Survei Model Performansi Sistem *File sharing* BitTorrent pada Jaringan *Peer*-to-*Peer*

Irwan Wardoyo

PT. Sigma Cipta Caraka (Telkom Sigma)
irwan.wardoyo@sigma.co.id

Abstrak

Sampai saat ini BitTorrent merupakan aplikasi *file sharing peer-to-peer* yang paling popular, dengan trafik saat ini sebesar 3,35% dari trafik internet di dunia. Selain menarik banyak pengguna, BitTorrent menarik perhatian para peneliti untuk meneliti cara kerja dan kinerja performansi pada BitTorrent. Makalah ini menampilkan survei terhadap beberapa model yang digunakan untuk mengukur performansi dari sistem *file* sharing BitTorrent pada fase yang berbeda dengan mempertimbangkan keadaan nyata menggunakan model deterministik, model *markov chain* dan model antrean jaringan.

***Keywords**:* BitTorrent, Model Performansi, Model Deterministik, Model *Markov Chain*, Model Antrean Jaringan.

Received Juni 2018
Accepted for Publication August 2018
DOI: 10.22441/incomtech.v8i2.4090

1. PENDAHULUAN

Sistem *file sharing* dengan menggunakan teknologi *peer*-to-*peer* mulai diperkenalkan pada tahun 1999 oleh Napster [1], sebuah layanan untuk berbagi musik. Sejak saat itu aplikasi untuk *peer*-to-*peer* (P2P) mulai banyak dikembangkan, diantaranya: Gnutella [2], eDonkey2000, Kazaa, FastTrack, Freenet [3], Chord, CAN dan BitTorrent [4]. Dibandingkan dengan solusi klien server tradisional, teknologi P2P mempunyai kelebihan dengan menggabungkan sumberdaya (*bandwitdth*, *storage* dan komputasi) terdistribusi dari seluruh *peer* yang tergabung. Antar *peer* akan saling berkontribusi (seolah-olah seperti server yang memberi layanan pada kliennya) kepada *peer* yang lainnya sebagai bentuk imbalan. Dengan cara kerja seperti itu sistem *file sharing* dapat terdistribusi, tidak tergantung pada salah satu server dan dapat menghemat sumber daya.

Berdasarkan topologi dan tingkat sentralisasi yang digunakan dalam sistem *file sharing* pada jaringan P2P [5], dapat di kelompokkan menjadi 4, yaitu: (1)

Topologi tersentralisasi, (2) topologi tidak terstruktur terdesentralisasi, (3) Topologi terstruktur terdesentralisasi, (4) topologi terdesentralisasi sebagian. Contoh aplikasi *file sharing* pada jaringan P2P yang menggunakan topologi tersentralisasi adalah Napster [1] dan BitTorrent [2], terdapat sebuah server pusat yang bertugas untuk mengkoordinasikan komunikasi antara *peer* yang satu dengan yang lainnya. Untuk aplikasi yang menggunakan topologi tidak terstruktur terdesentralisasi, seperti Gnutella [2] dan Freenet [3], setiap *peer* bertindak baik sebagai server dan klien secara merata dan membentuk jaringan *overlay* (topologi logik) yang menghubungkan antar *peer* yang satu dengan *peer* yang lainnya secara bebas tanpa ada yang mengatur. Untuk aplikasi yang menggunakan topologi terstruktur terdesentralisasi, seperti Chord dan CAN, setiap *peer* dapat bertindak sebagai server dan klien secara bersamaan dan membuat jaringan *overlay* (topologi logik) yang dibangun terkendali dan diatur dengan menggunakan algoritma tertentu, seperti *Distribution Hash Table* (DHT). Sedangkan untuk aplikasi yang menggunakan topologi terdesentralisasi sebagian, termasuk FastTrack dan Brocade, memiliki beberapa super-node atau super-*peer* yang memainkan peran yang lebih penting daripada yang lain.

Sampai saat ini salah satu aplikasi P2P yang paling popular digunakan utuk *file sharing* adalah BitTorrent, dengan total trafik 3,35% [6] dari total trafik internet dunia atau sekitar 423,08 Peta Byte. BitTorrent bekerja menggunakan protokol BitTorrent yang didesain oleh Bram Cohen pada April 2001 dan mulai dicoba digunakan pada 2 juli 2001. Dengan menggunakan sistem yang tersentralisasi, terdapat server pusat yang berfungsi memelihara dan meyimpan sebuah direktori berisi indeks *file* yang dibagi oleh *peer*. Server pusat akan melakukan pembaharuan data apabila terdapat *peer* yang baru bergabung atau meninggalkan sistem dengan cara menambahkan atau menghapus indeks dari direktori. Ketika p*eer* membutuhkan *file* tertentu, maka *peer* akan meminta informasi indeks *file* yang dibutuhkan kepada server pusat, server pusat akan mencari indeks *file* tersebut pada direktori, lalu memberikan infomasi tentang lokasi *file* tersebut kepada *peer* yang meminta informasi tersebut. Sedangkan untuk proses transfer *file* dilakukan antar *peer* secara terdistribusi sesuai dengan informasi indeks yang berasal dari server pusat.

BitTorrent menjadi perhatian para peneliti karena selain memiliki fitur umum yang dimiliki oleh semua sistem *file sharing* pada jaringan *peer*-to-*peer*, juga mempunya beberapa inovasi yang dapat memeperbaiki kinerja dari BitTorrent, seperti *Tit-for-Tat* (TFT) dan *rarest fisrt* (RF). Selain itu BitTorrent memiliki kelemahan dalam proses pemilihan *peer* yang dapat menyebabkan penggunaan sumber daya jaringan yang tidak effisien. Dengan adanya kelebihan dan kekurangan yang terdapat pada BitTorrent banyak peneliti mempelajari, meneliti performansi pada BitTorrent dan mengusulkan perbaikan pada protokol BitTorrent.

Pada makalah ini akan disajikan beberapa teknik pemodelan yang digunakan dalam mengukur performansi sistem *file sharing* BitTorrent pada jaringan *peer*-to-*peer* dengan mengunakan model deterministik, model *markov chain*, dan model antrean jaringan.

2. BITTORENT

2.1 Cara Kerja BitTorrent

BitTorrent merupakan teknologi atau protokol yang digunakan untuk layanan *file sharing* dengan memanfaatkan sistem *peer*-to-*peer*. Langkah pertama dalam menggunakan teknologi BitTorrent adalah membuat *file* metainfo yang biasanya dikenal dengan sebuah *file* dengan ekstensi ".torrent". *File* torrent berisi informasi nama *file*, ukuran *file*, informasi *hash file* dan URL dari server pusat atau biasa dinamakan "*Tracker*". *Tracker* akan menyimpan log dari setiap *peer* yang sedang mengunduh *file* dan membantu *peer* supaya dapat berhubungan dengan *peer* yang lainnya untuk melakukan proses unduh dan unggah *file*. Setiap *peer* akan berkomunikasi dengan *tracker* dan memberikan informasi kepada *tracker* mengenai *file* mana yang diunduh, port berapa yang digunakan, dll. Sedangkan respon dari *tracker* adalah memberikan informasi pengguna lain yang sedang mengunduh *file* dan informasi mengenai bagaimana dapat terhubung dengan *peer* yang lain. Kelompok *Peer* yang sedang terkoneksi dan saling berbagi torrent yang sama biasanya disebut dengan "*swarm*".

