BAB II

LANDASAN TEORI

2.1 Poin SKKM UMN

Universitas Multimedia Nusantara didirikan pada tahun 2006, dan di resmikan pada tahun 2007. Pada awalnya diresmikan, Universitas Multimedia Nusantara membuka fakultas ICT (*Information and Communication Technology)* dengan mengangkat tema "Pengembangan Sumber daya Manusia Menyongsong Era ICT". SKKM (Satuan Kredit Kegiatan Mahasiswa) poin menjadi salah satu syarat kelulusan bagi mahasiswa Universitas Multimedia Nusantara (Sejarah UMN, 2020).

SKKM yang diperlukan untuk mencapai kelulusan adalah 20 poin yang terdiri dari, 30% SKKM ilmiah dan penalaran, 20% minat dan bakat, 30% organisasi dan pengembangan kepribadian, dan yang terakhir adalah 20% pengabdian dan masyarakat (Untuk memperoleh poin SKKM, mahasiswa dapat mengikuti kegiatan yang berasal dari dalam kampus, maupun dari luar kampus) (UMN, 2020). Kegiatan yang dapat diikuti seperti seminar, kerja bakti di lingkungan RT/RW, Unit Kegiatan Mahasiswa (UKM), dan sebagainya (Achyarini, 2018).

Prosedur dalam melakukan *claim* poin SKKM, bisa mengirim *email*, dengan menyertakan bukti yang dapat diverifikasi, seperti sertifikat, nomor telepon dari Lembaga yang mengadakan, mengirim bukti foto kegiatan, dan sebagainya (Susanto J. dan Selarosa C., 2019). Mahasiswa yang mengajukan SKKM dari luar kampus dapat mengajukan dua SKKM atau lebih dalam satu pengajuan (Susanto, J. & Seto, A. B., 2019).

2.2 EOS

Blockchain merupakan sistem transaksi secara *peer-to-peer* atau langsung, tanpa perantara pihak ketiga. *Blockchain* memiliki beberapa *layer, Application Layer, Execution Layer, Semantic Layer, Propagation Layer,* dan *Consensus Layer* (EOSIO, 2020).

Application layer merupakan bagian aplikasi yang berisi perintah-perintah untuk menjalankan aplikasi *Blockchain. Execution Layer* merupakan bagian perintah yang dijalankan dari *Application Layer* berjalan. *Propagation Layer* adalah mengecek perintah-perintah yang berjalan pada *Execution Layer* merupakan perintah yang *valid*. *Propagation Layer* merupakan bagian berkomunikasi antar *peer*. *Consensus Layer* merupakan bagian yang paling dasar dari *blockchain*, *Consensus layer* berguna untuk membuat semua node yang ada pada *Blockchain Network* itu bisa memasukkan node yang baru (Singhal B., dkk, 2018).

EOS merupakan protokol *blockchain* dengan kecepatan transaksi yang mumpuni dan utilitas yang fleksibel. Diperkenalkan pada bulan Mei 2017, dan terkenal dalam performanya yang tinggi untuk bisnis. *Block one* sebagai perusahaan yang mengeluarkan *platform* EOSIO dan terus mengembangkannya hingga sekarang (EOSIO, 2020).

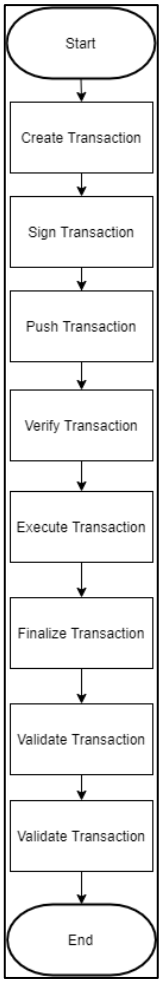
Blockchain EOS memiliki dua *layer consensus,* yang pertama adalah *Native Consensus Model*(aBFT) dan *Delegated Proof of Stake* (DPoS). Cara kerja *Native Consensus Model*(aBFT) adalah memastikan *block* yang diterima dan telah dilakukan proses *sign* oleh *Block Producer* (BP) valid. Setelah final, *Block Producer* (BP) akan secara permanen merekam *blockchain* secara keseluruhan,

mendapatkan *schedule* atau jadwal untuk memproduksi *block* dari *layer* kedua (DPoS) untuk menentukan *block* mana yang tertanda atau *signed* dengan benar dari *Block Producer* (BP). Sebagai sistem yang terdistribusi *blockchain* EOS, juga tidak luput dari *Byzantine general problem*, dalam *Byzantine fault*. Salah satu *node/block* yang menyebabkan kesalahan atau disinformasi akan merusak keseluruhan sistem *blockchain* (Baliga, 2017). Selain melakukan sinkronisasi blok dengan rantai utama, *Native Consensus Model* atau aBFT (*Asynchronous Bizantine Fault Tolerance*) bekerja untuk mencegah *byzantine fault*, yang dapat merusak rantai pada blok. ABFT juga memiliki *impact* yang tidak besar pada performa sistem konsensus (cepat) seperti pada protokol *blockchain* lain yaitu *Hashgraph* (Hassani H., dkk, 2019).

