Nama : A.Firdaus

Nim : 1924200423

Kelas : MTIR2

Tugas

Pertanyaan : Mengapa Internet melambat ketika jaringan sedang sibuk dari studi kasus diberikan melalui video yang diunggah oleh Dr Richard Mortier

A. Latar Belakang :

Menurut jurnal Ilmu Komputer Universitas California :

Dalam kasus tersebut dimana pertumbuhan internet semakin pesat dan banyak transisi perpindahan akses peneliti menganalsis dengan dua metode yaitu single Connection Behavior dan Parallel Connection Behavior

1. Single Connection Behavior :
   a. Loss Recovery : Bagaimanakah Pemulihan dan efektifitas pemulihan data dalam menghindari saat timeout
   b. Reciever Bottleneck : Berapa Seringkah Kinerja TCP menerima halaman dengan waktu yang terbatas
   c. Ack compression : Seberapa sering terjadi act compression saat kehilangan data
2. Parallel Connection Behavior :
   a. Bagaimana clien melihat jumlah koneksi saat server dibuka
   b. Congestion control: bagaimana reaksi koneksi pararel menerima saat kehilangan data yang disebabkan koneksi timeout
   c. Loss behavior : Bagaimana data yang didistribusikan saat koneksi sedang padat

B. Analisa :

DidalamPenelitan tersebut dijelaskan bagaimana cara mengalisa tentang koneksi TCP saat koneksi sedang sibuk dan seberapa baik pemulihan yang

terjadi. Ada beberapa analisa tentang bagaimana cara memperbaiki data yang diterima saat melakukan koneksi jaringan antara lain :

1. **Single Connection Behavior**

Total connections	16501	
With packet re-ordering	9703	6
With receiver window as	2339	14
Total packets	78216	
During slow-start	66620	85
# of slow-start pkts lost	3545	5
During congestion	11595	15
# of congestion	8218	7
Total retransmissions	8571	
Fast retransmissions	3753	44
Slow-start retransmissions	5981	7
Coarse timeouts	4220	49
Avoidable with SACKs	1871	4
Avoidable with enhanced	1042	25

Pada tabel diatas saat melakukan single connection dimana ukuran data yang panggil lebih besar daripada data yang diterima

2. **Parallel Connection Behavior**

Analisa Koneksi yang terbuang saat melakukan koneksi secara bersamaan

3. **Key Results of Trace Analysis**

Dari penelitian yang dikembangkan dimana koneksi tunggal lebih baik dalam pemulihan pengriman data ketimbang dengan koneksi yang bersamaan

C. Pembahasan

Ada beberapa metode yand dapat dilakukan antara lain :

1. **Application-level Multiplexing**

Solusi tingkat aplikasi menghindari penggunaan beberapa koneksi TCP paralel, dan masalah yang dihasilkan, dengan multiplexing beberapa

aliran data ke koneksi TCP tunggal. Karena TCP hanya menyediakan abstraksi byte-stream tunggal

Tapi ada beberapa kekurangan antara lain :

a. Mereka membutuhkan tambahan dan diperlukan server dan klien yang baik.
b. Mereka tidak mengizinkan multiplexing transfer yang dilakukan oleh lebih dari satu aplikasi.
c. Multiplexing melalui koneksi TCP tunggal memperkenalkan kopling yang tidak diinginkan antara transfer data yang secara logis independen

2. TCP-INT Implementation

Menggambarkan penerapan kontrol kemacetan terintegrasi dan skema pemulihan kehilangan yang hanya mengubah TCP pada pengirim. Untuk setiap host yang terkait dengan satu mesin, tumpukan TCP / IP membuat struktur (Gambar 9) untuk menyimpan informasi tentang komunikasi apa pun. Struktur baru ini memungkinkan kontrol kemacetan bersama yang diinginkan dan pemulihan kerugian dengan menyediakan satu titik koordinasi untuk semua koneksi ke host tertentu. Struktur chost berisi variabel-variabel standar TCP yang terkait dengan pemeliharaan jendela kongesti TCP (cwnd, ssthresh dan count). Struktur ini juga memperkenalkan beberapa variabel baru untuk membantu dalam kontrol kemacetan (ownd, decr_ts) dan variabel lain untuk mendukung pemulihan kerugian terintegrasi (pkts []). Di subbagian berikut, kami menjelaskan bagaimana berbagai rutinitas TCP menggunakan dan memperbarui informasi baru ini. Rutin pengiriman data baru: Ketika suatu koneksi ingin mengirim suatu paket, ia memeriksa untuk melihat apakah jumlah byte yang sudah ada dalam "pipa" dari pengirim (ownd) lebih besar daripada ukuran "pipa" yang diinginkan (cwnd). Jika tidak, koneksi akan menyiapkan paket yang akan dikirim dengan menambahkan entri pada ekor daftar paket yang beredar. Entri ini berisi ukuran nomor urut dan stempel waktu dari paket yang dikirimkan. Ketika paket dikirim, koneksi menambah kepemilikan