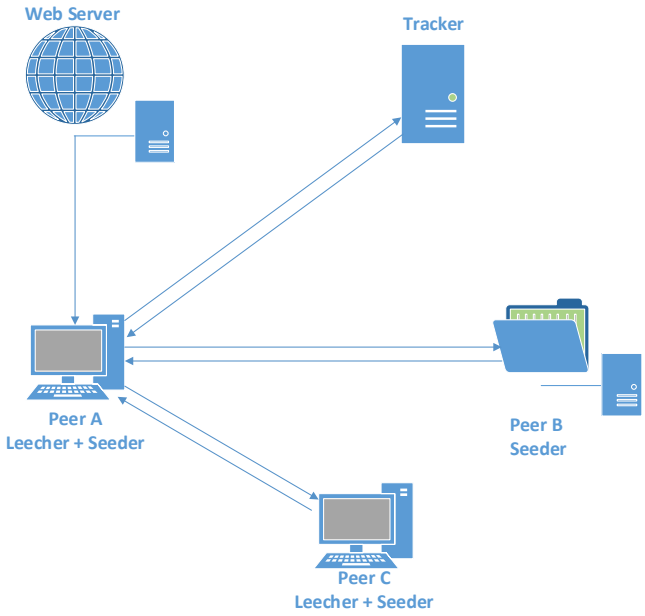
Peer dalam sistem BitTorrent diklasifikasikan menjadi 2 tipe: *seeder* dan *leecher*. *Seeder* merupakan *peer* yang memiliki *file* lengkap dari sebuah *file* torrent dan tetap aktif berkontribusi kepada *peer* lainnya (mengunggah). Sedangkan *leecher* adalah *peer* yang mengunduh *file* dan sama sekali tidak memiliki *file* torrent sebagian ataupun keseluruhan dari sebuah *file*. *Leecher* bisa diartikan sebuah klien yang mengunduh *file* pada *seeder* dan setelah *file* tersebut selesai diunduh maka *leecher* dapat pula bertindak sebagai *seeder* untuk *file* torrent yang baru diunduhnya. Dengan cara kerja seperti itu, apabila jumlah *leecher* semakin banyak maka kemungkinan jumlah *seeder* pun akan semakin banyak dan membuat sistem menjadi terdistribusi dan proses unggah tidak tepusat hanya pada satu *seeder*.

Dalam teknologi BitTorrent *file* asal akan dibagi dipecah menjadi bagian-bagian yang kecil yang dinamakan *piece*, dimana biasanya ukuran dari sebuah *piece* adalah 256KB atau 512KB. Untuk menjamin integritas dari *piece*, maka ditambahkan kode *hashing* SHA-1 pada setiap *piece*, sehingga *leecher* dapat melakukan verifikasi untuk kesempurnaan setiap *piece* yang sudah diunduh. Setiap *piece* yang behasil diunduh dan di verifikasi oleh *leecher*, maka akan langsung diinformasikan ke arah *peer* yang tergabung dalam *swarm* mengenai *piece* yang baru tersebut dan menyediakannya untuk *peer* yang lain.

Berikut adalah 5 langkah dasar untuk membuat sistem berabagi *file* pada bit torrent:

1. *Peer* A akan mengunduh sebuah *file* ".torrent" dari web server untuk dapat mengunduh *file* yang terdapat dalam sistem *file sharing* BitTorrent.
2. *Peer* A akan menghubungi *tracker* untuk mendapatkan list semua *peer* yang aktif yang berpartisipasi dalam torrent.
3. *Tracker* akan memberikan informasi list dari *peer* yang aktif dan tergabung dalam torrent.
4. *Peer* A akan menambahkan seluruh *peer* yang terdapat list tersebut dan menambahkan *peer* yang terdapat dalam daftar menjadi tetangga dan mengirimkan *piece* satu sama lainnya
5. Setelah permintaan *Peer* A diterima oleh tetangganya, maka *Peer* A bisa

saling menukarkan *piece* dengan tetangganya.



Gambar 1 Proses Sistem File sharing pada BitTorrent

2.2 MASALAH TEKNIS PADA BITTORRENT

2.2.1 Strategi Pemilihan *Piece*

Strategi pemilihan *piece* [7] mana yang akan diunduh memiliki pengaruh besar pada kinerja protokol. Pada dasarnya strategi pemilihan *piece* bertujuan meningkatkan ketersediaan salinan lengkap *file* di dalam jaringan torrent. Dengan begitu akan meningkatkan kecepatan unduhan dan memastikan bahwa salinan lengkap *file* masih terdapat di dalam jaringan apabila *seeder* menginggalkan jaringan torrent.

- *Sub-piece*

  Pada dalam sistem BitTorrent, *peer* akan memecah *file* torrent menjadi bagian-bagian kecil atau *piece* yang bisanya mempunyai ukuran 256KB atau 512KB lalu akan melakukan pertukaran *piece* satu sama lainnya. Pada proses proses pengiriman *file*, BitTorrent menggunakan protokol TCP pada *layer transport*. Didalam proses komunikasi dengan menggunakan TCP terdapat proses negosiasi antara kedua host yang akan melakukan komunikasi untuk menentukan berapa besar ukuran maksimum data yang bisa dikirimkan dalam satu waktu. Maka apabila terjadi *slow start* ataupun kapasitas penuh, BitTorrent akan menggunakan teknik *pipelining* dengan cara memecah *piece* menjadi beberapa bagian yang lebih kecil atau disebut dengan *sub-piece* atau *chunk*. Ukuran dari *sub-piece* atau *chunk* biasanya sekitar 16KB dan protokol BitTorrent akan memastikan akan selalu ada permintaan (biasanya 5) untuk *sub-piece*/*chunk* kapan saja.

- *Strict Policy*

  Ketika *sub-piece* yang satu sudah selesai diunduh maka akan ada *sub-piece* dari *piece* tersebut yang siap diunggah dan akan diselesaikan dulu untuk

sampai seluruh *sub-piece* selesai diunduh, sebelum meminta *sub-piece* dari *piece* yang lainnya. Dengan cara seperti itu proses transfer *file* selalu dapat berjalan setiap saat dan setiap *piece* dapat diunduh lebih cepat.

- *Rarest first*

  Rarest fisrt merupakan kebijakan pada protokol BitTorrent, dimana ketika sebuah *peer* memilih *piece* berikutnya akan diunduh maka *peer* tersebut akan memilih *piece* paling langka yang tersedia pada *peer* lainnya atau dengan kata lain yang jumlah salinan *piece*-nya paling sedikit. Tujuan dari kebijakan ini adalah menyebarkan *piece* langka yang tadinya hanya terdapat pada satu atau beberapa *seeder* ke *peer* yang lainnya dengan cepat. Dengan terserbarnya *piece* yang tadinya langka maka akan mempercepat proses unduhan untuk *peer* yang lainnya karena sumber daya tersedia pada banyak *peer* dan tidak akan menyebabkan *bottleneck* karena hanya satu atau beberapa *seeder* saja yang menyediakan *piece* langka tersebut.
- *Random First Piece*

  Ketika sebuah *peer* bergabung ke dalam sistem, maka *peer* tersebut siap untuk mengunduh *piece* tetapi belum ada *piece* yang bisa diunggah. Sangat penting untuk mendapatkan *piece* pertama dari arah *seeder* dengan cepat dan mulai mengunggah *piece* ke arah *peer* yang lainnya. Apabila menggunakan metode *Rarest First* pada saat mengunduh *piece* pertama, maka akan menyebabkan proses unduhan yang lambat karena harus mencari terlebih dahulu mana *piece* yang langka. Oleh karena itu proses pengunduhan *piece* pertama akan dilakukan secara random "*Random First Piece*", setelah itu untuk proses pemilihan *piece* yang berikutnya akan menggunakan "*Rarest Piece*".
- *Endgame Mode*

  Pada saat fase akhir proses mengunduh *piece* dari *peer* yang lain mengalami masalah dikarenakan kecepatan transfer yang lambat, ini dapat menyebabkan *delay* untuk menyelsaikan proses unduhan. Untuk mengatasi permasalah tersebut maka *peer* akan mengirimkan permintaan secara *broadcast* ke seluruh *peer* untuk mengirimkan *sub-piece* dari *piece* yang terakhir. Apabila *piece* yang dibutuhkan sudah diterima maka permintaan untuk mengirimkan sub-*piece* akan ditolak untuk mencegah duplikasi.

