2.3 Teori Pendukung

2.3.1 Peer-to-Peer

Jaringan *peer-to-peer* atau biasa disingkat P2P merupakan sebuah jaringan dimana setiap entitasnya berlaku sebagai sebuah *client* (penerima layanan) dan *server* (pemberi layanan) di saat yang bersamaan. Sistem ini berbeda dengan sistem *client/server* dimana tiap partisipan hanya dapat berlaku sebagai *client* atau *server*, dan tidak dapat keduanya. Sebuah jaringan yang terdistribusi dapat disebut sebagai jaringan *peer-to-peer* jika partisipan-partisipan didalamnya membagikan sebagian dari sumberdaya perangkat keras-nya, seperti kapasitas penyimpanan, daya pemrosesan, kapasitas jaringan, dll. Sumber daya yang dibagikan ini penting untuk menyediakan layanan dan konten yang ditawarkan. Layanan dan konten ini dapat diakses secara langsung oleh *peer* lainnya, tanpa melalui perantara [8].

Beberapa contoh aplikasi yang menggunakan P2P, yaitu:

1. Napster. Sebuah layanan berbagi file melalui internet dan merupakan pionir dalam berbagi file melalui P2P. File yang dibagikan difokuskan pada file audio, seperti musik dalam format MP3.
2. BitTorrent. Merupakan protokol berbagi file *peer-to-peer* yang digunakan untuk mendistribusikan data atau file elektronik melalui internet.
3. Skype. Aplikasi yang meyediakan layanan chat video dan panggilan suara antara komputer, tablet, perangkat mobile, konsol Xbox One, dan *smartwatch* melalui internet. Skype awalnya menggunakan sistem *hybrid peer-to-peer* dan *client-server* namun, berubah sepenuhnya menjadi *client-server* setelah diakuisisi oleh Microsoft.
4. Bitcoin. Bentuk mata uang digital yang terdesentralisasi tanpa adanya bank pusat atau satu administrator dan memungkinkan pengiriman uang antar pengguna tanpa adanya perantara.

Protokol pada aplikasi P2P harus disetujui oleh setiap *peer*-nya dan protokol ini merupakan kumpulan dari jenis-jenis pesan yang berbeda dan semantiknya. Berbagai protokol aplikasi P2P memiliki kesamaan fitur, diantaranya:

1. Protokol ini dibuat pada application layer protokol jaringan dan membentuk suatu jaringan *overlay* virtual;
2. Beberapa desain memberikan identitas unik kepada *peer*-nya, yang juga merupakan alamat dan identitas dari peer tersebut;
3. Jenis pesan yang digunakan pada beragam protokol P2P adalah sama;
4. Mendukung kemampuan untuk mengarahkan pesan yaitu, pesan dapat disampaikan melalui *peer* perantara [9].

2.3.1.1 Arsitektur *Peer-to-Peer*

Berdasarkan dari bagaimana setiap node terhubung satu sama lain dalam jaringan overlay, dan bagaimana sumber daya di-index dan di-lokasikan, klasifikasi jaringan dapat dibagi menjadi tidak terstruktur atau terstruktur atau hybrid diantara keduanya.