Delegated Proof of Stake (DPoS) sebagai bagian perlindungan kedua untuk menghindari *byzantine fault*, bekerja dengan melakukan voting kepada *stakeholders* (yang dimaksud *stakeholders* adalah *Block Producer* (BP)) untuk memverifikasi sebuah *block* agar bisa masuk ke dalam *chain* atau rantai. Proses *consensus* pada EOS, diawali dengan voting yang menggunakan algoritma *DPoS*. Selanjutnya *block* akan diproduksi atau di validasi dengan menggunakan *native consensus* atau aBFT (EOSIO Documentation, 2020).

Setiap *block* pada EOSIO sebelum menjadi final, dan bersifat *immutable* ada 3 tahap yang harus dilalui atau dinamakan *Block Lifecycle*. *Block Lifecycle* pada EOSIO terdiri dari: *production, validation,* dan *finality*. Tahap *Production* adalah tahap BP atau *Block Production* harus memproduksi *block* secara terus menerus. Tiap *block* akan diproduksi dalam waktu *interval* = 500ms (*millisecond)*. Di tahap

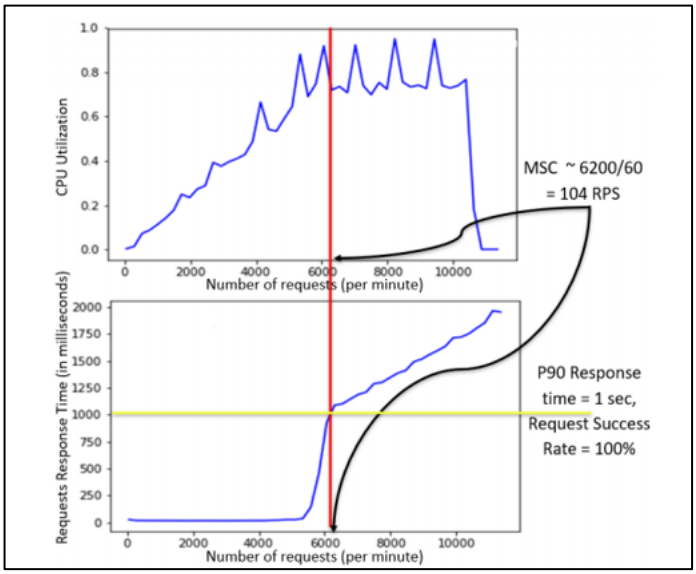
production block akan melewati beberapa tahap *apply, finalize, sign,* dan *commit*. *Tahap apply* merupakan tahap *block* akan dibentuk, di validasi dan akan di *push* ke *chain* utama *blockchain*. *Finalize* adalah tahap sebelum *block* bisa dilakukan *sign*, *block header* akan melakukan finalisasi validasi kriptografi sebelum dilakukan *process sign*. *Sign block* merupakan proses *block* akan di finalisasi dan diberi tanda oleh BP atau *Block Producer.* Terakhir adalah *commit block*, pada tahap ini *block* akan di *push* ke *local chain* (EOSIO Documentation, 2020). Pada penelitian ini, dikarenakan *block* hanya akan berjalan di *local*, *block* tidak akan melewati tahap *validation and finality.*



Gambar 2.1 *Eos transaction lifecycle* (EOSIO Documentation, 2020)

2.2 Request Per Minute (RPM)

Salah satu metode untuk melakukan *benchmarking* pada *web server*, salah satunya adalah *request rate* atau *number of request* atau banyaknya koneksi yang dapat di-*handle* pada suatu satuan waktu (Haiyong, dkk, 2002). *Request per minute* merupakan banyaknya *request* yang dapat di-*handle* oleh *web service*, dalam waktu satu menit.



Gambar 2.2 Pe*model*an benchmark dalam kapasitas *microservice* (Jindal, dkk., 2019)

Request Per Minute dibandingkan dengan utilitas dari *resource* yang dimiliki (*CPU*), kapasitas sistem *microservice* dapat diketahui. Utilisasi *resource* selalu berubah-ubah, tergantung dari *workload* pada sistem, ketika *workload* bertambah, otomatis utilisasi *resource* (*CPU)* juga semakin meningkat. Jika melewati kapasitas *resource*, maka *request* yang dikirim oleh *user* akan diproses lebih lama. Jumlah maksimal *request* yang diproses secara optimal ini yang dinamakan *Microservice Capacity* (MSC) (Jindal, dkk., 2019).