2.2.2 Strategi Pemilihan *Peer*

Tidak adanya alokasi sumber daya yang terpusat pada BitTorrent, maka setiap *peer* harus dapat memilih *peer* yang akan dijadikan sebagai tetangga. Strategi pemilihan *peer* [7] menjadi penting untuk memperbaiki kualitas unduhan atau insentif untuk *peer* yang sudah berkontribusi pada sistem dan memberikan hukuman pada "*Free rider*", yang merupakan *peer* yang hanya melakukan pengunduhan tanpa berkontribusi pada sistem atau dengan kata lain tidak pernah menggungah. Untuk strategi pemilihan *peer* dapat dilakukan dengan mekanisme *Tit-for-Tat* (TFT), *Optimistic unchoking*, *Anti-snubbing*, dan *upload only*.

- *Tit-for-Tat* (TFT)

  Tit-for-tat bekerja berdasarkan algoritma *chocking*, dengan cara memberikan kebebasan kepada 4 *peer* yang memberikan tingkat unduhan paling tinggi pada *peer* tersebut. Pada dasarnya mekanis *tit-for-tat* ini dijalankan setiap 10 detik oleh setiap *peer*, dengan melihat tingkat unduhan yang ditentukan dengan jumlah data yang diterima oleh *peer* tersebut selama

20 Detik. Sebagai insentif kepada 4 *peer* yang sudah memberikan tingkat unggahan paling tinggi, maka *peer* tersebut diberikan kebebasan untuk mengunduh *piece* yang ada, sementara *peer* yang lain yang terhubung akan dibatasi *bandwidth*-nya atau dicekik "*choke*".

- *Optimistic unchoking*

  Optimistic unchoking merupakan mekanisme yang digunakan untuk mencari apakah terdapat *peer* baru yang mempunya performansi lebih baik. Setiap 30 detik sebuah *peer* akan memberikan kebebasan kepada 5 *peer* yang melakukan pengunduhan pada *peer* tersebut. Dari waktu yang sudah ditentukan maka akan didapatkan kesimpulan apakah terdapat *peer* baru yang performansinya lebih baik untuk melakukan pertukaran *piece*. Jika *peer* hanya menggunakan mekanisme *tit-for-tat* maka tidak dapat mengetahui apabila terdapat *peer* yang baru bergabung dan memiliki performansi yang lebih baik. Kombinasi *tit-for-tat* dan *optimistic unchocking* dapat secara berkala mencegah *free rider*.
- *Anti-snubbing*

  Anti-snubbing digunakan apabila terdapat sebuah *peer* yang selama 60 menit tidak mendapatkan *piece* apa-apa dikarenakan proses dari *tit-for-tat* dan *optimistic unchoking*. Maka *peer* tersebut dianggap sebagai "*snubbed*". Maka mengikuti mekanisme dari *tit-for-tat,* tetangga dari *peer* tersebut tidak akan menguggah *piece* pada *peer* tersebut. Hal ini akan meningkatkan jumlah *optimistic unchokes* yang dapat memberikan kesempatan pada *peer* tersebut untuk menemukan koneksi yang lebih cepat.
- *Upload-only*

  Mekanisme *Upload-only* digunakan untuk *seeder* karena pada *seeder* tidak terdapat kegiatan mengunduh tapi hanya terdapat kegiatan mengunggah. Oleh karena itu mekanisme *tit-fot-tat* tidak berlaku untuk *peer* yang hanya bertindak sebagai *seeder*. *Seeder* akan memilih *peer* yang mempunyai kemampuan untuk dapat mengunggah *piece* ke arah *peer* yang lainnya. S*eeder* dapat mengunggah *piece* ke lebih banyak *peer* dan akan memilih *peer* yang tidak sedang mendunduh *piece* yang sama dari *peer* lain saat ini.

3. PEMODELAN PERFORMANSI

3.1 Ukuran Performansi

Sistem *file sharing* BitTorrent pada jaringan *peer*-to-*peer* adalah jaringan yang terdistribusi, dimana pembagian tugas dan beban kerjanya dibagi antar *peer* dan membutuhkan kerjasama antar *peer*. Pengaruh seperti *bandwidth* yang tidak sama, perilaku pengguna yang tidak diharapkan, *free rider* dan *churn* membuat tidak mudah untuk menentukan performansi sistem tersebut. Dari penilitian-penelitian sebelumnya sudah dilakukan banyak riset untuk membuat pemodelan kinerja sistem berbagai *file* pada jaringan *peer*-to-*peer*, diantaranya:

Kapasitas layanan (C)

Dari perspektif sistem, kapasitas layanan adalah keseluruhan *throughput* yang dapat dicapai sistem yang dapat ditawarkan kepada pengunduh dalam torrent. Kapasitas layanan memperhitungkan keseluruhan *bandwidth* unggahan yang efektif dari *seeder* dan *leecher*. Hal ini mencerminkan bahwa *seeder* dapat mengunggah pada *bandwidth* yang tersedia, sementara *leecher* hanya dapat mengunggah sebagian kecil dari *bandwidth* yang tersedia karena tidak memiliki salinan lengkap dari *file.*

***Throughput* unduhan per *Peer* (α)**

Dari perspektif pengguna, *throughput* pengunduhan per *peer* dianggap sebagai kapasitas layanan untuk setiap *peer*. Dapat diartikan sebagai anggregasi kapasitas layanan unggahan dibagi dengan jumlah pengunduh.

Latensi yang dibutuhkan untuk megunduh *File* (d)

Latensi yang dibutuhkan untuk megunduh *file* adalah waktu dari permintaan *file* yang dikirim sampai *file* selesai diunduh sepenuhnya. Ini terdiri dari dua komponen: waktu untuk pencarian dan waktu transfer *file*.

Ketersediaan *file* / Torrent *lifetime* (T)

Sebuah *file* hanya tersedia ketika *peer* dapat mengunduh semua bagian yang dibutuhkan dari *seeder* atau *peer* lain yang bertindak sebagai *seeder* di dalam sistem. Jika terdapat setidaknya satu *seeder* dalam sistem, maka *file* dijamin akan tersedia sepanjang waktu. Namun apabila *seeder* meninggalkan sistem setelah mendapatkan semua *file*, atau mungkin ada terdapat perilaku pengguna yang tidak terduga dan kegagalan jaringan, maka dapat menyebabkan proses unduhan yang tidah sempurna dan menyebabkan torrent mati.