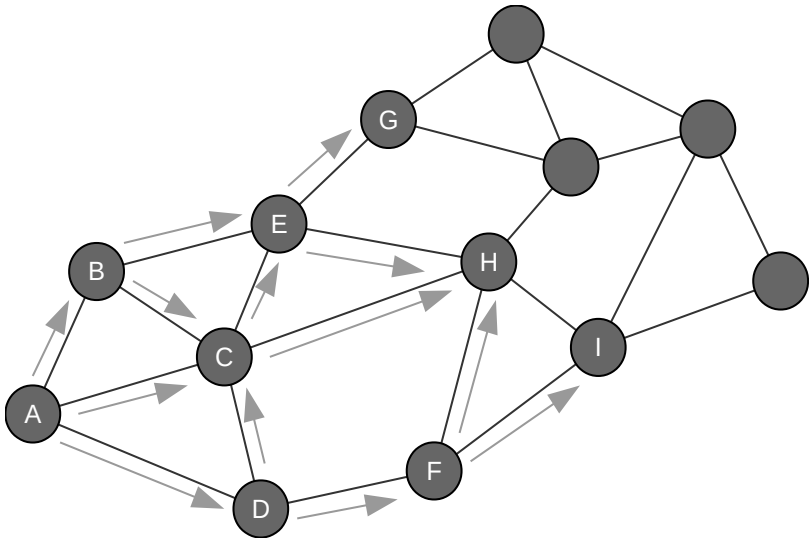
a) Jaringan Tidak Terstruktur

Jaringan tidak terstruktur tidak memaksakan struktur terentu pada desainnya. Jaringan ini dibentuk oleh node yang secara acak membuat sebuah koneksi satu sama lain dan dilakukan tanpa adanya aturan terpusat yang mengatur koneksi/hubungan antar *peer* [10]. Contoh dari aplikasi P2P dengan arsitektur *overlay* tidak terstruktur adalah Gnutella, FastTrack, KaZaA, dan FreeNet [9]. Jaringan tidak terstruktur ini lebih mudah untuk dibuat dan memungkinkan optimasi terlokalisasi untuk setiap area yang berbeda dengan overlay yang sama. Selain itu, jaringan tidak terstruktur memiliki kecepatan "*churn*" (*peer* yang bergabung atau meninggalkan *overlay*) yang tinggi.

Dalam melakukan pencarian *peer* pada *overlay* tidak struktur dapat dilakukan dengan dua cara/pendekatan. Pendekatan pertama adalah dengan melakukannya secara acak/*random.* Pendekatan yang kedua adalah dengan memanfaatkan *server* terpusat yang disebut sebagai *trackers* [10].

Flooding merupakan salah satu mekanisme yang digunakan dalam pendekatan pertama. Pada mekanisme *flooding* ini, untuk mencari *peer* lain yang aktif *client* melakukan *broadcast* pesan yang disebut "*ping*" pada jaringan. Saat

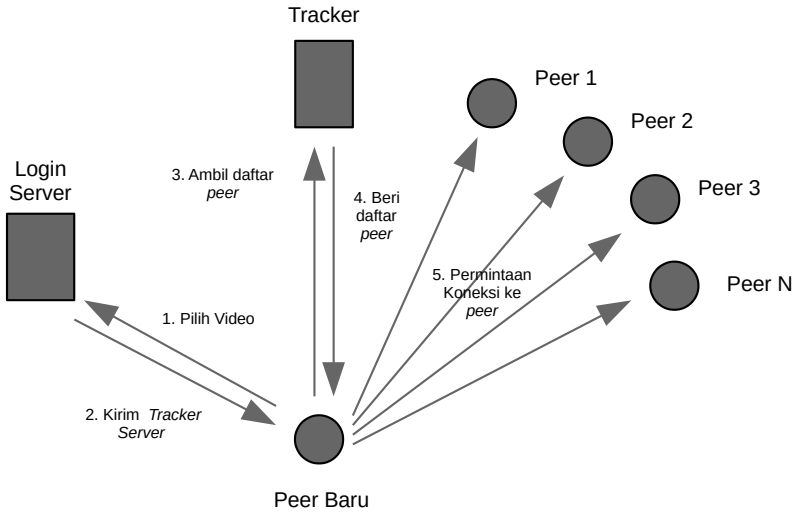
pesan *ping* tersebut diterima oleh *peer* aktif lain, *peer* ini akan membalasnya dengan pesan "*pong*" ke *client* awal tersebut. *Peer* yang pertamakali di berikan *request* atau di-*ping* ini dapat ditentukan melalui sebuah daftar *peer* yang sudah dikenal untuk memulai proses *bootstrapping.*



Gambar 2.1: Flooding pada arsitektur tidak terstruktur

Mekanisme *flooding* ini ditunjukkan seperti pada Gambar 2.1. Pada gambar tersebut, *Node* A merupakan node *client* yang melakukan *request* konten atau *resource.* Karena *node* A tidak mengetahui dimana atau siapa yang memiliki *peer* tersebut, *node* A dapat mencarinya dengan mengirimkan pesan ke *peer* tetangganya terlebih dahulu, yaitu *node* B, C, dan D. Apabila konten atau *resource* yang diminta tidak ditemukan pada pada ketiga *peer* tersebut, ketiga *peer* tersebut akan meneruskannya ke *peer* tetangganya juga, *node* B akan meneruskan ke *node* E dan C, *node* C ke *node* E dan H, dan *node* D ke *node* C dan F. Hal ini akan terus dilanjutkan hingga *resource* atau konten yang diminta oleh *node* A ditemukan.

