ONLINE MILIS ANALISIS

Bayu Kusuma Wijaya [1] , Budi Susanto[2], Willy Sudiarto R[3]

Abstrak:

Mailing list atau milis adalah sarana komunikasi melalui email dimana tiap anggotanya bisa belangganan dan berpartisipasi didalamnya. Setiap kali ada thead atau reply maka email tersebut akan dikirim semua anggota milis. Anggota milis yang menerima surat tersebut dapat mereply, mengirim thread topik baru ataupun hanya sekedar membaca saja.

Dalam suatu milis lalu lintas informasinya bergantung pada seberapa aktif anggota-anggota milis tersebut. Keaktifan anggota suatu milis sebenarnya dapat diukur dengan cara melihat seberapa sering anggota milis tersebut membuat thread atau mereply email dari para anggota milis lainnya. Hal ini dapat dilakukan dengan menganalisis email yang datang dari group milis kita. Untuk itu aplikasi yang dibuat harus dapat menganalisis tingkat keaktifan suatu group milis dan para anggotanya secara otomatis.

Kata Kunci : *Milis, Thread, Email*

1. Pendahuluan

Mailing list atau milis merupakan sarana komunikasi melalui email dimana tiap anggotanya bisa belangganan dan berpartisipasi didalamnya. Setiap kali ada thread atau reply maka email tersebut akan dikirim semua anggota milis. Karena dalam milis, yang terjadi benar-benar hanya saling mengirim email, para anggota milis sulit untuk mengetahui seberapa aktifkah milis ini, seberapa aktif juga anggota-anggota milis tersebut, dan juga untuk mengetahui apa saja topik-topik yang ada.

Keaktifan anggota suatu milis dapat diukur dengan cara melihat seberapa sering anggota milis tersebut membuat topik baru atau membalas email dari para anggota milis lainnya. Hal ini dapat dilakukan dengan menganalisis email yang datang dari group milis kita.

Penulis akan membuat suatu aplikasi yang menerima *file mbox* suatu *email* yang sudah terdaftar dalam suatu *milis* dan menganalisis keaktifan *milis* tersebut beserta keaktifan para anggotanya dan juga mendeteksi *threading topik-topik* yang ada pada milis tersebut. Keaktifan anggota yang dimaksud disini adalah seberapa sering anggota *milis* memposting topik baru atau me*reply posting* di *milis*. Dan yang dimaksud dengan keaktifan *milis* disini adalah seberapa sering muncul topik baru dan *reply* baru dalam periode tertentu.

2. Landasan Teori

2.1 E-Mail

Kata *E-mail* sering ditulis sebagai *e-mail*, *email*, atau *eMail*.

Email tediri dari dua bagian utama :

- *Header* : Berisi informasi mengenai subjek, pengirim, penerima, dan informasi-informasi lain mengenai *email* tersebut.
- *Body* : Berisi pesan itu sendiri.

Tiap pesan hanya memiliki satu *header*. Pada bagian *header* ini biasanya berisi beberapa *field* yaitu (Jacob, 2002) :

- *From* : Berisi alamat *email* pengirim, dan kadang berisi nama pengirim juga
- *To* : Berisi alamat *email* penerima, alamat *email* penerima bisa lebih dari satu. Pada *field* ini kadang disertakan juga nama dari penerima *email*.
- *Subject* : Ringkasan singkat mengenai isi dari pesan.

---

[1] *Bayu Kusuma Wijaya, Mahasiswa Teknik Informatika, Fakultas Teknik, Universitas Kristen Duta Wacana. Email : bayu.k.w@gmail.com*

[2] *Budi Susanto, S.Kom., M.T., Dosen Teknik Informatika, Fakultas Teknik, Universitas Kristen Duta Wacana. Email : budsus@gmail.com*

[3] *Willy Sudiarto R, S.Kom, M.Cs., Dosen Teknik Informatika, Fakultas Teknik, Universitas Kristen Duta Wacana. Email : willysr@gmail.com*

- *Date* : Waktu lokal kapan pesan tersebut ditulis.

Tetapi *field Form* dan *field To* tidak selalu benar-benar berisi alamat email pengirim dan penerima. Daftar pengiriman yang sebenarnya diatasi oleh protokol *SMTP*, tidak diambil dari *header email*.

Beberapa *header field* yang lain antara lain (Jacob, 2002) :

- *CC* : *Carbon Copy*
- *BCC* : *Blind Carbon Coby*
- *Reply-To* : Biasanya berisi alamat *email* kemana *email* ini harus *direply*.
- *In-Reply-To* : Berisi *Message ID* dari *email* yang *direply* oleh *email* ini.
- *References* : Berisi *Message-ID* dari *email-email* yang merupakan parent dari *email* ini.

2.2 Milis

Milis atau *Mailing List* merupakan pendistribusian *email* secara otomatis, dimana tiap anggotanya bisa belangganan dan berpartisipasi didalamnya. Apabila kita ikut suatu *milis* maka tiap kali ada *posting* atau *reply* maka *email* tersebut akan dikirim semua anggota *milis* (Dorman, 1999). Anggota *milis* yang menerima surat tersebut dapat *mereply*, mengirim *email* baru ataupun hanya sekedar membaca saja.