Stabilitas jaringan

Dalam jaringan *file sharing* P2P yang stabil, semua permintaan unduhan akan dilayani dan dihapus dalam waktu yang terbatas, jika beban kerja rata-rata tidak melebihi kapasitas sistem layanan rata-rata. Jika jumlah *peer* dan kinerja setiap *peer* relatif stabil, maka kita katakan sistem memasuki "*Steady State*". Oleh karena itu, stabilitas jaringan P2P adalah masalah mendasar untuk setiap analisis kondisi *Steady State*.

3.2 Beberapa Faktor yang Mempengaruhi Performansi

Churn

Efek dari kedatangan dan kepergian ribuan atau jutaan *peer* disebut *churn*, yang merupakan sifat bawaan dari sistem P2P. Setiap *peer* dapat dengan acak bergabung atau meninggalkan sistem ketika pengguna ingin memulai atau keluar dari aplikasi. Mengingat sifat kerjasama *peer* jaringan P2P, konektivitas masuk dan keluarnya *peer* dapat menyebabkan penurunan kinerja yang dalam pengunduhan *file*. Oleh karena itu, keterlibatan dan kebiasaan dari pengguna harus diperhitungkan dalam analisis kinerja sistem P2P.

Free rider

Free rider adalah pengguna yang memperoleh sumber daya dari jaringan tanpa berkontribusi atau egois. Perilaku ini merugikan *peer* lain yang sudah berkontibusi pada sistem. Jika proporsi *free rider* cukup banyak maka akan berpengaruh kepada kinerja unduhan dan lebih buruknya dapat membuat kontributor untuk menginggalkan sistem. Oleh karena itu *free rider* merupakan ancaman utama terhadap kinerja jaringan P2P.

Heterogenitas *Bandwidth*

Heterogenitas *bandwidth* pada *peer* mempunyai dampak serius terhadap kinerja seluruh jaringan dan *peer* individu. Keragaman *bandwidth peer* berasal dari dua alasan. Pertama-tama, ada heterogenitas kapasitas akses tautan di infrastruktur (Broadband, DSL, Cable, PON). Jelas, akses *bandwidth* yang tidak sama, dapat secara langsung menurunkan pengalaman unduhan *peer*. Kedua, banyak strategi alokasi bandwidth telah diusulkan untuk meningkatkan kinerja jaringan BitTorrent atau mengontrol trafik BitTorrent di internet.

3.3 Teknik Pemodelan Performansi

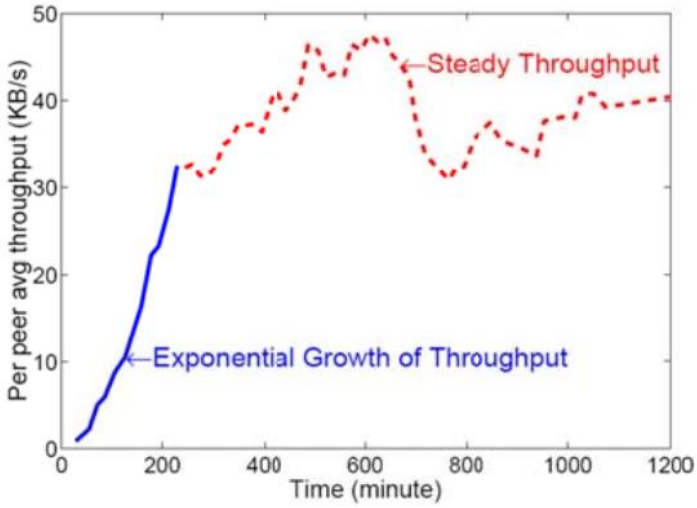
Beberapa model analisis untuk performasi sistem *file sharing* BitTorrent pada jaringan P2P telah dilakukan dan diusulkan pada beberapa litelatur. Beberapa teknik yang sudah digunakan untuk pemodelan performansi sistem *file sharing* BitTorrent pada jaringan P2P adalah (1) model deterministik, (2) model *markov chain* dan (3) model antrean jaringan. Dengan beberapa model analitik yang berbeda, seharusnya dapat mencerminkan efek dari parameter yang berbeda pada kinerja jaringan BitTorrent. Secara umum, parameter khusus yang menjadi ciri sistem *file sharing* BitTorrent pada jaringan *peer*-to-*peer*:

- **x**, jumlah pengunduh / *leecher*
- **y**, jumlah *seeder* dalam sistem.
- **λ**, tingkat kedatangan permintaan baru.
- **μ**, *bandwidth* unggahan dari *peer*.
- **c**, *bandwidth* unduhan *peer*.
- **θ**, tingkat dimana *peer* membatalkan unduhan.
- **γ**, tingkat dimana *seeder* meninggalkan sistem.
- **η**, efektivitas *file sharing*, berkisar dalam [0, 1].

Selain parameter umum diatas, terdapat beberapa buah tambahan parameter yang digunakan dalam model analitik yang berbeda. Misalnya apakah *bandwidth* untuk yang dimiliki setiap peer untuk melakukan pengunggahan dan pengunduhan homogen, dimana semua peer memiliki tingkat unggahan dan unduhan yang sama atau heterogen, dimana *peer* memiliki tingkat pengunggahan dan unduhan yang berbeda. Contoh lain adalah bahwa orang-orang biasanya menganggap pembagian *piece* sebagai proses independen dari bagian lain, namun dalam kenyataannya, dinamika pembagian *piece* tergantung pada *piece* yang lain juga berdasarkan pada strategi pemilihan *piece* yang digunakan oleh BitTorrent. Selain itu beberapa model memeriksa berbagai tahapan pengunduhan yang mungkin dimiliki oleh *peer* pada suatu titik waktu. Ini biasanya dilakukan dengan membagi *peer*

pengunduh x ke dalam sub-himpunan berbeda x1, x2, x3, x4, x5, dan lain sebagainya yang mewakili tahap penyelesaian pengunduhan yang berbeda. Demikian pula, beberapa model memisahkan *peer* x ke xn dan xf sesuai dengan *free rider* dan bukan *free rider*.

Dari hasil pengukuran dan pengamatan nyata [9] [10], sistem BitTorrent memiliki 2 fase, yaitu *transient phase* dan *stationary phase* dalam sudut pandang *throughput* unduh per *peer* untuk *file* yang diberikan seperti terlihat pada gambar 2. Dari hasil pengukuran yang ditunjukkan pada gambar 2 pada *transient phase* merupakan masa-masa awal *file* pertama kali dipublikasikan dan terlihat terdapat lonjakan besar permintaan *file* yang dipublikasikan tersebut. Pada fase ini hanya sedikit *peer* yang memiliki *piece* dari *file* yang dipublikasikan tadi, lalu antar *peer* saling bertukar satu sama lainnya. Setelah itu akan semakin banyak *peer* yang memiliki keseluruhan atau sebagian dari *file* dan melayani limpahan permintaan *peer* yang lainnya, hal ini memungkinkan pertumbuhan eksponensial dalam kapasitas layanan sistem. Setelah tingkat permintaan menjadi stabil, jaringan memasuki "*stationary phase*" di mana kinerja seluruh jaringan dan setiap *peer* menjadi stabil.



