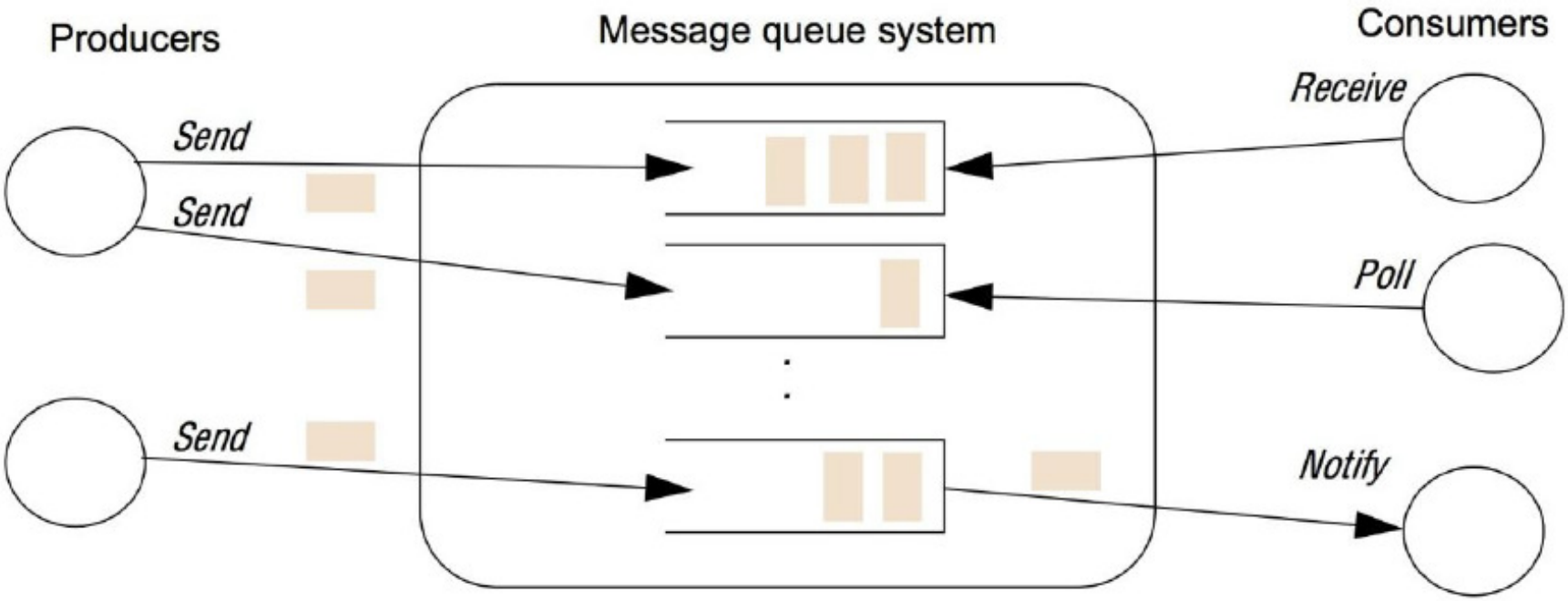
Contoh Sistem *Publish-Subscribe*

Sistem	Model *Subscription*	Model Distribusi	Routing Event
CORBA *Event Service* (bab 8)	Channel-based	Tersentralisasi	-
TIB Rendezvous	Topic-based	Terdistribusi	Filtering
Scribe	Topic-based	P2P (DHT)	Rendezvous
TERA	Topic-based	Peer-to-peer	Informed gossip
Siena	Content-based	Terdistribusi	Filtering
Gryphon	Content-based	Terdistribusi	Filtering
Hermes	Topic and content-based	Terdistribusi	Rendezvous & filtering