Proses dimana seseorang anggota *milis mereply* sebuah *email* dan saat *email* tersebut akan didistribusikan ke setiap *mailbox* yang ada dalam daftar *milis* dijalankan oleh suatu program yang bernama *Mailing List Manager (MLM)* atau *Mail Server*. Program tersebut biasanya berjalan secara *full-automatic* atau *semi-automatic* dimana *milis* memiliki sebuah alamat *reflektor* yang dipasang pada *server* yang menerima *email*. Pesan yang dikirim ke alamat *reflektor* diproses oleh program dan dikirimkan ke semua alamat *email* yang menjadi anggota dari *milis* tersebut. Kadang-kadang diciptakan pula alamat tambahan yang dipergunakan untuk mengirimkan perintah ke *server* untuk berhenti berlangganan *milis* atau meneruskan kembali berlangganan *milis*. Apabila seseorang anggota berhenti berlangganan *milis* maka *server* akan berhenti mengirimkan *email* ke alamat anggota tersebut. Sederhananya *Mailing list* merupakan sebuah daftar alamat-alamat *email*. Penyedia layanan *milis* yang cukup terkenal antara lain *googlegroups.com* dan *yahoogroups.com*.

2.3 Algoritma Jamie Zawinski

Algoritma Jamie Zawinski adalah *algoritma* yang digunakan untuk membangun *thread* dari pesan-pesan yang ada. *Algortima* ini mengelompokan pesan-pesan menjadi satu dalam *relasi parent/child* berdasarkan pesan yang *mereply* pesan lain (Zawinski, 2002).

Algoritma ini memanfaatkan dua *header* dalam *email*, yaitu :

- *In-Reply-To* → Berisi *Message ID* dari pesan yang *direply* oleh pesan ini.
- *References* → Berisi *Message ID* dari pesan yang mempunyai hubungan dengan pesan ini.

Berikut ini alur *algoritma Jamie Zawinski* :

- Buat sebuah *class Message* yang berisi *subjek* dari *email*, *message id*, dan *referensi*. *Referensi* adalah *array Message ID* dari pesan *parent*.
- Buat sebuah *class Container* yang berisi *class Message* dan 3 buah *pointer Container* yang diberi nama *parent*, *child*, dan *next*. *Pointer child* menunjuk ke *first child* dari *Container* tersebut sedangkan *next* menunjuk ke pesan lain yang sama-sama merupakan *child* dari pesan *parent*. *Pointer next* bisa saja *null* apabila memang *parentnya* hanya punya satu *child* saja.
- *Referensi* diisi dengan *message ID*, dalam *header references* atau *In-Reply-To*. Jika kedua *header* mempunyai isi maka ambil *message ID* yang pertama pada header *In-The-Reply* dan tambahkan pada *header references*.
- Buat *id_table* yang merupakan *tabel hash* yang berisi *message ID* dan *Container*.
- Untuk tiap pesan yang ada:

- Jika *id_table* berisi *Container* kosong untuk *ID* pada pesan ini, maka simpan pesan pada *Container* tersebut
- Jika tidak maka buat *Container* baru untuk membungkus pesan ini, lalu tambahkan *Container* dan *message id* pada *id_table*.

- Untuk tiap elemen yang ada pada *field references* pada pesan:
  - Cari *Container* yang berisi *message ID* tersebut, jika ditemukan gunakan *Container* tersebut. Jika tidak ditemukan maka buat *Container* baru dengan isi *message null* dan jangan lupa tambahkan *Container* tersebut di *id_table*.
  - Hubungkan *Container* yang tertera di *references* berdasarkan *references header*
    a. Jika *Container* tersebut sudah terhubung maka jangan ubah *link* yang ada
    b. Jangan tambahkan *link* jika dengan menambahkan *link* maka terjadi *loop*. Caranya sebelum kita menambahkan A→ B, cari dulu *children* pada B apakah ada A. dan juga cari *child* A apakah ada B. Jika memang sudah ada maka jangan tambahkan *link*.
- Set *parent* dari pesan ini berdasarkan *elemen* terakhir pada *refrenceses*. Jika pada pesan *parent* sudah ada maka replace dengan yang baru. Pesan bisa saja tidak punya *parent* jika *reference* pesan tersebut kosong.
- **Temukan pesan-pesan *root***
  Jelajahi *table_id* dan kelompokan semua *Container* yang tidak memiliki *parent*.
- **Buang *Container* kosong.**
  Jelajahi satu persatu semua *Container root* sampai ke *leafnya*.
  Untuk tiap *Container*:
  - Jika ada *Container* kosong yang tidak memiliki *child*, maka hapus *Container* tersebut.
  - Jika *Container* kosong tetapi memiliki *child* maka hapus *Container* ini dan pindahkan *child* nya pada *level Container* ini. Tetapi jangan pindahkan *child* jika setelah kita pindahkan maka *child* menjadi *root*, kecuali jika hanya memiliki satu *child*.
- **Kelompokan *root* berdasarkan *subjek*nya**
  Jika ada dua *root* yang memiliki subjek sama maka gabungkan menjadi satu. Hal ini dapat sedikit menolong jika ada pesan yang tidak memiliki *header references* agar tetap di kelompokan berdasarkan *threadnya*.
  - Buat *tabel hash* baru yang berisi *subject string* dan *Container*.
  - Untuk tiap *container* pada kelompok *root*:
    Temukan *subjek* pada *tree*:
    a. Jika ada pesan pada *Container* maka gunakan subjek pada pesan tersebut.
    b. jika tidak ada pesan pada *Container*, cari subjek pada pesan yang merupakan *childnya*.
    c. Hapus kata ``Re:", ``RE:", ``RE[5]:", ``Re: Re[4]: Re:" dan seterusnya.
    d. Jika *subjeknya* hanya *string* kosong "", lewati saja *Container* ini.
- Sekarang *tabel subjek* sudah terisi dengan *tiap subjek* yang ada pada *kelompok root*.