dengan ukuran paket. Kami menggunakan penjadwalan round-robin di seluruh koneksi meskipun itu bukan persyaratan penting. Rutin recv ack baru: Ketika ack baru tiba, pengirim meningkatkan variabel cwnd yang sesuai. Juga pada saat kedatangan ACK baru, pengirim menghapus paket dari daftar pkts [] yang telah mencapai

3. **Integrated Congestion Control/Loss Recovery**

Motivasi untuk kontrol kemacetan terintegrasi dan pemulihan kerugian adalah untuk memungkinkan aplikasi menggunakan koneksi TCP terpisah untuk setiap transfer (seperti yang mereka lakukan hari ini), tetapi untuk menghindari masalah yang disebutkan dalam Bagian 3.2 dengan membuat modifikasi yang sesuai untuk tumpukan jaringan. Kami membagi fungsionalitas TCP ke dalam dua kategori: yang berkaitan dengan abstraksi TCP byte-stream yang andal, dan yang berkaitan dengan kontrol kemacetan dan pemulihan kerugian berbasis data. Yang terakhir ini dilakukan secara terintegrasi di set koneksi paralel. Kami menyebutnya versi modifikasi TCP TCP-Int. Dengan membuka n koneksi TCP terpisah untuk transfer, aplikasi memiliki n-stream stream yang andal dan independen untuk digunakan. Kontrol aliran untuk setiap koneksi terjadi secara independen dari yang lain, sehingga pengiriman data ke aplikasi penerima juga terjadi secara independen untuk setiap koneksi. Pada saat yang sama, kontrol kemacetan terintegrasi di seluruh koneksi TCP. Ada satu jendela kemacetan untuk set koneksi TCP antara klien dan server yang menentukan jumlah total data luar biasa yang dimiliki oleh set koneksi dalam jaringan. Ketika terjadi kerugian pada salah satu koneksi,

4. **Simulation Results: One Client Host Case**

Cara ini menjelaskan hasil dari simulasi ns yang dirancang untuk menguji kontrol kemacetan dengan pengenal integer dan pemulihan kehilangan pada koneksi TCP simultan. Tes pertama menggunakan topologi pada Gambar 7. Ukuran buffer router diatur ke 3 paket. Ini cukup kecil untuk memaksa transfer untuk memiliki jendela kemacetan kecil dan sering mengalami kerugian. Sekali lagi, topologi dan parameter dipilih untuk menciptakan kembali situasi yang sering terjadi di jejak kami, dan

tidak meniru jaringan yang sebenarnya. Dalam tes ini, node pengirim melakukan 4 transfer TCP ke penerima. Transfer mulai dari 0, 2, 4 dan 6 detik dan semua berakhir pada 10 detik. Pilihan aktual dari nilai 0, 2, 4, dan 6 tidak penting, hanya saja waktu mulai setiap koneksi terhuyung-huyung dalam waktu. Gambar 10 menunjukkan plot urutan untuk pengujian menggunakan pengirim berbasis SACK. Ini menunjukkan bahwa biasanya hanya satu koneksi berperforma memuaskan pada satu waktu. Misalnya, pada waktu 2 detik, koneksi yang memulai mengalami beberapa kerugian awal dan dipaksa untuk memulihkannya melalui timeout kasar. Faktanya, koneksi ini tidak mengirim sejumlah besar data hingga 4 detik kemudian (pada waktu 6 detik). Selama periode 10 detik, koneksi dimulai pada waktu 2 detik. dan waktu 6 detik. akun untuk fraksi sangat kecil (<10%) dari total byte yang ditransfer. Ketidakadilan dan kinerja yang tidak dapat diprediksi (karena timeout kasar) tidak diinginkan dari sudut pandang aplikasi karena koneksi yang membawa data penting dapat diperlambat sementara yang lain yang membawa data yang kurang penting lebih baik.