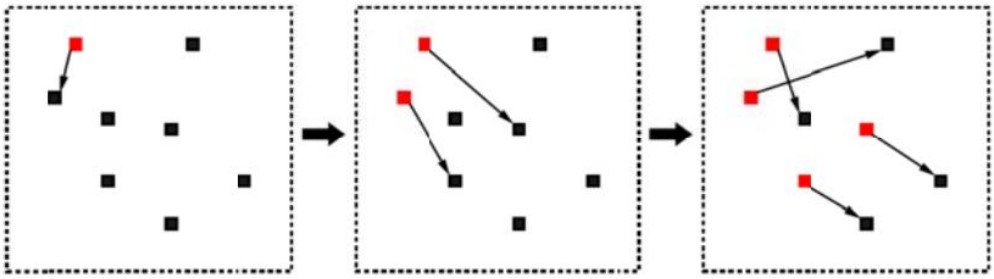
Gambar 2. 2 Fase evolusi dari transient phase menuju stationary phase

3.3.1 Model Deterministik

Model deterministik [8] [9] [10] biasanya diterapkan untuk menyelidiki kinerja dalam pada *transient phase*. Pada *transient phase* terdapat lonjakan permintaan pada jaringan P2P, katakanlah *n*. Lalu *peer* yang memiliki *file* masih terbatas, kita asumsikan hanya satu yang dapat melayani unduh *file* untuk *peer* yang lainnya. Skenario seperti itu mungkin muncul karena *file* baru dipublikasikan dan belum terjadi proses replikasi *file*. Prosedur replikasi *file* ini diilustrasikan pada Gambar. 3, dapat dijelaskan dengan baik oleh model deterministik.

Dapat diusulkan bahwa ada $n = 2^k$ *peer* yang meminta *file* dalam jaringan, dengan *bandwidth* untuk menguggah yang homogen $\mu = b$ dan tidak ada batasan untuk melakukan unduhan $c = \infty$. Seluruh *file* diasumsikan sebagai satu *piece* dengan ukuran *s* bits, dan *peer* hanya dapat melayani *peer* lain setelah sepenuhnya mengunduh *file*. Untuk melayani permintaan *n*, bit *ns* harus direplikasi dan ditransfer ke semua *peer*. Strategi terbaik adalah pertama-tama melayani satu

pengguna dengan rate *b*, sehingga kapasitas layanan sistem tumbuh menjadi *2b*, dan kemudian dua *peer* yang sudah mempunya *file* lengkap akan melayani *peer* lain, sampai semua *n* pengguna mendapatkan layanan, seperti yang diilustrasikan pada Gambar. 3.



Gambar 3. Sistem File Sharing pada Jaringan P2P

Berbasis pada strategi ini, *peer* akan selesai mengunduh *file* setiap $\tau = s/b$ detik, dan kemudian melayani *peer* yang lain. Oleh karena itu, setelah setiap τ detik, layanan sistem bisa digunakan, yang mengarah ke pertumbuhan eksponensial jumlah $2^{t/\tau}$ untuk jumlah unit yang tersedia melayani *peer* lain. Maka *n peer* dapat dilayani oleh waktu log2 $n = k$. Oleh karena itu, rata-rata latensi unduhan $\bar{d}$ yang dialami oleh *peer*, sebagai berikut :

$$\bar{d} = \frac{1}{n}\sum_{j=1}^{n} d_j = \sum_{i=0}^{k-1} 2^{i-k}\,\tau(i+1) = k\tau - \frac{n-1}{n}\tau = \tau\left(\log_2 n - \frac{n-1}{n}\right) \approx \tau \log_2 n \qquad (1)$$

Misalkan d_j yang menunjukkan *delay* yang dialami oleh *peer* ke-*j* untuk menyelesaikan pengunduhan, dan perhatikan bahwa 2^{i-k} *n peer* menyelesaikan layanan pada waktu *i+1*. Oleh karena itu, untuk jaringan P2P dengan kondisi terdapat lonjakan permintaan awal, rata-rata *delay* pada proses pengunduhan per *peer* adalah $\log_2 n$.

Untuk mengurangi *delay* pada proses pengunduhan, dapat melakukan pengunuhan *multi-piece* yang sudah diimplementasikan dalam banyak sistem *file* sharing P2P, seperti BitTorrent. Sebuah *file* dibagi menjadi *piece* dengan ukuran yang identik dengan begitu akan memudahkan ketikan melakukan pengundugan multi-*piece* dan *peer* yang sudah mendapatkan *file* seluruhnya atau sebagian dapat melayani *peer* yang lain dengan memberikan *piece* tersebut segera setelah diunduh. Ini adalah dasar yang memungkinkan transmisi *file* pipelining, dan mempercepat penundaan pengunduhan dengan faktor 1/m [9] [10].

Model deterministik di atas menggambarkan proses dasar latensi pengunduhan *file* yang mungkin terdapat dari sistem BitTorrent selama *transient phase*. Model deterministik memberikan gambaran ide dasar tentang skalabilitas sistem BitTorrent selama *transient phase*, dalam arti bahwa rata-rata latensi pengunduhan tumbuh secara logaritmik sebagai konsekuensi dari permintaan yang melojak (jumlah total *peer* yang meminta).

3.3.2 Model Markov Chain

Model *markov chain*[9] digunakan untuk analisis *steady state phase* pada sistem *file sharing* BitTorrent pada jaringan P2P, ketika permintaan menjadi

cukup stasioner dan berkelanjutan setelah *transient phase*. Pada keadaan stabil, tidak ada lonjakan permintaan mengunduh *file* dari *peer*, oleh karena itu dapat dilihat sebagai proses *poisson* dengan laju λ. Status sistem dapat dinotasikan sebagai pasangan (*x, y*), dimana *x* mewakili jumlah *leecher*, dan *y* menunjukkan jumlah *seeder* yang masih melayani sistem. *File* tersebut dibagi-bagi menjadi beberapa bagian *piece* dan memungkinkan dilakukannya unduhan *multi-piece*, sehingga *peer* dalam proses unduhan dapat menyediakan *piece* yang tersedia bagi *peer* lain. Oleh karena itu, baik *seeder* dan *leecher* berkontribusi pada kapasitas layanan sistem. Kontribusi *leecher* hanya sebagian kecil dari *seeder* yang memiliki *file* lengkap. Setelah *peer* sudah selesai mengunduh sebuah *file*, mereka berhak menjadi *seeder* untuk melayani *peer* yang lain. Oleh karena itu, total kapasitas layanan dilambangkan dengan $\mu(\eta x + y)$. Namun, *seeder* dapat meninggalkan sistem dengan kecepatan γ. Proses transisi keadaan sistem ini dapat dijelaskan oleh *Continuous Time Markov Chain* (CTMC) model dengan matriks Q

$$\begin{aligned} &q\big((x,y),(x+1,y)\big) = \lambda \text{ kedatangan baru} \\ &q\big((x,y),(x-1,y+1)\big) = \mu(\eta x + y) \text{melayani } peer \\ &q((x,y),(x,y-1)) = \gamma\ y \text{ keluar sistem.} \end{aligned} \quad (2)$$