2.4 MBox

Mbox adalah salah satu *format file* yang digunakan untuk menampung sekumpulan pesan *email*. Semua pesan yang berada di dalam *Mbox* disatukan dan disimpan dalam satu *file text*. Tiap permulaan pesan ditandai dengan *"From "* diikuti dengan serangkaian baris yang tidak diawali dengan *"From "* lalu diakhiri dengan sebuah baris kosong ditambahkan untuk menandai akhir dari suatu *email*.

2.4.1 Proses penambahan pesan ke dalam file Mbox

Pertama-tama program akan membuat baris *"from "* dan memberikan alamat pengirim dan waktu saat ini. Jika alamat pengirim kosong maka program akan mengisinya dengan *MAILER-DAEMON*. Jika alamat pengirim berisi spasi, tab, atau baris baru maka program akan

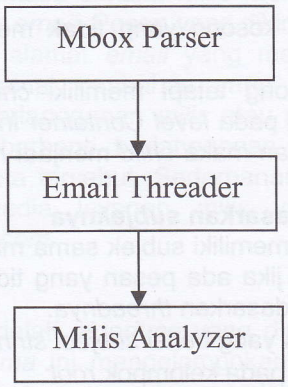
menggantikannya dengan tanda penghubung. Lalu program mengkopi semua pesan dan menambakan tanda > didepan tiap baris yang mempunyai awalan *"from "* untuk memastikan tidak ada baris dengan awalan *"from "* terbentuk. Lalu program menambahkan satu baris kosong untuk menandai akhir dari suatu pesan. Jika didalam pesan ditemukan satu baris kosong, maka program akan menuliskannya sebagai dua baris kosong.

2.4.2 Proses membaca pesan dari file Mbox

Program membaca satu per satu baris dalam *Mbox* mencari baris yang diawali *"From "*. Tiap baris yang berawalan *"From "* menandai awal dari sebuah pesan. Saat program pembaca menemukan baris *"From "* ini maka program akan mengambil alamat pengirim dan tangal pengiriman dari baris *"From "* tersebut. Lalu program akan terus membaca sampai baris *"From "* berikutnya atau akhir file. Program akan menghapus baris kosong penanda akhir pesan dan juga menghapus tanda > yang tadinya ditambahkan.

3. Arsitektur Sistem

Dalam sistem terdapat 3 komponen utama yaitu *Mbox parser*, *email threader*, dan *milis analyzer*. Sistem menerima inputan berupa file *Mbox*, kemudian *Mbox parser* akan mengenali tiap *email* yang ada di dalam *file Mbox*. Semua data *email* akan dimasukan kedalam suatu *database* penampung sementara. Setelah semua *email* sudah dimasukan ke dalam database, *email threader* akan mengelompokan *email* berdasarkan *relasi parent-child* dimana *email* yang satu *mereply* dan atau *direply email* lainnya. Sistem akan menganalisis *email* yang sudah di *threading* tadi dan membangun data statistik mengenai group milis. Ilustrasinya dapat dilihat pada Gambar 3.1.



Gambar 1. Arsitektur Sistem

3.1 Mbox Parser

Mbox Parser akan melakukan *parsing* terhadap *file mbox*. *Mbox Parser* akan mengambil data-data dalam *header email* seperti *header to*, *header from*, *header in-reply-to*, *header references*, *header date* yang berisi tanggal dan jam pengiriman *email*, dan *header subject* dari tiap *email* yang ada didalam file *Mbox*. Semua data tersebut akan disimpan di dalam *database* sistem untuk keperluan *threading* dan *analisis* lebih lanjut.

3.2 Email Threader

Email Threader akan melakukan *threading* terhadap data *email* yang sudah disusun oleh *mbox parser*. *Email threader* akan menghasilkan data *root email* dan *data relasi email*. Apabila *email root* tidak didapatkan maka akan dibuat suatu *dummy email* dalam *tree email* yang dihasilkan. Untuk melakukan *threading*, penulis akan mengimplementasikan *algoritma jamie zawinski* (2002).

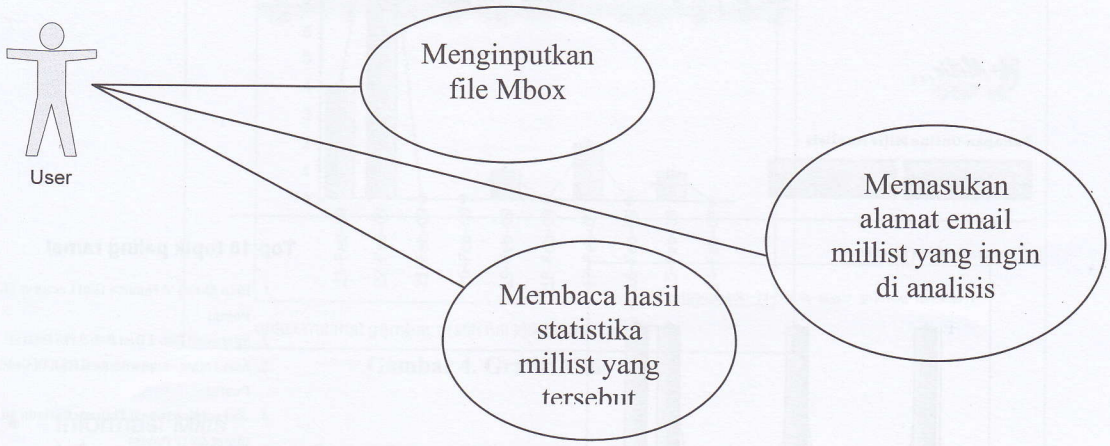
3.3 Milis Analyzer

Milis Anaylzer akan membangun data keaktifan *milis* dan anggota-anggotanya sebagai mana yang ada pada spesifikasi sistem. Untuk mengetahui alur *posting* dan *reply milis* tersebut

maka *email analyzer* memanfaatkan data *root email* dan data *relasi* antar *email* yang sudah disusun oleh *email threader*.