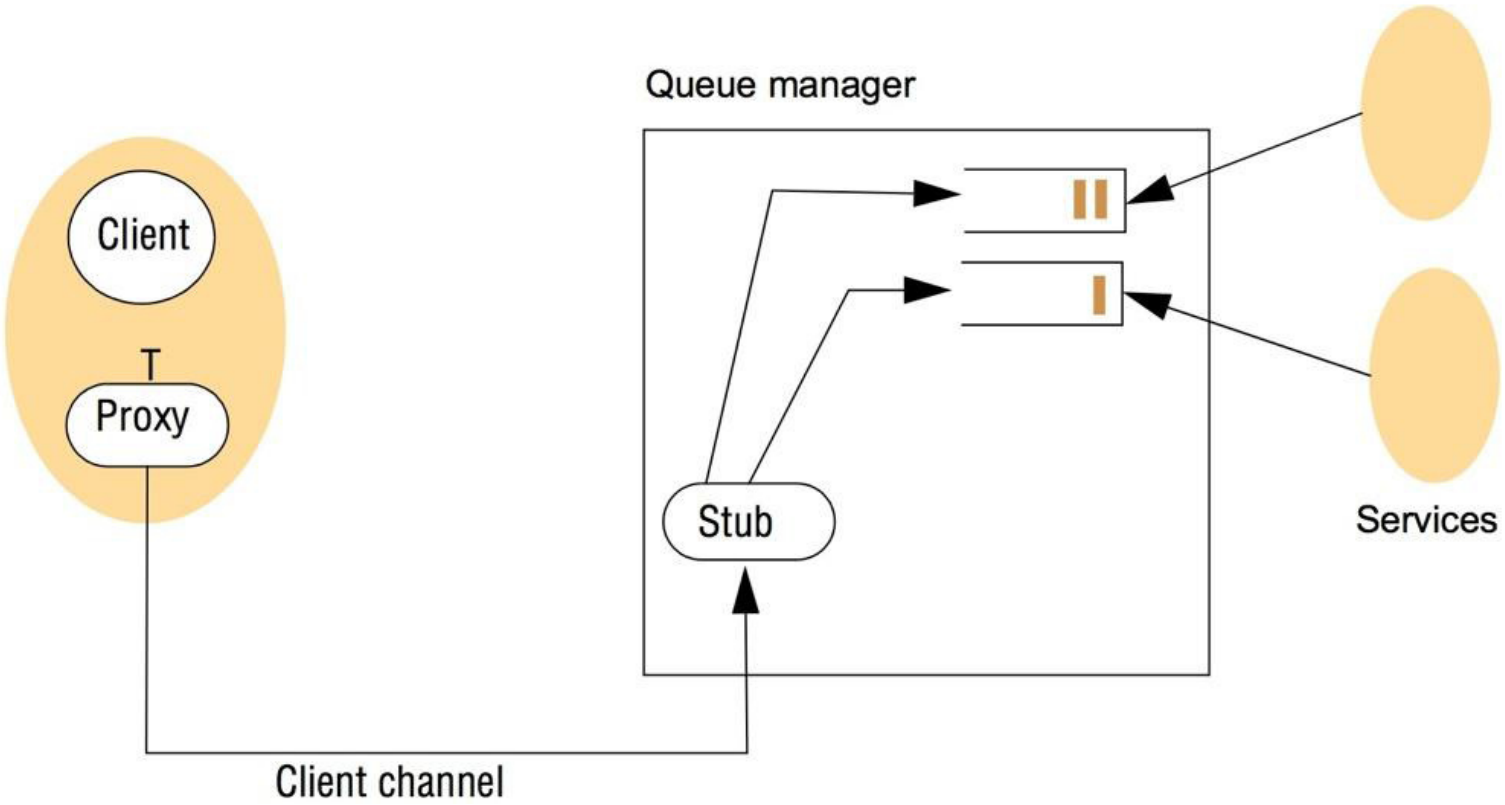
Contoh Sistem *Publish-Subscribe* (cont'd)