Pendekatan menggunakan mekanisme *flooding* ini akan menghasilkan volume trafik yang sangat besar, bahkan dengan *hop-count* maksimum suatu *request* dapat menjalar hanya dibatasi hingga 7 [10]. Untuk mengatasinya dapat dibuat sebuah *super node* yang berfungsi sebagai *hub* atau *router,* sehingga membuat jaringan lebih terstruktur.



Gambar 2.2: Gambaran umum sistem P2P video streaming

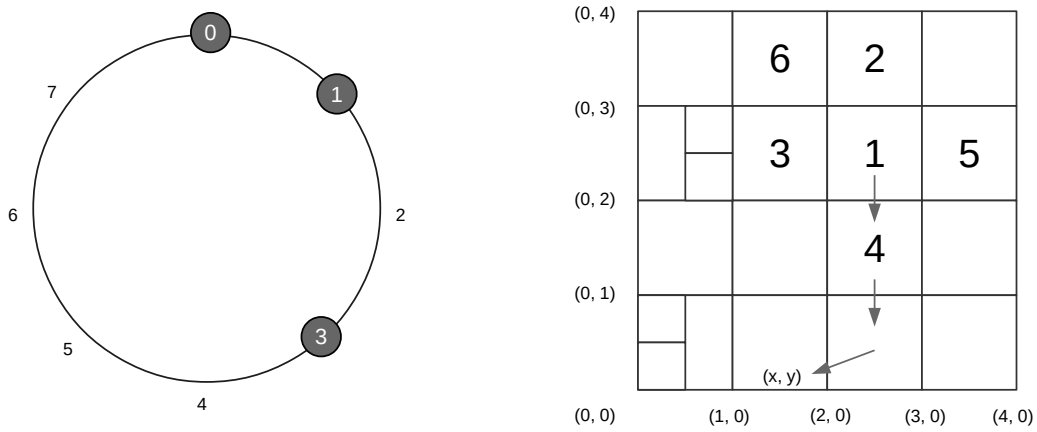
Untuk pendekatan kedua yang menggunakan *server* sebagai *trackers,* salah satu pengaplikasiannya adalah pada P2P *video streaming.* Secara umum, gambaran prosesnya terlihat seperti pada Gambar 2.2. Awalnya *peer* baru akan mengunjungi sebuah *log-in server* untuk memilih video atau film yang ingin ditonton. *Server* kemudian akan memberikan daftar *peer* yang sedang menonton video atau film yang sama kepada *peer* baru ini. Daftar *peer* tersebut dipilih secara acak. *Peer* baru ini kemudian akan memilih beberapa *peer* pada daftar *peer* yang telah diberikan untuk selanjutnya dilakukan proses *download.*

b) Jaringan Terstruktur

Pada jaringan *peer-to-peer* terstruktur *overlay* di organisir kedalam topologi tertentu, dan protokol akan memastikan bahwa setiap node dapat dengan efisien mencari file/resource pada jaringan, bahkan ketika resource tersebut sangat langka.

Jenis jaringan P2P terstruktur yang umum diimplementasikan adalah *distributed hash table* (DHT), dimana *hashing* yang konsisten digunakan untuk menandai kepemilikan pada setiap file untuk peer tertentu. Ini memungkinkan *peer* untuk mencari *resource* pada jaringan menggunakan *hash table* yang didalamnya terdapat sepasang *key* dan *value*, dan setiap *node* yang berpartisiasi

pada jaringan tersebut dapat dengan efisien mendapatkan *value* dengan *key* yang diberikan. *Node* pada jaringan terstruktur harus memiliki daftar "tetangga" yang memenuhi kriteria tertentu agar *node* tersebut dapat menyalurkan trafik secara efisien. Beberapa desain dari DHT adalah *Chord dan* CAN (*Content Addressable Network).* Desain Chord dan CAN ditunjukkan gambar Gambar 2.3 [10].