3.4 Diagram *Use Case*

Berdasarkan uraian analisis kebutuhan yang telah dijelaskan, berikut ini adalah bentuk diagram *use case* dari aplikasi yang akan dibuat :



Gambar 2. Use Case Diagram

User akan memasukan *file mbox* ke dalam sistem. Kemudian User harus memasukan alamat *email milis* yang ingin dianalisis. Kemudian sistem akan menganalisis semua email yang berasal dari *grup milis* tersebut. Setelah itu sistem akan menyusun data statistika sebagaimana yang ada pada spesifikasi sistem diatas.

4. Implementasi Sistem

Sistem akan membaca per baris dan akan mengenali bahwa suatu baris adalah start dari suatu header email apabila menemui baris yang diawali dengan "From " (kata from dengan F kapital dan diikuti sebuah spasi).

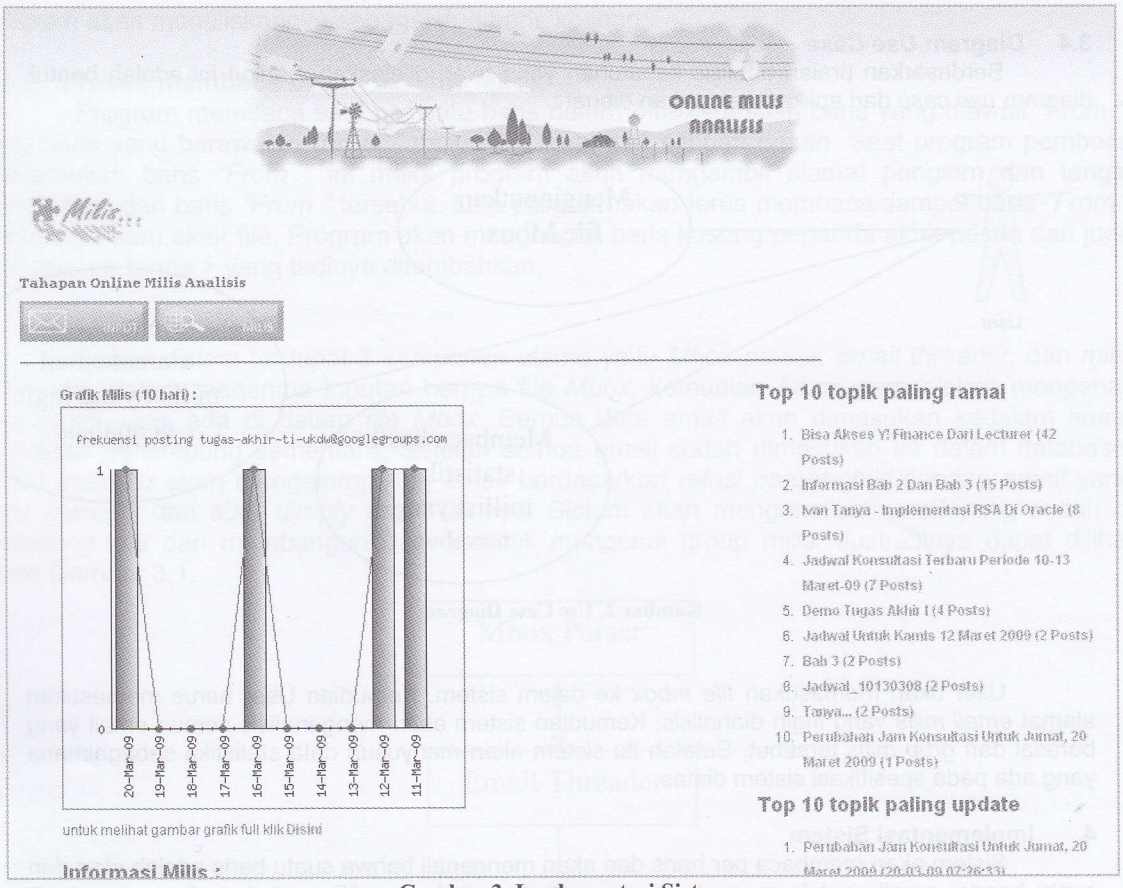
Setelah *header email* dikenali maka sistem akan melanjutkan pembacaan per baris sambil mengamati apakah diawali dengan *keyword-keyword* sebagai berikut :

In-Reply-To: .....
References: .....
Date: .....
Message-ID: .....
Subject: .....
From: .....
To: .....

Apabila menemui *keyword* tersebut sistem akan menyimban data-data teks setelah tanda : (titik dua) untuk dianggap sebagai informasi *header email*. Sebelum dimasukan ke dalam *database* maka dicocok dahulu apakah *header To* berisi salah satu alamat *milis* yang diinputkan *user*. Apabila tidak maka informasi di buang dan sistem akan melanjutkan untuk pencarian awal *header* berikutnya. Khusus saat alamat *milis* diisi *keyword* "semuanya" maka *data header* akan langsung disimpan.