Sistem	Model *Subscription*	Model Distribusi	Routing Event
MEDYM	Content-based	Tersentralisasi	Flooding
Meghdoot	Content-based	Peer-to-peer	Rendezvous
Structure-less CBR	Content-based	Peer-to-peer	Informed gossip

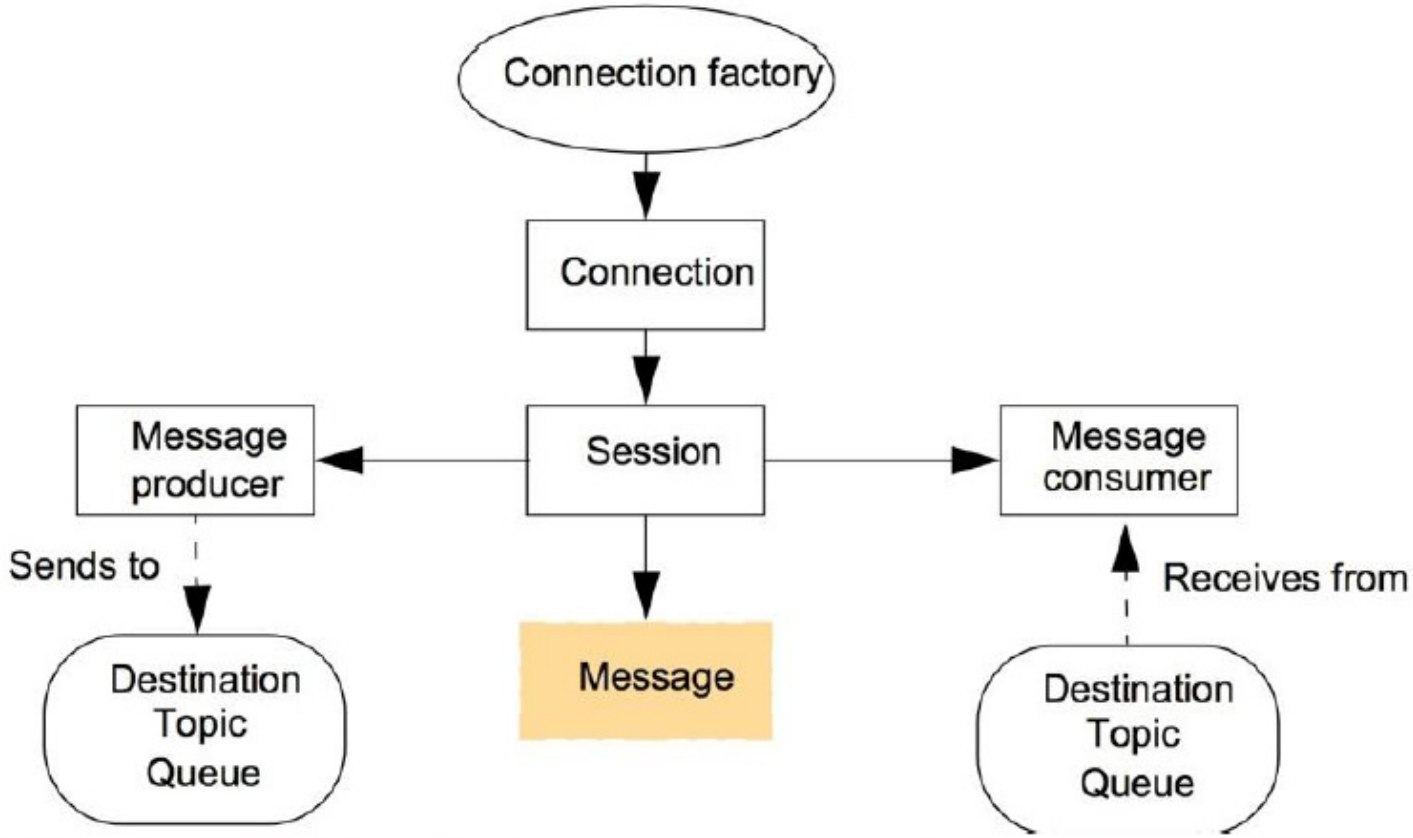
Paradigma *Message Queue*



Sebuah Topologi Jaringan Sederhana dari WebSphere MQ



Model Pemrograman Untuk *Message Queue* Menggunakan JMS



Contoh Implementasi JMS pada Kasus FireAlarm

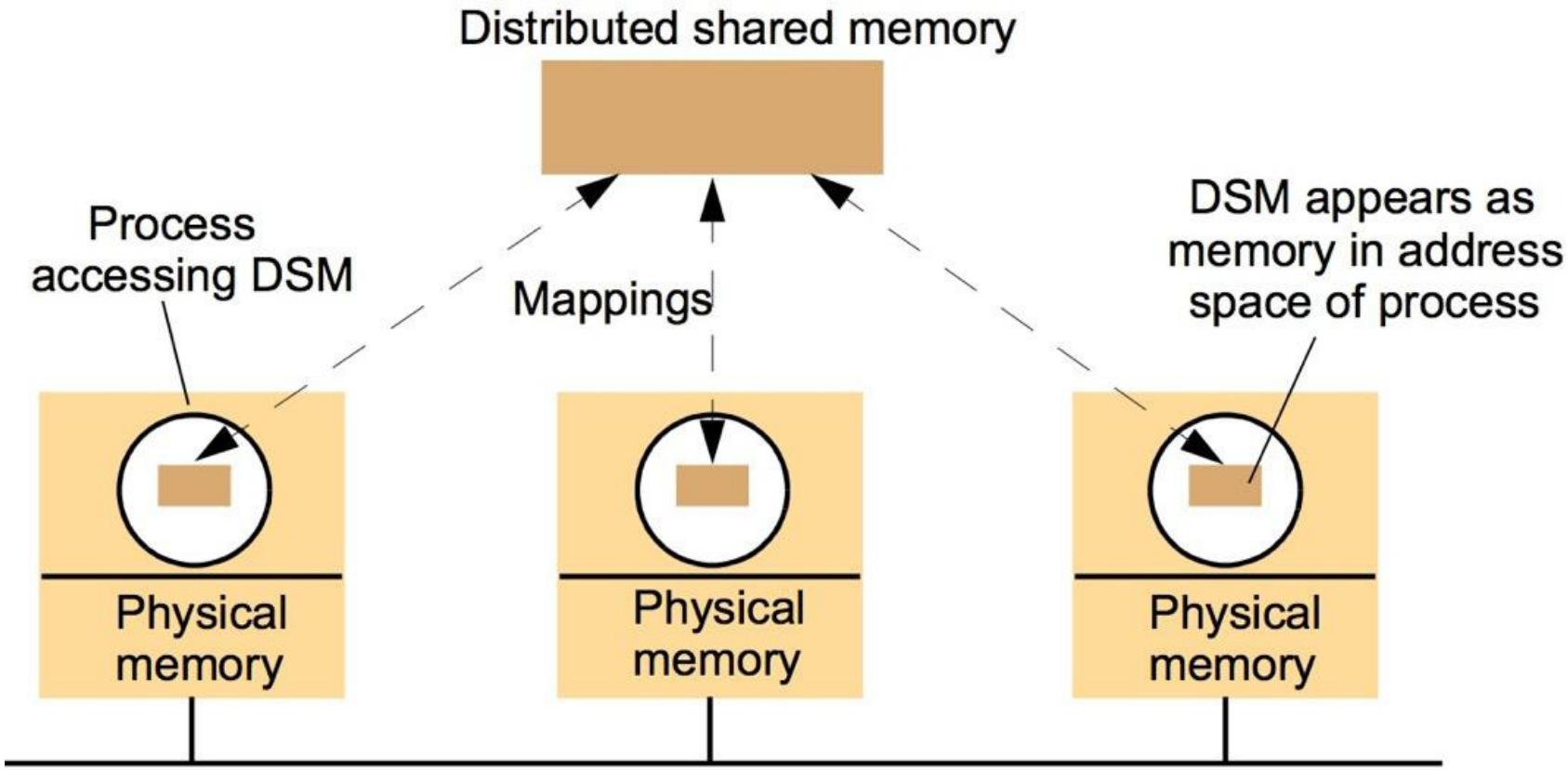
```
import javax.jms.*;
import javax.naming.*;
public class FireAlarmJMS {

public void raise() {
    try {                                                    1
        Context ctx = new InitialContext();          2
        TopicConnectionFactory topicFactory =        3
        (TopicConnectionFactory)ctx.lookup ("TopicConnectionFactory"); 4
        Topic topic = (Topic)ctx.lookup("Alarms");   5
        TopicConnection topicConn =          6
            topicConnectionFactory.createTopicConnection(); 7
        TopicSession topicSess = topicConn.createTopicSession(false, 8
            Session.AUTO_ACKNOWLEDGE);                                  9
        TopicPublisher topicPub = topicSess.createPublisher(topic);          10;
        TextMessage msg = topicSess.createTextMessage();          11
         msg.setText("Fire!");12
        topicPub.publish(message);       13
        } catch (Exception e) {      14
        } 15
}
```