5. **Simulation Results: Multiple Client Hosts Case**

Simulasi ini menggunakan topologi jaringan dimana saat waktu mulai koneksi di waktu 0, transfer TCP tunggal dimulai setengah dari bandwith dapat diterima kembali. berfungsi untuk mengimplementasikan fungsi yang lebih komplex dalam limit bandwidth, dimana penggunaan packet mark nya memiliki fungsi yang lebih baik. Digunakan untuk membatasi satu arah koneksi saja baik itu download maupun upload. Secara umum Queue Tree ini tidak terlihat berbeda dari Simple Queue. Perbedaan yang bisa kita lihat langsung yaitu hanya dari sisi cara pakai atau penggunaannya saja. Dimana Queue Simple secara khusus memang dirancang untuk kemudahan konfigurasi sementara Queue Tree dirancang untuk melaksanakan tugas antrian yang lebih kompleks dan butuh pemahaman yang baik tentang aliran trafik

Nama:Muhammad Iqbal Rizky Tanjung
Kelas:MTI Reg B1 Nim :192420045

Why Internet Slows Down When it's Busy

Analisa:

Dari Video tersebut pembicara menjelaskan mengenai akses internet yang diberlakukan oleh ISP ke customer, dimana secara teknis akses yang diberikan tidak murni sebesar bandwith yang ditawarkan untuk kondisi jika semua customer menggunakan full bandwith secara bersamaan pada ISP tersebut. Hal ini disebabkan karena ISP menggunakan teknik Multiplexing dalam mendistribusi bandwith ke customer, dimana ISP sendiri meyakini bahwa tidak semua customer menggunakan full bandwith diwaktu bersamaan, sehingga bandwith yang tidak digunakan/sedikit digunakan oleh beberapa customer akan dishare ke customer lain yang mengunakan lebih tinggi diwaktu bersamaan.

Saya menganalisa bahwa teknik ini digunakan ISP dengan memanfaatkan link internet yang dimilikinya untuk memperoleh jumlah customer yang lebih banyak, jika dibandingkan ISP mendisribusikan link nya secara utuh/real ke customer. Sehingga ini akan berdampak pada keuntungan bisnis ISP itu sendiri.

Meskipun ISP sendiri dalam melakukan teknik multilexing berdasarkan penelitian terhadap pola penggunaan/pattern internet oleh customer (statistic multiplexing), tentunya peluang terjadinya internet lambat pasti terjadi, ini tentunya harus menjadi perhatian bagi ISP.

Issue berkembang dari Analisa:

Service dari internet ISP yang memberlakukan multiplexing pada bandwith customer bedasarkan statistical multiplexing jika diperhatikan lebih dalam lagi akan sangat berdampak sekali terutama jika kita sebagai customer bisnis yang menggunakan internet sebagai sumber layanan/server (sebagai aplikasi web https, api-rest, ftp server, dan lainnya) untuk client-client kita, dimana jika link internet ISP tersebut dalam posisi load tinggi, maka sudah dipastikan client-client kita akan mengalami delay, bahkan drop koneksi dan hal ini tentunya dapat merugikan, karena berdampak pada availabilty layanan, serta tidak menutup kemungkinan akan timbul banyak komplain yang berdampak pada bisnis itu sendiri.

Solusi terhadap Issue:

Dari Issue yang berkembang diatas, saya mengusulkan untuk tidak menggunakan hanya satu link internet dari satu ISP saja, melainkan menggunakan beberapa link dari ISP yang berbeda untuk layanan aplikasi kita terhadap client kita. Dimana saat ini harga link internet publik semakin murah dan ini bertujuan untuk memastikan availability dari layanan dari aplikasi kita terhadap issue sebagai disampaikan diatas tidak terjadi.

Secara best practice multi link ISP ini sendiri akan dimanajemen mengunakan teknologi pembagi jalur jaringan berupa perangkat Link Controller, dimana teknologi pada perangkat ini akan memanajemen Link-link ISP yang kita gunakan tersebut dan melakukan re-route koneksi untuk client berdasarkan tipe dan kualitas link pada saat client akses ke server, dengan demikian jika saat salah satu link ISP mengalami penurunan kuliatas, atau terputus, atau pun load jaringannya tinggi, maka client kita akan diarakan ke link ISP yang lain yang kualiatasnya lebih baik, sehingga client tidak akan merasakan adanya hambatan saat mengakses layanan aplikasi kita.