Gambar 2.3: Contoh dari Chord (kiri) dengan 3 node pada jaringan dan CAN (kanan) dengan 6 node dimana node 1 melewati 4 untuk sampai ke (x, y).

c) Jaringan *Hybrid*

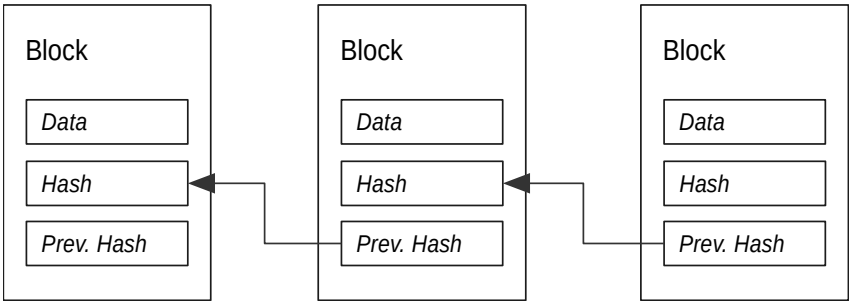
Jaringan dengan model *hybrid* merupakan kombinasi dari *peer-to-peer* dan *client-server.* Model jaringan ini biasanya memiliki *server* pusat yang membantu peer mencari dan menemukan satu sama lain. Pengguna model *hybrid* ini dijumpai pada aplikasi musik Spotify (hingga 2014) dan *Skype* (hingga dibeli oleh Microsoft). Saat ini, model hybrid memiliki performa yang lebih baik dari model jaringan terstruktur murni ataupun model jaringan tidak terstruktur murni.

2.3.2 Blockchain

Blockchain pertama kali diperkenalkan bersamaan dengan dibuatnya Bitcoin pada tahun 2008 oleh seseorang dengan nama Satoshi Nakamoto. Pada Bitcoin, *blockchain* merupakan sebuah *ledger* ("buku kas"), dimana didalamnya semua transaksi Bitcoin dicatat dan disimpan dengan aman [11]. Penggunaan *blockchain* tidak hanya terbatas pada sistem uang digital, bahkan penggunaan

awalnya sendiri adalah untuk menandai waktu pada dokumen *digital* secara aman yang diajukan oleh Haber dan Stornetta pada tahun 1991 [11].

Blockchain sendiri terdiri atas blok-blok yang saling terikat menggunakan kriptografi seperti pada Gambar 2.4. Masing-masing blok memiliki kriptografi hash dari blok sebelumya, *timestamp* (pendanda waktu), dan data (misalnya data transaksi). Karena blok selanjutnya terikat dengan blok sebelumnya, melakukan modifikasi blok akan menyebabkan perubahan *hash*, sehingga blok selanjutnya tidak lagi terikat ke blok sebelumnya.



Gambar 2.4: Diagram Blockchain

Misalnya, pada Gambar 2.4 yang terdiri dari tiga buah blok tersebut dilakukan modifikasi pada blok paling awal (paling kiri). *Hash* blok tersebut akan berubah yang menyebabkan blok kedua (tengah) tidak menunjuk ke blok pertama karena *prev. hash* tidak sama dengan *hash* pertama. Untuk memperbaikinya, perlu dilakukan *hash* ulang (*rehash)* pada blok kedua tersebut. Akan tetapi, dengan melakukan *hash* ulang blok kedua akan menyebabkan blok ketiga (paling kanan) tidak menunjuk ke blok kedua karena *prev. hash* blok ketiga sudah tidak sama lagi dengan *hash* blok kedua. Oleh karena itu, perubahan suatu blok akan merusak *chain* dengan blok-blok setelahnya.

2.3.2.1 Desentralisasi

Blockchain menghilangkan resiko yang mungkin terjadi pada data yang disimpan secara terpusat dengan menyimpan data pada jaringan *peer-to-peer* nya. Dengan jaringan *peer-to-peer*, celah keamanan yang mungkin dapat dieksploitasi

oleh peretas dapat diminimalisir karena tidak adanya titik pusat kegagalan (*central point of failure*).

Setiap *node* pada jaringan terdesentralisasi tersebut memiliki salinan dari *blockchain*. Dengan replikasi seperti ini kualitas data dapat terjamin. Tidak ada salinan resmi yang terpusat dan tidak ada pengguna yang dapat lebih dipercaya dari pengguna lainnya.