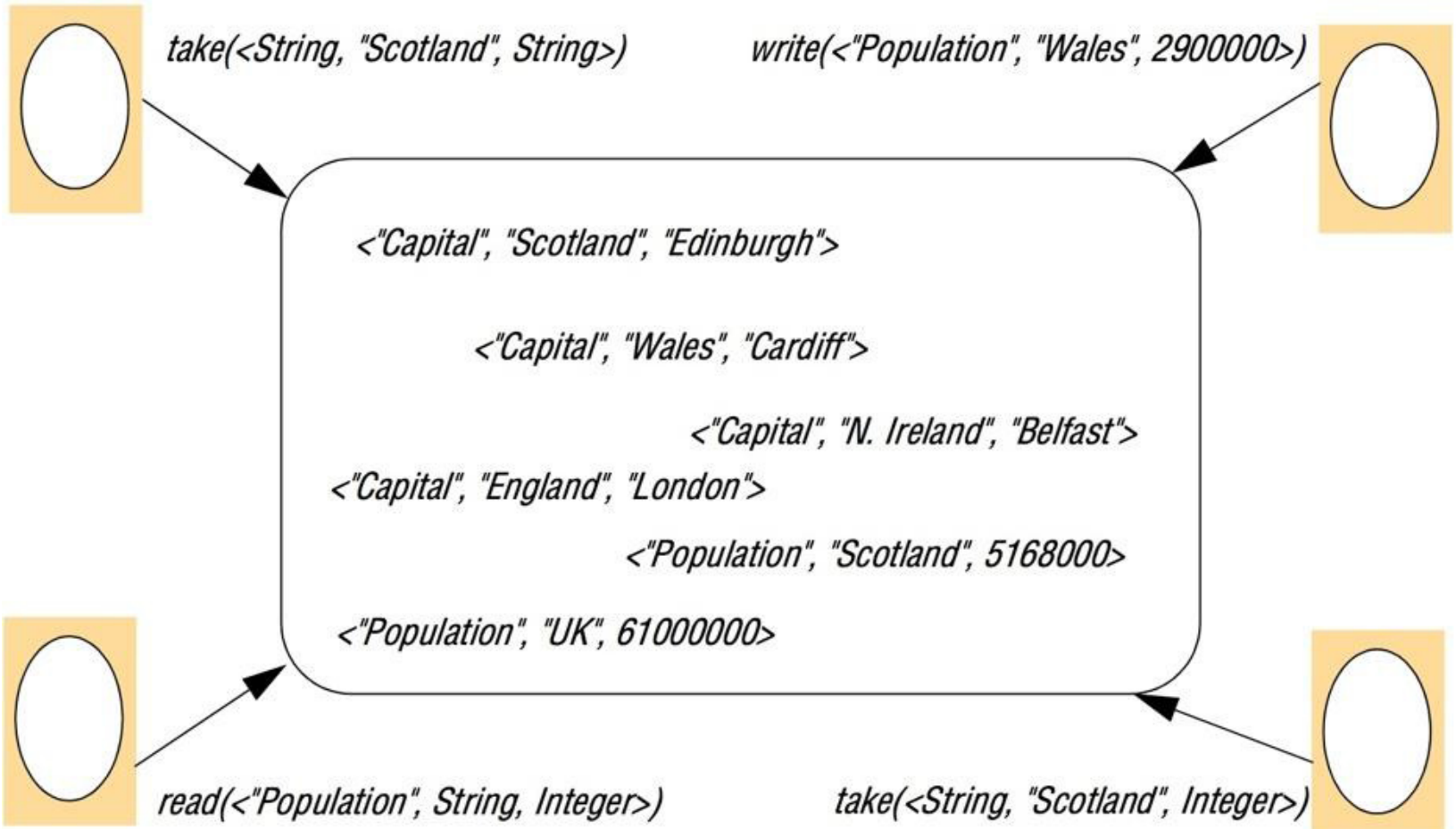
Class Java FireAlarmConsumer

```
import javax.jms.*;  import javax.naming.*;
public class FireAlarmConsumerJMS
public String await() {
    try {                                                    1
        Context ctx = new InitialContext();      2
        TopicConnectionFactory topicFactory =  3
            (TopicConnectionFactory)ctx.lookup("TopicConnectionFactory"); 4
         Topic topic = (Topic)ctx.lookup("Alarms"); 5
         TopicConnection topicConn =        6
               topicConnectionFactory.createTopicConnection();        7
         TopicSession topicSess = topicConn.createTopicSession(false,        8
                  Session.AUTO_ACKNOWLEDGE);        9
         TopicSubscriber topicSub = topicSess.createSubscriber(topic);            10
         topicSub.start();  11
         TextMessage msg = (TextMessage) topicSub.receive();         12
         return msg.getText(); 13
         } catch (Exception e) {    14
                  return null; 15
 }16
```