Nama : Muhammad Fajar
NIM : 192400237

Permasalahan

Permasalahan yang terjadi di dalam video “why internet slow down when it’s busy” adalah bottleneck antara kapasitas dari user dan kapasitas dari ISP. Network bottleneck atau hambatan jaringan mengacu pada kondisi di mana aliran data dibatasi oleh komputer atau sumber daya jaringan. Aliran data dikontrol sesuai dengan bandwidth berbagai sumber daya sistem. Jika sistem yang bekerja pada jaringan memberikan volume data yang lebih tinggi dari apa yang didukung oleh kapasitas jaringan yang ada, maka kemacetan jaringan akan terjadi. Kemacetan atau traffic pada internet inilah yang menjadi permasalahan. Traffic ini terjadi pada saat user mengakses keluar dengan bandwith 100Mb/s setiap usernya dan bandwidth yang di sediakan oleh ISP sebesar 1 Gb/s. Jumlah user yang ada dan jumlah bandwidth yang di sediakan ISP tidak sebanding. User yang ada berjumlah 50, setiap user memiliki bandwidth 100Mb/s, dan jika seluruh user menggunakan seluruh bandwidth nya makan membutuhkan kapasitas jaringan 5Gb/s. kapasitas jaringan tersebut melebihi dari bandwidth yang disediakan oleh ISP. ISP juga tidak selalu benar menyatakan kapasitas yang diberikan, mereka bertaruh semua pelanggan tidak menggunakan semua bandwidth mereka sepanjang waktu. Angka maksimal yang diberika pada ISP belum tentu benar. Secara teori kecepatan maksimal yang diberikan berdampak jika setiap pengguna lain tidak menggunakan bandwidth sama sekali. ISP juga tidak menjamin 100% layanan yang mereka berikan lancar dan tidak ada gangguan. Jadi ini bisa menjadi faktor internet melambat. Selain itu juga perangkat keras atau device yang digunakan harus sudah mampu dengan kapasitas, perangkat yang tidak mencukupi atau tidak support akan menimbulkan juga bottleneck. Penumpukan data yang akan dikirimkan biasanya jika terlalu lama data menunggu untuk dikirim maka akan langsung di buang paket data tersebut sehingga terjadinya packetloss.

Solusi

Karena user input lebih banyak dari kapasitas yang diberikan, otomatis sistem output akan menekan dan memaksa sistem input untuk bekerja mengimbanginya. Untk mengimbanginya makan di klasifikasikan pengiriman data tersebut menjadi data prioritas, seperti data voice akan di jadikan prioritas pengiriman di bandingkan data teks, jadi pembagian data tersebut dari klasifikasi prioritas. Dengan adanya klasifikasi data prioritas, data tersebut akan lebih dahlu sampai ke penerima. Jadi ini tidak akan membuang data prioritas. Paketloss hanya akan terjadi pada data yang prioritasnya rendah. Selanjutnya dengan cara membagi bandwidth dengan mengklasifikasikan user yang di prioritaskan, biasanya bandwidth yang tidak dibagi atau hanya satu saluran dan bandwidth yang diberikan pada setiap user sama besar, maka akan menimbulkan sebagian user mendapat penundaan paket Time-Wait. Untuk itu user yang penting atau user yang prioritas menggunakan jalur khusus yang bandwithnya telah diatur sehingga tidak menumbulkan terlalu banyak delay yang diterima oleh user yang di prioritaskan. Akses yang banyak dalam satu network membuat menumpuknya IP yang membuat jalur bandwidth yang kita miliki penuh sehingga proses pengiriman data menjadi sangat terganggu dan terasa lambat. Hal ini juga bisa disebut dengan istilah Flooding. Flooding ini terjadi karena jalur bandwidth terbanjiri oleh IP address. Untuk mengatasi flooding ini yaitu mendisable proxy internal dan menambahkan filter rules untuk drop flooding attack. Penambahan rules bisa di setting melalui device, perangkat internet atau penambahan hardwarare tertent untuk membatasi IP address yang masuk seperti firewall.

Lalu diluhat dari hardware yang digunakan. Hardware yang digunakan juga harus compatible dengan kapsitas yang disediakan oleh ISP. Jika ISP memberikan bandwith sebesar 1Gb/s maka perangkat yang digunakan harus sudah support Gigabit Ethernet. Dengan perangkat yang sesuai dengan kapasitasnya maka kapasitas yang diberikan oleh ISP dapat terpakai seluruh bandwidth yang disediakan.