Sistem akan mengenali bahwa baris saat ini adalah suatu *body* dengan membaca apakah ada baris kosong. Apabila ada baris kosong maka sistem akan berhenti mengambil informasi header dan melanjutkan membaca baris sampai penanda *header email* yang baru muncul.

Pada *header from* selain mengambil alamat *email*, sistem juga akan berusaha mengambil *alias name* sang pemilik alamat *email* dengan cara menghapus alamat *email* dan keyword "from:" sehingga menyisakan alias name dan menyimpannya ke dalam database.



Gambar 3. Implementasi Sistem

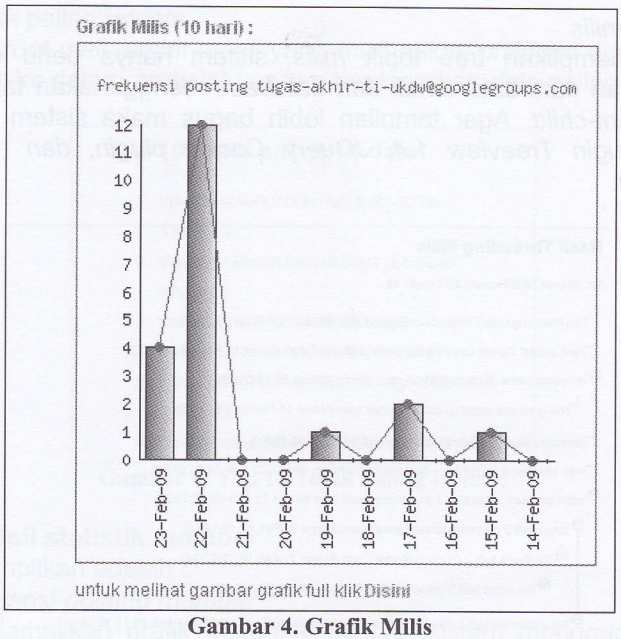
4.1 Menampilkan hasil statistik milis

Data yang ditampilkan dalam *milis* adalah

- Grafik frekuensi posting *milis*

  Untuk menampilkan grafik frekuensi *posting* sistem menggunakan *PHP Graphing Library v2.02* dari www.ebrueggeman.com. Data diambil dari database. Dengan algoritma sebagai berikut:

  1. Ambil *email* dengan *date* terbaru simpan ke variabel *lastday* dengan membuang informasi detik, menit, dan jam.
  2. Ambil *email* dengan *date* terlama simpan ke variabel *firstday* dengan membuang informasi detik, menik, dan jam.
  3. Ambil rentang jumlah hari antara *lastday* dan *firstday* dan *generate* hari-hari antara *firstday* dan *lastday* lalu jadikan tanggal tersebut sebagai index *array* dengan nilai *default* 0.
  4. Baca kembali surat satu per satu lalu ambil data *date*nya lalu tambahkan nilai pada *array* dimana *index*nya merupakan tanggal *email* yang sedang diperiksa.
  5. Apabila jumlah data pada *array* kurang dari 10 tambahkan beberapa *dummy index* dengan nilai 0 pada hari setelah *lastday*. Untuk gambar *preview*, potong data sampai 10 hari saja apabila data lebih dari 10.
  6. Gunakan *library PHP Graphing Library v2.02* untuk meng*enerate* gambar grafik.



Gambar 4. Grafik Milis

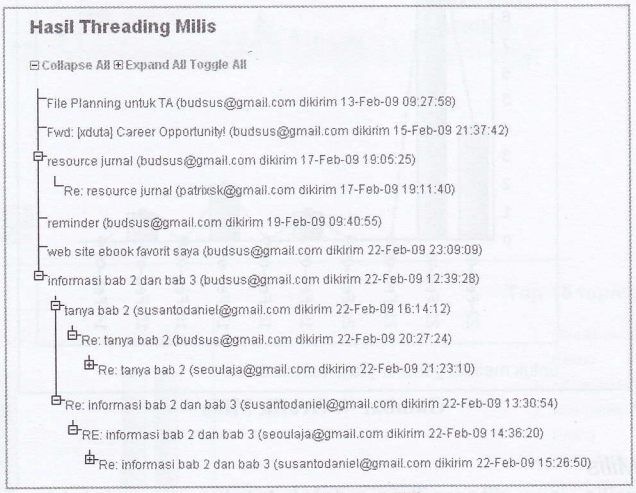
- Informasi *Milis*

  Informasi *milis* yang ditampilkan adalah total *posting*, total topik, total *reply*, member aktif yang ditemukan, waktu rata-rata *posting*, wakru rata-rata topik baru, dan waktu rata-rata *reply*.

  1. Untuk total *posting* sistem hanya menghitung total jumlah *email* yang ada pada *database*.
  2. Untuk total topik sistem menghitung total *email* yang tidak mempunyai *parent*.
  3. Total *reply* didapatkan dengan menghitung total *email* yang mempunyai *parent*.
  4. *Member* aktif yang ditemukan didapatkan dengan menghitung jumlah alamat *email* yang ada pada tabel *email*.
  5. Waktu rata-rata *posting* didapatkan dengan cara menghitung mengurangi *timestamp email* terakhir dengan *email* pertama lalu dibagi dengan jumlah *posting*. Ubah *timestamp* yang sudah dibagi tersebut kedalam hari, jam, menit, dan detik.
  6. Waktu rata-rata topik didapatkan dengan cara menghitung mengurangi *timestamp email* terakhir yang tidak mempunyai *parent* dengan *email* pertama yang tidak mempunyai *parent* lalu dibagi dengan jumlah *posting*. Ubah *timestamp* yang sudah dibagi tersebut ke dalam hari, jam, menit, dan detik.
  7. Waktu rata-rata *reply* didapatkan dengan cara menghitung mengurangi *timestamp email* terakhir yang mempunyai *parent* dengan *email* pertama yang mempunyai *parent* lalu dibagi dengan jumlah *posting*. Ubah *timestamp* yang sudah dibagi tersebut kedalam hari, jam, menit, dan detik.