Abstraksi *Shared Distributed Memory*



Abstraksi dari *Tuple Space*



Replikasi dan Operasi pada *Tuple Space* – Xu & Liskov

- **Write**
  - Proses *request* dikirimkan dengan cara *multicast* kepada semua anggota (*member*)
  - Pada saat *request* telah diterima, tiap anggota menyisipkan *tuple* ke dalam replika yang dimiliki oleh masing-masing anggota kemudian mengkonfirmasi operasi tersebut dengan mengirimkan ACK
  - Step pertama dilakukan berulang kali hingga semua ACK diterima oleh si pengirim *request*

Replikasi dan Operasi pada *Tuple Space* – Xu & Liskov (cont'd)

- **Read**
  - *Request* dikirimkan dengan cara *multicast* kepada semua anggota (*member*)
  - Pada saat *request* telah diterima, satu anggota memberikan *tuple* yang cocok kepada si pengirim *request*
  - Si pengirim *request* akan mengembalikan sebuah *tuple* yang cocok dan pertama kali diterima sebagai hasil dari operasi yang dilakukan
  - Step awal dilakukan berulang-ulang hingga menerima minimal satu buah respon

Replikasi dan Operasi pada *Tuple Space* – Xu & Liskov (cont'd)

- **Take (tahap 1 – memilih *tuple* untuk dihapus)**
  - *Request* dikirimkan dengan cara *multicast* kepada semua anggota (*member*)
  - Pada saat *request* telah diterima, tiap replika yang berasosiasi dengan himpunan *tuple* akan di-*"lock"*. Jika proses *lock* gagal, maka *request* akan ditolak
  - Semua anggota memberikan balasan berupa himpunan semua *tuple* yang cocok *(matching tuples)* dengan permintaan si pengirim *request*