Selain itu, kapasitas layanan sistem P2P dan *delay* dari proses pengunduhan *file* per *peer* akan menjadi jumlah *throughput* unggahan dari kedua *leecher* dan *seeder*, adalah :

$$C = \mu\eta\, x + \mu\, y \quad (3)$$

Kemudian, dapat memperoleh *throughput* unduhan per *peer* dalam kondisi stabil, adalah :

$$\alpha = \frac{C}{x} = \frac{\mu\eta\, x + \mu\, y}{x} \quad (4)$$

Dalam arsitektur terpusat, waktu proses kueri pencarian informasi *peer* yang diharapkan dapat diabaikan. Oleh karena itu, latensi unduhan *file* pada *steady state* dapat secara langsung terkait dengan *throughput* unduhan per *peer* adalah

$$d = \frac{1}{\alpha} = \frac{x}{\mu\eta\, x + \mu\, y} \quad (5)$$

Kondisi *(x, y)*, dimana *y = 0* pada *markov chain* dalam keadaan menyerap (*absorbing state*). Keadaan ini terjadi ketika semua *seeder* meninggalkan sistem, lalu kumpulan *file* lengkap dalam jaringan tidak dapat dijamin dan kematian seluruh proses *file sharing* tidak dapat dihindarkan. Oleh karena itu, waktu mulai dari keadaan awal (0, 1) sampai dengan waktu ketika keaadaan penyerapan dapat diartikan sebagai waktu hidup sistem [12] [13].

$$T = \sum_{(x,y):y>0} z_{(x,y)} = \sum_{(x,y):y>0} \int_0^\infty \pi_{(x,y)}(t)dt \quad (6)$$

Dimana *z (x, y)* menunjukkan waktu rata-rata yang dihabiskan dalam keadaan *(x, y)* dari awal *(t = 0)* ke penyerapan sistem, dan π *(x, y) (t)* adalah probabilitas keadaan *(x, y)* pada waktu *t*.

Model *markov chain* dapat juga digunakan untuk menganalisis performansi pada sistem *file sharing* BitTorrent pada jaringan P2P apabila faktor *bandwidth*, *multiclass Peer*, *multiple piece download* dan faktor *free rider* menjadi pertimbangan. Ketika kendala *bandwidth* pada proses unggah dan unduh dalam sistem dianggap nyata [12] [13], tingkat layanan total sistem menjadi min $\{cx, \mu\,(\eta\, x + y)\}$. Dengan kata lain, tingkat transisi $q\,((x,y),(x-1,y+1))$ dalam matrik transisi *markov chain* Q menjadi min $\{cx, \mu\,(\eta\, x + y)\}$. Jika $cx < \mu\,(\eta\, x + y)$, pengunduh tidak dapat menggunakan semua kapasitas unggah dan *bandwidth* unduhan adalah kendala. Jika sebaliknya, maka *bandwidth* unggahan yang menjadi kendala.

Dalam [14], model *markov chain* dapat digunakan untuk mempelajari *peer* yang terdistribusi dengan kondisi yang berbeda-beda dalam penyelesaian unduhan. *S0, S1, ..., SN-1* dengan $\left[\frac{0}{N},\frac{1}{N}\right], \left[\frac{1}{N},\frac{2}{N}\right], \ldots, \left[\frac{N-1}{N},1\right]$ bagian-bagian dari *file* masing-masing, dan *SN* adalah status untuk *seeder*. Ketika *peer* baru pertama masuk ke sistem BitTorrent, ia akan mulai sebagai *leecher* dengan bagian *0/N* dari *file* dan status *S0*. Setelah beberapa *peer* dalam sistem telah mengunggah *piece* kepada *peer* tersebut, maka *peer* tersebut akan memiliki *1 / N* dan status *seeder* berubah menjadi *S*. Proses pengunggahan dan pengunduhan berlanjut sampai *peer* mendapatkan semua bagian dan status *seeder* menjadi status *SN*. Namun, kondisi ini tidak selalu berhasil apabila *seeder* pergi atau berubah ke keadaan *offline* selama proses pengunduhan sebelum menyelesaikan. Akhirnya, *seeder* akan meninggalkan sistem setelah itu berfungsi dalam sistem untuk jangka waktu tertentu. Kondisi yang dapat diamati dengan CTMC.

Dampak dari *free rider* dianggap [15], mengganggu kinerja jaringan dan degradasi. Rasio *free rider* dan populasi keseluruhan diasumsikan f dalam jaringan. Karena *free rider* hanya mengunduh dari orang lain tetapi tidak pernah berkontribusi ke jaringan, efektivitas *file sharing* dari *free rider* sama dengan *0*. Oleh karena itu, efektivitas rata-rata dari *file sharing* per *peer* dikurangi dari η ke η *(1) - f*. Selain itu, setelah *free rider* meninggalkan sistem segera setelah menyelesaikan unduhannya, tingkat rata-rata *seeder* meninggalkan sistem meningkat dari γ ke γ *(1) - f*.

Sebuah model g*eneric markov chain* diperkenalkan [16] untuk memodelkan sistem *file* sharing P2P dengan kebijakan yang berbeda. Untuk mewakili model antrean untuk layanan P2P sistem di mana kedua pekerjaan dan server secara dinamis tiba dan pergi, maka menggunakan *markov chain* dua dimensi untuk model jaringan P2P dengan notasi A / B / (C / E) / POLICY, di mana A / B dan C / E mewakili dinamika pekerjaan dan server, masing-masing. Selain itu, kebijakan layanan (POLICY) termasuk (1) FCFS (*First-Come-First-Served*), yang mewakili pekerjaan dilayani dalam urutan kedatangan oleh semua server saat ini dalam sistem; (2) PS (k) (*k-Processor-Sharing*), yang menunjukkan setiap pekerjaan

dalam sistem dilayani tidak lebih dari server k secara bersamaan. Dalam sistem nyata, batasan *k-processor-sharing* ini berasal dari pembatasan kapasitas *downlink* dari *peer*. Untuk mewakili sistem dengan dinamika pekerjaan dan server yang berbeda, notasi yang digunakan untuk distribusi kedatangan (A atau C) dan distribusi seumur hidup (B atau E) dapat mengikuti (1) Proses *Poisson* M, (2) proses deterministik D, dan (3) general atau *free rider* G. Selain itu, dinamika server mungkin terkait dengan dinamika pekerjaan, karena setiap pengunduh di jaringan juga bertindak sebagai server. Ada dua kemungkinan kasus korelasi. Kasus pertama adalah ketika kedatangan server dan distribusi seumur hidup identik dengan kedatangan pekerjaan dan distribusi seumur hidup, maka notasi "-" digunakan untuk C dan E. Kasus kedua adalah ketika umur server sama dengan penjumlahan usia kerja dan periode tambahan waktu setelah distribusi eksponensial atau sewenang-wenang, notasi + M / + G digunakan untuk menggambarkan distribusi seumur hidup server E. Dalam hal ini, setelah *peer* selesai mengunduh *file* lengkap, ia tetap online sebagai *seeder* untuk sementara waktu untuk melayani orang lain. Oleh karena itu, empat kelas sistem P2P dianalisis, yaitu (1) M / M / (M / M) / FCFS, (2) M / M / (M / M) / PS (k), (3) M / M / (- / -), (4) M / M / (- / + G). Dalam pekerjaan ini, kondisi stabilitas diberikan untuk empat kelas sistem berdasarkan model *markov chain* di atas. Mereka menunjukkan bahwa jika beban kerja rata-rata tidak melebihi kapasitas layanan sistem rata-rata, maka sistem P2P stabil, yaitu semua pekerjaan yang tiba akan dilayani dan dihapus dalam waktu yang terbatas.