Gambar 5. Informasi Milis

- *Tree* topik *milis*
  Untuk menampilkan *tree* topik *milis*, sistem hanya perlu mengambil data dan menampilkan data-data *email* dari database menggunakan tabel *container* sebagai *relasi parent-child*. Agar tampilan lebih bagus maka sistem menggunakan *library JQuery Plugin Treeview 1.4, JQuery Cookie plugin, dan JQuery 1.2.2b2* dari *jquery.com*.



Gambar 6. Tree Milis

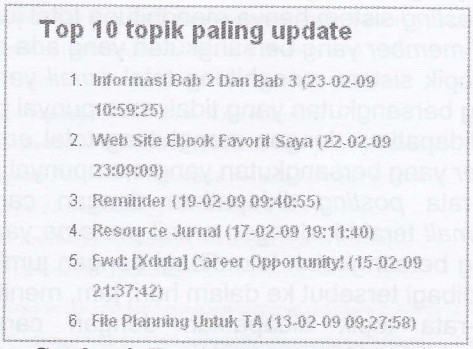
- *Member* yang aktif
  Untuk menampilkan data *member*, sistem mengambil informasi alamat *email* dari tabel *email*, dan untuk tiap alamat *email* sistem juga mencari jumlah *email* yang dikirim oleh alamat *email* tersebut.

Gambar 7. Member yang aktif

- Top 10 topik paling ramai
  Untuk tiap *root mail*, hitung jumlah total anaknya dan simpan ke dalam array lalu urukan berdasarkan nilai yang tertinggi.

Gambar 8. Top 10 Topik Paling Ramai

- Top 10 topik paling *update*
  Untuk tiap *root mail*, ambil *email* yang merupakan *child*nya dengan *date* paling baru dan simpan ke dalam *array* lalu urutan berdasarkan *date* paling baru.

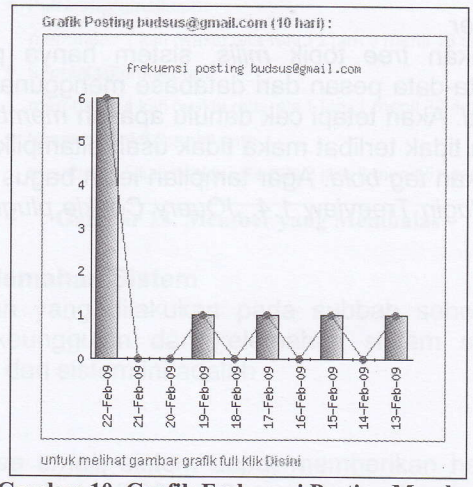


Gambar 9. Top 10 Topik Paling Update

4.2 Menampilkan hasil statistik *member*

Data yang ditampilkan adalah :

- Grafik frekuensi *posting member*
  Untuk menampilkan grafik frekuensi *posting* sistem menggunakan PHP *Graphing Library v2.02* dari www.ebrueggeman.com. Data diambil dari database. Dengan algoritma sebagai berikut:
  1. Ambil *email* dengan *date* terbaru yang dikirim oleh *member* yang bersangkutan simpan ke *variabel lastday* dengan membuang informasi detik, menik, dan jam.
  2. Ambil *email* dengan *date* terlama yang dikirim oleh *member* yang bersangkutan simpan ke *variabel firstday* dengan membuang informasi detik, menik, dan jam.
  3. Ambil rentang jumlah hari antara *lastday* dan *firstday* dan *generate* hari-hari antara *firstday* dan *lastday* lalu jadikan tanggal tersebut sebagai index *array* dengan nilai *default* 0.
  4. Baca kembali surat satu per satu lalu ambil data *date*nya lalu tambahkan nilai pada *array* dimana *index*nya merupakan tanggal *email* yang sedang diperiksa.
  5. Apabila jumlah data pada *array* kurang dari 10 tambahkan beberapa *dummy index* dengan nilai 0 pada hari setelah *lastday.* Untuk gambar *preview*, potong data sampai 10 hari saja apabila data lebih dari 10.
  6. Gunakan library *PHP Graphing Library v2.02* untuk meng*generate* gambar grafik.



Gambar 10. Grafik Frekuensi Posting Member

- Informasi *Member*

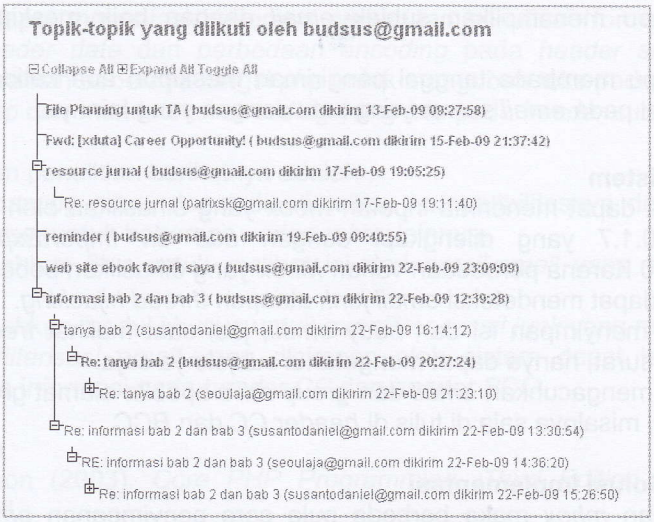
Informasi *milis* yang ditampilkan adalah nama alias, total *posting*, total topik, total *reply*, waktu rata-rata *posting*, wakru rata-rata topik baru, dan waktu rata-rata *reply*.