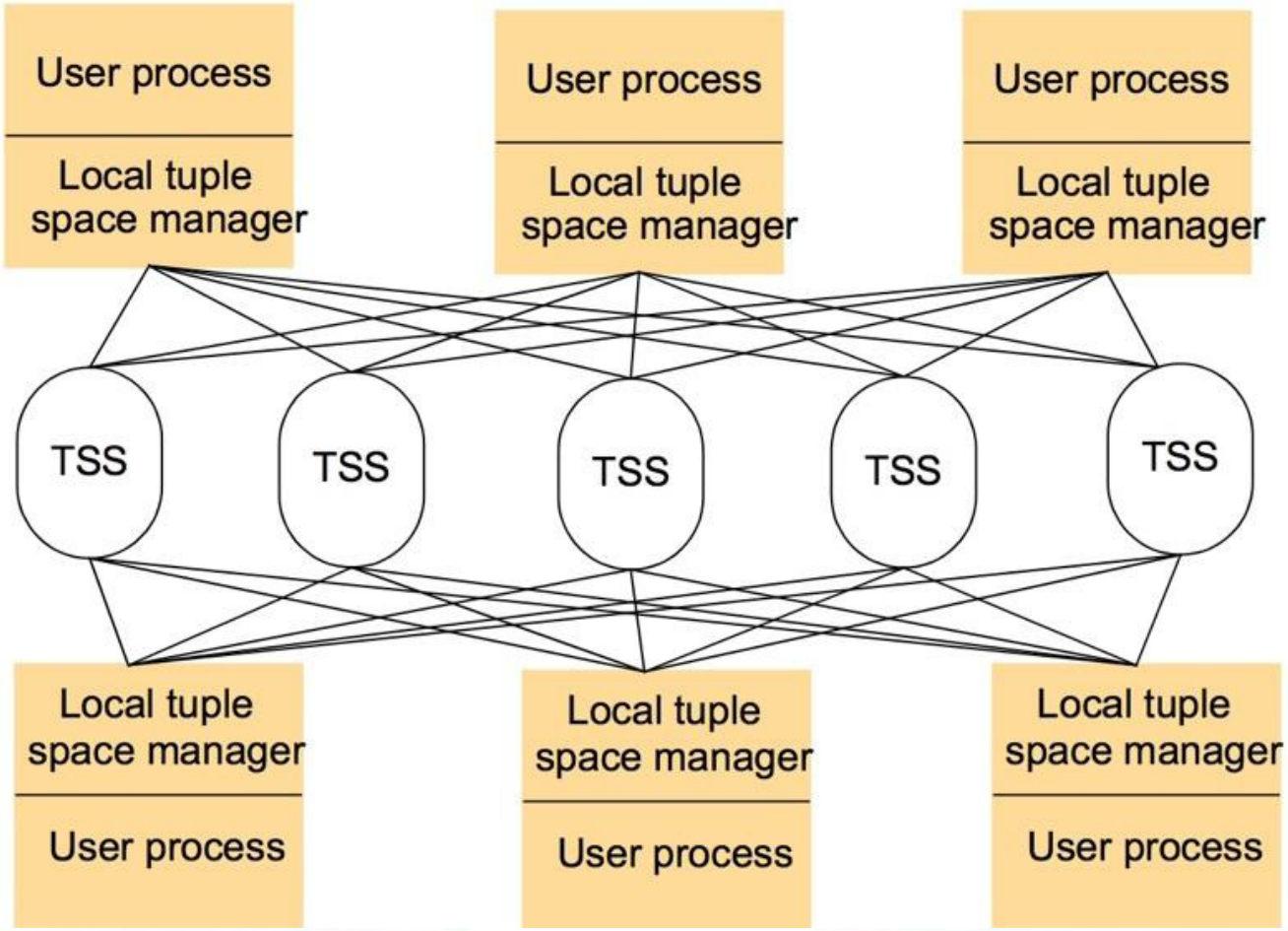
Replikasi dan Operasi pada *Tuple Space* – Xu & Liskov (cont'd)