3.3.3 Model Antrean Jaringan

Model antrean jaringan mewakili struktur berbagai sistem dengan sejumlah besar sumber daya. Model antrean jaringan adalah pendekatan khusus untuk memodelkan sistem komputer yang terdiri dari banyak stasiun layanan yang terhubung satu sama lain. Sebuah node, dalam jaringan mewakili sumber daya dalam sistem yang sebenarnya. Pekerjaan pada prinsipnya dapat ditransfer antara dua node dari jaringan. Dalam sistem *file* sharing BitTorrent P2P, antrean model jaringan dapat digunakan untuk memodelkan struktur banyak *peer* yang terhubung sebagai jaringan, atau struktur urutan perilaku *peer* pada tahap operasi yang berbeda oleh abstraksi model antrean jaringan yang berbeda menekankan pada berbagai kinerja metrik, seperti yang digambarkan pada *File Transfer Latency Modeling* dan *Peers Behavior Modeling*. *File Transfer Latency Modeling* [17], model antrean terbuka dikembangkan untuk mengevaluasi latensi *file* transfer P2P dalam jaringan yang terdiri pada peer tujuan dan router inti dikedua sistem P2P, terpusat dan terdesentralisasi. Total *delay* transfer *file* dihasilkan dari penjumlahan (1) waktu pencarian kueri, (2) level *delay peer*, yaitu waktu transmisi *file* yang sedang diunduh, dan (3) delay pada jaringan inti, yaitu penundaan antrean pada perangkat router yang digunakan sebagai perantara transmisi *file*. Dalam arsitektur terpusat, seperti sistem BitTorrent, server pusat berisi indeks semua *file* yang digunakan oleh *peer* dalam sistem P2P dan waktu pencarian untuk kueri adalah waktu pencarian rata-rata server pusat tersebut untuk mengambil informasi. Dengan demikian, waktu pencarian kueri yang diharapkan sangat kecil dibandingkan dengan waktu unduhan *file*, dan dapat diperlakukan sebagai nilai konstan. Dapat dimodelkan setiap *peer* sebagai antrean M/G/1/K *processor sharing* (PS) untuk mengevaluasi *delay* pemrosesan *peer*, sementara

memodelkan setiap router sebagai GI/G/1 antrean untuk memperkirakan penundaan jaringan inti. Model ini menyumbang sejumlah faktor dalam sistem P2P nyata, seperti popularitas/ukuran *file* yang heterogen, jumlah unduhan simultan dalam pengaturan *peer*, tingkat tautan yang berbeda, topologi fisik yang beragam, dan *churn*. *Peers Behavior Modeling* [18], pada jaringan antrean tertutup diusulkan untuk memodelkan perilaku *peer* untuk jaringan *file sharing* P2P secara umum. Perilaku *peer* terdiri dari serangkaian fase yang terhubung dengan masing-masing *peer* lainnya, koneksi apa yang digunakan pengguna, pencarian konten dan pemrosesan kueri, transfer *file*, dan pemutusan koneksi pengguna. Model antrean jaringan ini cukup umum, dapat digunakan untuk menggambarkan arsiterktur jaringan P2P yang berbeda (jaringan tersentralisasi, jaringan terstruktur terdesentralisasi dan jaringan tidak terstruktur terdesentralisasi) dan dua kelas *peer* (*free riders* dan *non-free riders*). Walaupun terdapat perbedaan, tetapi terdapat 3 kebiasaan utama yang dilakukan oleh setiap peer (1) pemiliharaan infrastruktur jaringan P2P, (2) penanganan kueri, dan (3) *file* transfer. Pada jaringan topologi tersentralisasi, server inti akan menangani seluruh pemeriharaan infrasruktur sistem P2P dan pemrosesan kueri, sedangkan pada jaringan topologi terdesentralisasi tersebar diseluruh *peer*.

4. PERBANDINGAN PERFORMANSI MODEL

Tabel 1. memberikan ringkasan tentang berbagai teknik pemodelan performansi sistem *file sharing* P2P BitTorrent yang dibahas dalam makalah ini. Dalam tabel ini, model-model dasar dan kemungkinan perluasan dengan mempertimbangkan faktor-faktor yang lebih realistis ditunjukkan dalam dua kolom pertama untuk model deterministik, model rantai Markov, namun model jaringan antrean adalah dispersif, masing-masing pekerjaan memiliki formulasi jaringan antrean sendiri dan tidak ada model yang khas atau diperluas. Kolom ketiga memberikan metrik kinerja yang diselidiki dalam model yang berbeda, dan metrik tersebut didefinisikan dalam bagian 3.3.

Model deterministik diterapkan pada *transient phase* atau pada masa awal setelah *file* dimasukkan ke dalam jaringan oleh *seeder*. Pada saat itu terjadi lonjakan besar permintaan yang secara bersamaan memasuki sistem, tetapi disaat yang besamaan hanya satu atau beberapa *peer* saja yang dapat melayani dan permintaan dari *peer* yang lain. Dalam hal ini, kapasitas layanan sistem tumbuh secara eksponensial yang digambarkan sebagai proses replikasi deterministik, yang mudah diperoleh. Namun, model deterministik kurang ideal dan jauh dari kenyataan. Tidak ada pertimbangan tentang berbagai kapasitas tautan, perilaku *peer* yang heterogen, proses kedatangan *peer* yang acak, dan *free rider*.

Setelah tingkat kedatangan permintaan *file* menjadi stabil, sistem masuk ke kondisi stabil dimana kapasitas layanan sistem dan *throughput* per *peer* stasioner. Oleh karena itu, model *markov chain* adalah alat pemodelan yang tepat untuk mengevaluasi kinerja *steady state*. Berdasarkan hasil pengukuran pada [9], dapat disimpulkan hanya jika *file* tersebut cukup populer untuk menarik sejumlah besar *peer*, maka sistem akan mencapai kondisi *steady state* dan oleh karena itu model *markov chain* dapat diterapkan. Analisis dengan menggunakan *markov chain* dapat dikembangkan untuk meningkatkan keakuratan dengan cara menambahkan

status pada *markov chain* dan tingkat transisi yang lebih banyak menggunakan parameter tambahan, seperti *bandwidth*, *free rider*, kondisi *seeder* dan lain-lain.

Model antrean jaringan juga berlaku untuk analisis performansi pada *steady state phase* dan telah menjadi alat matematika populer untuk memodelkan banyak jaringan komputer dunia dalam nyata. Untuk jaringan P2P, *host*/*node* dalam antrean jaringan bisa menjadi *peer* atau fase perilaku *peer* dengan abstraksi. Oleh karena itu, model antrean jaringan cukup fleksibel untuk dirancang. Hasil simulasi dalam [17] memvalidasi model antrean jaringan, di mana setiap router *peer* dan *core* dimodelkan sebagai antrean M/G/1/K dan antrean GI/G/1. Model antrean jaringan tidak dapat skalabel dengan baik karena keterbatasan solusi algoritma numerik. Antrean jaringan dapat dibagi menjadi dua jenis: (1) *Product-form queuing network* (PFQN), dan (2) *Non-Product-form queuing network* (NPFQN). PFQN ditampilkan dengan waktu antar-kedatangan dan layanan yang terdistribusi secara eksponensial. Solusi untuk probabilitas *steady state* dapat dinyatakan sebagai produk dari faktor yang menggambarkan keadaan setiap *node*. Meskipun PFQN dapat diungkapkan dengan mudah, cukup perhitungan diperlukan untuk menganalisa bahkan jaringan kecil. Oleh karena itu, sebagai pengganti solusi eksak, algoritma pendekatan dikembangkan untuk memecahkan jaringan PFQN berskala besar. Untuk memilih algoritma perkiraan yang tepat, trade-off antara iterasi dan akurasi harus dibuat tergantung pada aplikasi tertentu [19]. Jaringan antrean lain yang lebih umum adalah NPFQN, yang merupakan fitur dengan waktu layanan non-eksponensial terdistribusi. Mirip dengan PFQN, algoritma aproksimasi dikembangkan untuk jaringan NPFQN skala besar. Namun, berbeda algoritma aproksimasi hanya berlaku untuk menyelesaikan kasus-kasus tertentu dari jaringan NPFQN .