1. Data nama alias dapat diambil dari tabel nama dimana pada kolom *email* berisi alamat *email* yang bersangkutan.
2. Untuk total *posting* sistem hanya menghitung total jumlah *email* yang dikirim oleh alamat *email member* yang bersangkutan yang ada pada *database*.
3. Untuk total topik sistem menghitung total *email* yang dikirim oleh alamat *email member* yang bersangkutan yang tidak mempunyai *parent*.
4. Total *reply* didapatkan dengan menghitung total *email* yang dikirim oleh alamat *email member* yang bersangkutan yang mempunyai *parent*.
5. Waktu rata-rata *posting* didapatkan dengan cara menghitung mengurangi *timestamp email* terakhir dengan email *pertama* yang dikirim oleh alamat *email member* yang bersangkutan lalu dibagi dengan jumlah *posting*. Ubah *timestamp* yang sudah dibagi tersebut ke dalam hari, jam, menit, dan detik.
6. Waktu rata-rata topik didapatkan dengan cara menghitung mengurangi *timestamp email* terakhir yang tidak mempunyai *parent* dengan *email* pertama yang tidak mempunyai *parent* yang dikirim oleh alamat *email member* yang bersangkutan lalu dibagi dengan jumlah *posting*. Ubah *timestamp* yang sudah dibagi tersebut kedalam hari, jam, menit, dan detik.
7. Waktu rata-rata *reply* didapatkan dengan cara menghitung mengurangi *timestamp email* terakhir yang mempunyai *parent* dengan *email* pertama yang mempunyai *parent* yang dikirim oleh alamat *email member* yang bersangkutan lalu dibagi dengan jumlah *posting*. Ubah *timestamp* yang sudah dibagi tersebut kedalam hari, jam, menit, dan detik.

Gambar 11. Informasi Member

- *Tree* topik *member*

  Untuk menampilkan *tree* topik *milis*, sistem hanya perlu mengambil data dan menampilkan data-data pesan dari database menggunakan tabel *container* sebagai relasi *parent-child*. Akan tetapi cek dahulu apakah *member* terlibat pada *thread* topik tersebuk, apabila tidak terlibat maka tidak usah ditampilkan. Pada *email* yang dikirim member tambahkan *tag bold*. Agar tampilan lebih bagus maka sistem menggunakan *library JQuery Plugin Treeview 1.4, JQuery Cookie plugin*, dan *JQuery 1.2.2b2* dari *jquery.com*.



Gambar 12. Tree Topik Member

- *Member* yang membalas

  Untuk menampilkan *member* yang mebalas *topik* atau *reply* yang dikirim oleh *member* yang bersangkutan sistem menggunakan algoritma berikut ini:

  1. Ambil semua *member milis* dari tabel *email* kecuali *email* yang bersangkutan dan untuk tiap *member* tersebut baca satu per satu *email* yang dikirimkan olehnya. Dan kerjakan langkah 2.
  2. Jika *email* tersebut merupakan salah satu *child* dari email yang dikirim oleh alamat *member* yang bersangkutan maka simpan data *date*nya ke *array timesand*, dan tambahkan *counter* jumlah *reply*nya. Ambil data *date* terbaru dan terlama dari *array time sand*, dan jika data yang ada pada *timesand* lebih dari 2 maka tampilkan rata-rata waktu *reply*nya dengan mengurangi *timestamp date* pertama dan terakhir dan membaginya dengan jumlah *email* yang ada pada *counter*. Tampilkan pula jumlah *reply* berdasarkan data yang ada pada *counter email*.

Gambar 13. Member yang Membalas

4.3 Keunggulan dan Kelemahan Sistem

Dari hasil penelitian yang dilakukan pada subbab sebelumnya penulis menemukan beberapa hal mengenai keunggulan dan kelemahan sistem secara keseluruhan. Adapun kelemahan dan keunggulan dari sistem ini adalah :

4.3.1 Keunggulan Sistem

a. Dilihat dari hasil *tree email*, sistem dapat memberikan hasil yang cukup baik dengan Algoritma *Jamie Zawinski* (2002). Sistem juga dapat menggambarkan pada *tree* hubungan *parent-child* beberapa *email* yang mempunyai *parent* yang sama meskipun pada *mbox* tidak ditemukan *email parent* tersebut.

b. Sistem mampu menampilkan subjek *email* dengan baik meskipun subjek *email* di *encode*.
c. Sistem mampu membaca tanggal pengiriman meskipun ada ketidaksamaan penulisan format tanggal pada *email* server yang satu dengan yang lainnya.

4.3.2 Kelemahan Sistem

a. Sistem hanya dapat menerima inputan *mbox* yang dihasilkan oleh *Mozzila Thunderbird version* 2.0.0.1.7 yang dilengkapi dengan add on *ImportExportTools(Mboximport enhanced)* 2.0 Karena perbedaan varian *Mbox* yang dihasilkan beberapa aplikasi *email*.
b. Sistem tidak dapat mendeteksi *email junk* ataupun *thread hijacking*.
c. Sistem tidak menyimpan isi dari *body email*, jadi saat melihat *tree thread* tidak dapat membaca isi surat, hanya dapat mengetahui subjeknya saja.
d. Sistem akan mengacuhkan *email* dari group *milis* apabila alamat group *milis* tidak ditulis di *header TO*, misalnya saja di tulis di *header CC* dan *BCC*.