- Langkah awal dilakukan berulang kali hingga semua anggota merespon *request* dan telah memberikan balasan
- Beberapa *tuple* dipilih sebagai hasil dari operasi yang dilakukan (pemilihan dilakukan secara acak)
- Jika hanya sedikit yang menerima *request* maka pihak yang menerima *request* akan meng-*unlock tuple* yang berasosiasi dengan masing-masing replika milik penerima dan mengulangi langkah pertama

Replikasi dan Operasi pada *Tuple Space* – Xu & Liskov (cont'd)

- **Take (tahap 2 – menghapus *tuple*)**
  - Pengirim mengirim *request* kepada semua anggota dengan cara *multicast* dengan menyisipkan *tuple* yang akan dihapus
  - Masing-masing anggota melakukan proses penghapusan *tuple* dari replikanya kemudian mengirimkan ACK dan meng-*unlock* kembali *tuple* yang berasosiasi dengan replikanya.

Model Partisi pada Kernel "York Linda"



API JavaSpaces

Operasi	Efek
Lease write(Entry e, Transaction t, ***long*** *lease)*	Menuliskan sebuah *entry* ke sebuah JavaScape tertentu
Entry read(Entry tpl, Transaction t, ***long*** *timeout)*	Mengembalikan nilai salinan (*copy*) dari *entry* yang cocok dengan nilai parameter *template*
Entry readIfExists(Entry tpl, Transaction t, ***long*** *timeout)*	Sama dengan fungsi *read* tetapi tidak "*blocking*"
Entry take(Entry tpl, Transaction t, ***long*** *timeout)*	Mengembalikan nilai *entry* yang cocok dengan nilai parameter *template* kemudian menghapusnya dari *tuple*
Entry takeIfExists(Entry tpl, Transaction t, ***long*** *timeout)*	Sama dengan fungsi *read* tetapi tidak "*blocking*"
EventRegistration notify(Entry tpl, Transaction t, RemoteeventListener rel, ***long*** *lease, MarshalledObject handback)*	Memberi notifikasi pada sebuah proses jika sebuah ada proses *write* pada *tuple* yang cocok dengan nilai parameter *template*

Ringkasan

	Group Communication	Publish-Subscribe System	Message queues	DSM	Tuple Spaces
Space-uncoupled	Ya	Ya	Ya	Ya	Ya
Time-uncoupled	Memungkinkan	Memungkinkan	Ya	Ya	Ya
Style layanan	*Communication-based*	*Communication-based*	*Communication-based*	State-based	State-based
Pola Interaksi	1-n	1-n	1-1	1-n	1-n / 1-1
Scalability	Terbatas	Memungkinkan	Memungkinkan	Terba-Tas	Terbatas

Ringkasan (cont'd)

	Group Communication	Publish-Subscribe System	Message queues	DSM	Tuple Spaces
Asosiatif	Tidak	Hanya *Content-based publish-subscribe*	Tidak	Tidak	Ya
Fokus utama	Komputasi tersebar yang handal	Penyebaran informasi, sistem *ubiquitous* dan *mobile*	Penyebaran informasi, proses transaksi komersial	Komputasi paralel dan ter-sebar	Kompu-Tasi *mobile*, *ubiqui-tous*, paralel dan ter-sebar

Ada Pertanyaan?

Referensi

- Coulouris, G. F., Dollimore, J., & Kindberg, T. (2012). *Distributed Systems: Concepts and Design 5th Edition*. London: Pearson Education.

Fakultas Informatika
School of Computing
Telkom University

THANK YOU