Tabel 1. Ringkasan Teknik Pemodelan Performansi dengan mempertimbangkan faktor teknik dan metrik

Model	**Fase**	**Faktor (*extension*)**	**Metrik Performansi**
Model Deterministik	*Transient*	*Multi-piece download*	Latensi pengunduhan *file*
Model *Markov Chain Steady State*		*Download/upload bandwidth constraints*	Kapasitas layanan
			Throughput unduhan per *peer*
		Multiclass peers	Latensi pengunduhan *file*
		Multi-pieces download	Ketersediaan *file*
		Impact of free rider	Ketersediaan *file*
		Generalized model	Stabilitas jaringan
Model Antrean Jaringan	*Steady State*	*Heterogeneous peer behavior*	Kapasitas layanan
		Bandwidth allocation strategy	*Throughput* unduhan per *peer*
		Multiclass peers	Latensi pengunduhan *file*

5. KESIMPULAN

Dalam makalah dilakukan survei pemodelan performansi untuk sistem *file sharing* BitTorrent pada jaringan *peer-to-peer*. Teknik pemodelan untuk performansi sistem *file sharing* BitTorrent pada jaringan peer-to-peer menggunakan (1) model deterministik, (2) model *markov chain*, dan (3) model antrean jaringan. Model deterministik dapat diterapkan untuk menganalisis performansi pada saat transient phase, tetapi kurang ideal dan jauh dari kenyataan karena tidak pertimbangan tentang berbagai kapasitas tautan, perilaku *peer* yang heterogen, proses kedatangan *peer* yang acak, dan *free rider*. Untuk model *markov chain* dan model antrean jaringan dapat digunakan pada *steady-state phase* dan dapat ditambahkan berbagai faktor untuk dapat memodelkan performasi dari sistem *file sharing* BitTorrent pada jaringan *peer-to-peer* lebih nyata, dengan menambahkan faktor keterbatasan *bandwidth* untuk pengunggaha/pengunduhan, *multiclass peer*, *multiple piece download* dan faktor *free rider* sebagai faktor pertimbangan. Metrik performansi yang dapat dijadikan acuan, diantaranya kapasitas layanan, throughput unduhan per peer, latensi pengunduhan *file*, ketersediaan *file* dan stabilitas jaringan. Selanjutnya,dipelukan survei teknik pemodelan performansi pada jaringan BitTorrent untuk aplikasi *realtime*, seperti video streaming.

REFERENCES

[1] " napster", https://us.napster.com/
[2] "Gnutella," http://www.gnutelliums.com/.
[3] "Freenet," http://www.freenetproject.org/.
[4] "Bittorrent," http://www.bittorrent.com/.
[5] Xu Yong, Deng Chi, Gao Min, The Topology of P2P Jaringan, Journal of Emerging Trends in Computing and Information Sciences, VOL. 3, NO. 8 Aug, 2012,
[6] "Application Usage worldwide", http://researchcenter.paloaltojaringans.com/app-usage-risk-report-visualization/#sthash.X34kOdHm.dpbs
[7] Bolch Gunter, Greiner Stefan, de Meer Hermann, S. Trivedi Kishor, Queueing Jaringans and Markov Chains *Modeling and Performance Evaluation with Computer Science Applications*, A JOHN WILEY & SONS, INC., PUBLICATION, 2006
[8] Arne Johnsen Jahn, Erik Karlsen Lars, Sæther Birkelandm Sebjørn, Peer-to-peer jaringaning with BitTorrent, Department of Telematics, NTNU - December 2005
[9] X. Yang and G. de Veciana, "Service Capacity of Peer to Peer Jaringans," in *Proc. IEEE INFOCOM*, vol.4, March 2004, pp. 2242-2252
[10] G. de Veciana and X. Yang. "Fairness, incentives and performance in peer-to peer jaringans," in *Proc.Forty-first Annual Allerton Conference on Communication, Control and Computing*, USA, Oct. 2003.
[11] Xiangying Yang, Gustavo de Veciana. "Performance of peer-to-peer jaringans: Service capacity and role of resource sharing policies," *Performance Evaluation in P2P Computing Systems*, vol. 63, no. 3, pp. 175- 94, March 2006.
[12] R. Susitaival, S. Aalto and J. Virtamo, "Analyzing the dynamics and resource usage of P2P *file* sharing systems by a spatio-temporal model," in *Proc. International Workshop on P2P for High Performance Computational Sciences* (*P2P-HPCS'06*) *in conjunction with ICCS*, 2006.
[13] R. Susitaival and S. Aalto, "Modeling the population dynamics and the *file* availability in a BitTorrent p2p system with decreasing Peer Arrival rate," in *Proc. International Workshop on SelfOrganizing Systems* (*IWSOS*), 2006.

[14] Taoyu Li, Minghua Chen, Dah-Ming Chiu, and Maoke Chen, "Queuing models for peer-to-peer systems," in *Proc. 8th International Conference on Peer-to-peer systems* (*IPTPS*), 2009, pp. 4-4.

[15] Taoyu Li, Minghua Chen, Dah-Ming Chiu, and Maoke Chen, "Queuing models for peer-to-peer systems," in *Proc. 8th International Conference on Peer-to-peer systems* (*IPTPS*), 2009, pp. 4-4.

[16] R. Susitaival, S. Aalto and J. Virtamo, "Analyzing the dynamics and resource usage of P2P *file* sharing systems by a spatio-temporal model," in *Proc. International Workshop on P2P for High Performance Computational Sciences* (*P2P-HPCS'06*) *in conjunction with ICCS*, 2006. Taoyu Li, Minghua Chen, Dah

[17] Ming Chiu, and Maoke Chen, "Queuing models for peer-to-peer systems," in *Proc. 8th International Conference on Peer-to-peer systems* (*IPTPS*), 2009, pp. 4-4. Taoyu Li, Minghua Chen, Dah-Ming Chiu, and Maoke Chen, "Queuing models for peer-to-peer systems," in *Proc. 8th International Conference on Peer-to-peer systems* (*IPTPS*), 2009, pp. 4-4.

[18] Ge, Z., Figueiredo, D.R., Sharad Jaiswal, Kurose, J., Towsley, D., "Modeling peer-peer *file* sharing systems," in *Proc. IEEE INFOCOM*, 2003, pp. 2188-2198.

[19] Gunter Bolch, Stefan Greiner, Hermann de Meer, Kishor S. Trivedi, *Queueing jaringans and Markov chains: modeling and performance evaluation with computer science applications*. WileyInterscience, New York, NY, 1998.