Nama : Muhammad Fajar
NIM : 192400237

Selanjutnya mengimplementasikan ukuran buffet socket harus dibesarkan dari ukuran defaultnya. Dengan membesarkan socket buffer maka akan menghindari receiver window dari bottleneck. Buffer soket yang besar memungkinkan anggota user untuk mendistribusikan data dan events menjadi lebih cepat tetapi juga mengambil banyak memori.

Paparan Study Case Network Topology

Palembang, 20 Desember 2019

By. Hendra Yada Putra & Ichan Syaputra (MTI-Reg B Angk 21)

Agenda

- Pendahuluan
- Topologi exisiting
- Assessment
- Rekomendasi
- Budget

Sekenario

1. Perusahaan Jasa Pengantaran Barang sejak 2012
2. Memiliki 1 Kantor pusat dan 12 Kantor cabang
3. Jumlah pegawai 300 orang
4. Bisnis menggunakan Channel Kantor dan internet
5. Mengadopsi teknologi Digital kedalam bisnis perusahaan pada tahun 2016, transaksi perusahaan meningkat pesat hampir 500%.
6. Peningkatan transaksi didominasi pada Channel Internet, dimana transaksi melalui kantor menurun tidak lebih dari 20% terhadap total transaksi.
7. Peningkatan transaksi diiringi dengan peningkatan keluhan dari customer dan internal staff.
    1. apikasi sering dikeluhkan lambat,
    2. SLA Layanan setiap bulannya tidak tercapai
    3. Virus dan Malware
8. Telah melakukan Tunning terhadap Aplikasi, Server, dan internet, namun keluhan masih tinggi.
9. Dari sisi Compliant. perusahaan juga diharuskan untuk mengikuti standar ISO27001.
10. Manajemen meminta untuk diadakan assessment dari sisi Jaringan