4.4 Kendala dan Solusi Implementasi

Berbeda varian *mbox* maka berbeda pula cara penyimpanan *email* ke dalam *mbox*. Meskipun varian sama bisa saja aplikasi email menambahkan atau merubah informasi tertentu pada header saat membuat *file mbox*. Jadi sistem harus diperuntukan secara spesifik untuk membaca aplikasi *mbox* dari aplikasi tertentu.

Informasi pada header *email* yang berbeda-beda pada beberapa *server mail* membuat pembacaan informasi sedikit lebih sulit. Misalnya saja beberapa *server email* ada yang menggunakan *encoding* pada subjek *email*nya. Oleh karena itu sistem harus men*decode* subjek sebelum ditampilkan pada *tree*. Penulisan format tanggal dan waktu yang berbeda-beda antara *server* satu dengan yang lainya juga mempersulit pembacaan informasi tanggal pengiriman surat. Untuk itu sistem menggunakan fungsi pada php untuk merubah string berisi informasi tanggal menjadi *timestamp unix*. Ditemui juga beberapa *header reference* yang isinya dipisahkan dengan *newline*. Hal ini cukup mempersulit proses *parsing* yang dilakukan per baris. Sistem mengatasi masalah ini dengan menggunakan suatu variabel *flag* khusus dan *regular expression* saat mengambil informasi *references* untuk membedakan baris berikutnya setelah *references* adalah informasi *references* yang terpisah atau merupakan *header baru*.

Inputan *Mbox* user tidak dijamin berisi *email-email* secara utuh. Oleh karena itu sistem terkadang perlu membuat *dummy email* (*container kosong*) untuk membangun *tree email*. Sistem juga perlu menghapus *dummy email* yang tidak terpakai agar didapatkan *email* yang lebih baik.

Algoritma *jamie zawinski* (2002) sebenarnya tidak berhenti sampai tahap *pruning*. Setelah *tree* di *pruning*, algoritma *jamie zawinski* (2002) akan mulai mengurutkan relasi *parent-child* mengandalkan kemiripan subjek. Jadi apabila ada 2 surat yang memiliki subjek mirip maka akan direlasikan *parent child*. Akan tetapi hal ini akan mengakibatkan *email junk* yang merupakan *promosi* dengan subjek sama dan dikirim oleh suatu alamat berbeda akan direlasikan *parent child*. Ini tentu akan membuat *email-email junk* tersebut seperti *tree* topik sendiri yang aktif berbalas-balasan. Oleh karena itu penulis mengimplementasikan algoritma *jamie zawinski* (2002) sampai pada tahap *pruning* saja.

5. Kesimpulan

Dari hasil penelitian yang dilakukan maka dapat diambil kesimpulan sebagai berikut :

a. Dengan memparsing data *email* pada *Mbox* dan melakukan *threading email-email* tersebut sistem dapat membangun data statistik keaktifan suatu group *milis* dan para anggotanya.
b. Pemanfaatan algoritma *jamie-zawinski* (2002) dalam menganalisis suatu group *milis* sebenarnya cukup baik. Akan tetapi *email-email spam* yang merupakan promosi justru terlihat seperti *thread* aktif yang berbalas-balasan karena algoritma ini juga merelasikan *parent-child* terhadap *email-email* dengan subjek yang sama. Oleh Karena itu algoritma ini harus dipotong sampai tahap *pruning* saja dan tidak perlu diteruskan sampai tahap *threading* berdasarkan subjek.

c. Sistem dapat mengatasi beberapa perbedaan penulisan format tanggal dan waktu pada *header date* dan *perbedaan encoding* pada *header subject email* dengan memanfaatkan beberapa fungsi *php* untuk mengubah teks pada *header date* menjadi *timestamp* dan mendeteksi *encoding* yang ada pada *header* subjek.

Sebagai saran penelitian berikutnya adalah :

a. Proses *parsing mbox* dapat dioptimalkan kompatibilitasnya dengan menambahkan *mbox parser* untuk beberapa varian *mbox* lainnya.
b. Menambahkan fitur untuk melihat isi dari *email-email* yang ditampilkan oleh *tree email*.
c. Menambahkan modul khusus untuk memfilter *email junk* yang masuk ke dalam *milis*.
d. Proses *filterisasi email* yang disimpan oleh sistem dapat lebih optimal dengan membaca informasi pada *header CC* dan *header BCC*.

6. Daftar Pustaka

[1] Atkinson, Leon (2003). *Core PHP Programming, Third Edition*. New Jersey : PTR Prentince Hall.
[2] Dorman, Steve M. (1999). *Electronic Mailing Lists*. The Journal of School health vol 69.
[3] Palme, Jacob (2002). *Common Internet Message Header Fields*. Sweden : Stockholm University.
[4] Rose, Marshall T. (1993). *The Internet Message : Closing the Book with Electronic Mail*. New Jersey : P T R Prentice Hall, Inc.
[5] Qmail. dalam http://www.qmail.org/man/man5/mbox.htm; 28 September 2008
[6] Zawinski, Jamie, (2002). "Message Threading", dalam http://www.jwz.org/doc/threading.html; 28 September 2008.