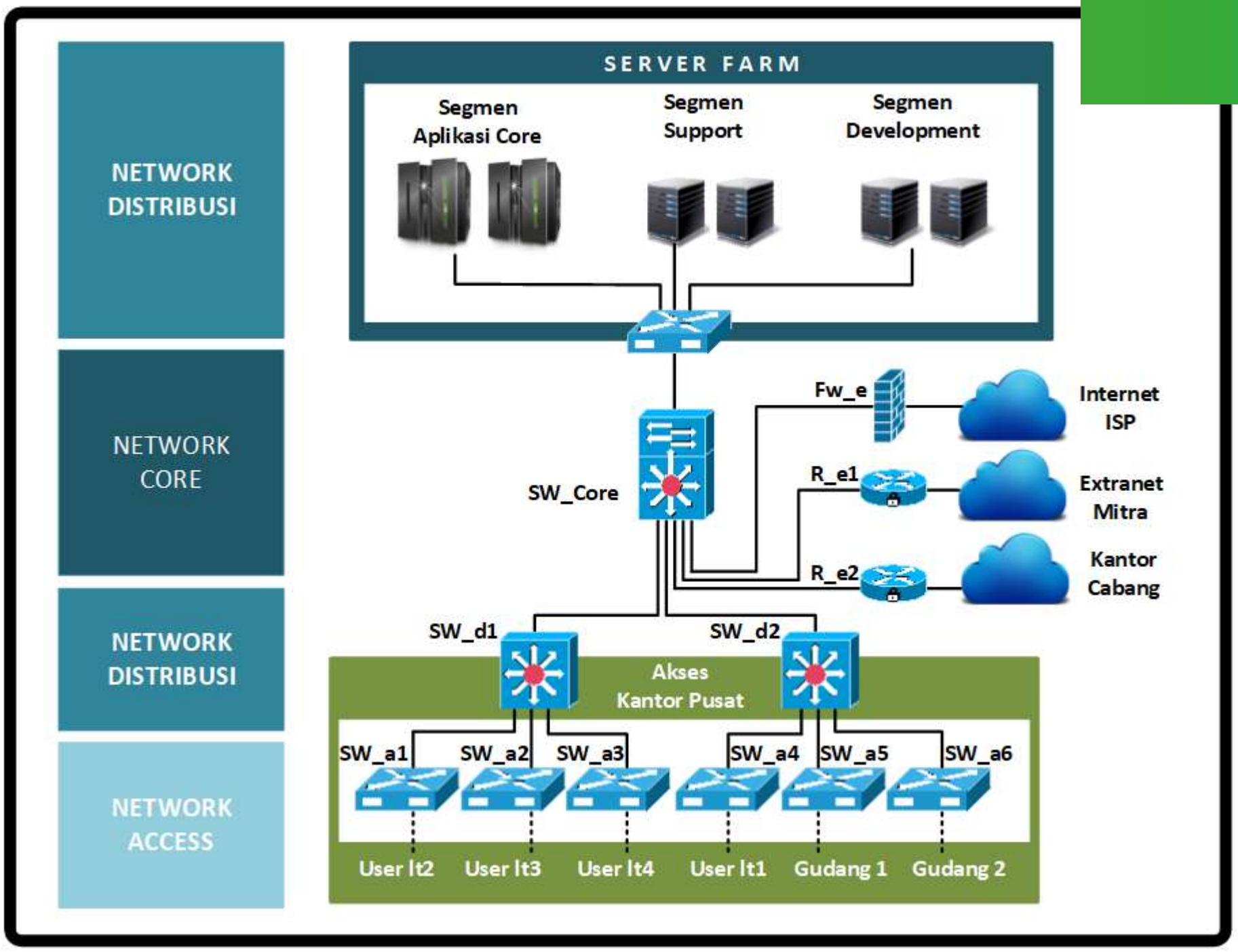
Latar Belakang

"Pemenuhan Tugas Mata Kuliah Computer Network and Communication"

Tujuan

"Memaparkan Rekomendasi Topologi Network beserta Perangkat pendukungnya untuk Perusahaan kelas Menengah Keatas, dengan Konsentrasi Peningkatan Availbaility, Performance dan keamamanan"

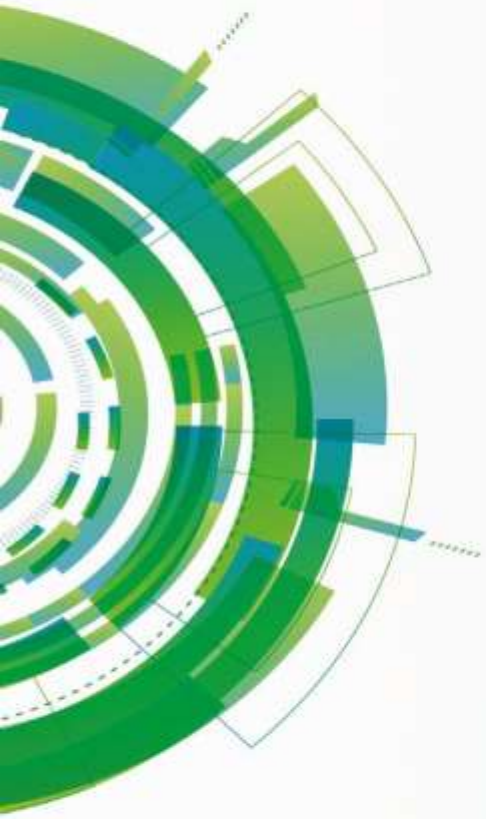
TOPOLOGI



Perangkat terdiri dari:

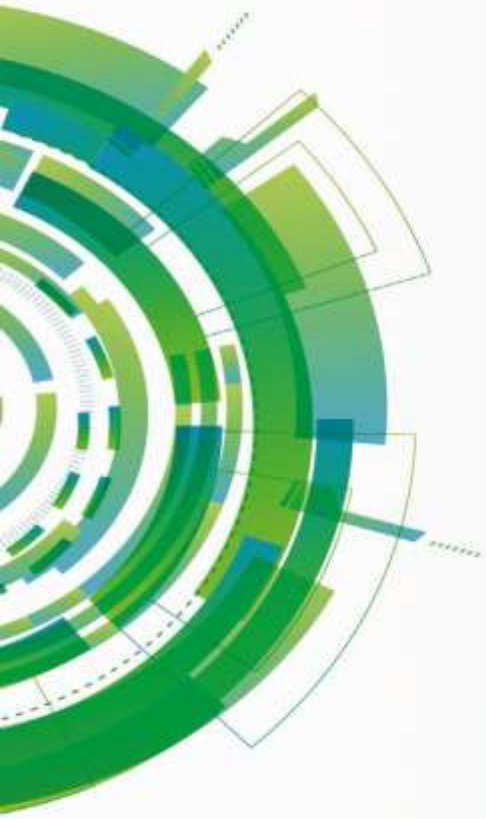
1. 1 Swich Core
2. 7 Switch Access
3. 2 Switch Distribusi
4. 2 Router
5. 1 Firewall Edge

Pengumpulan data



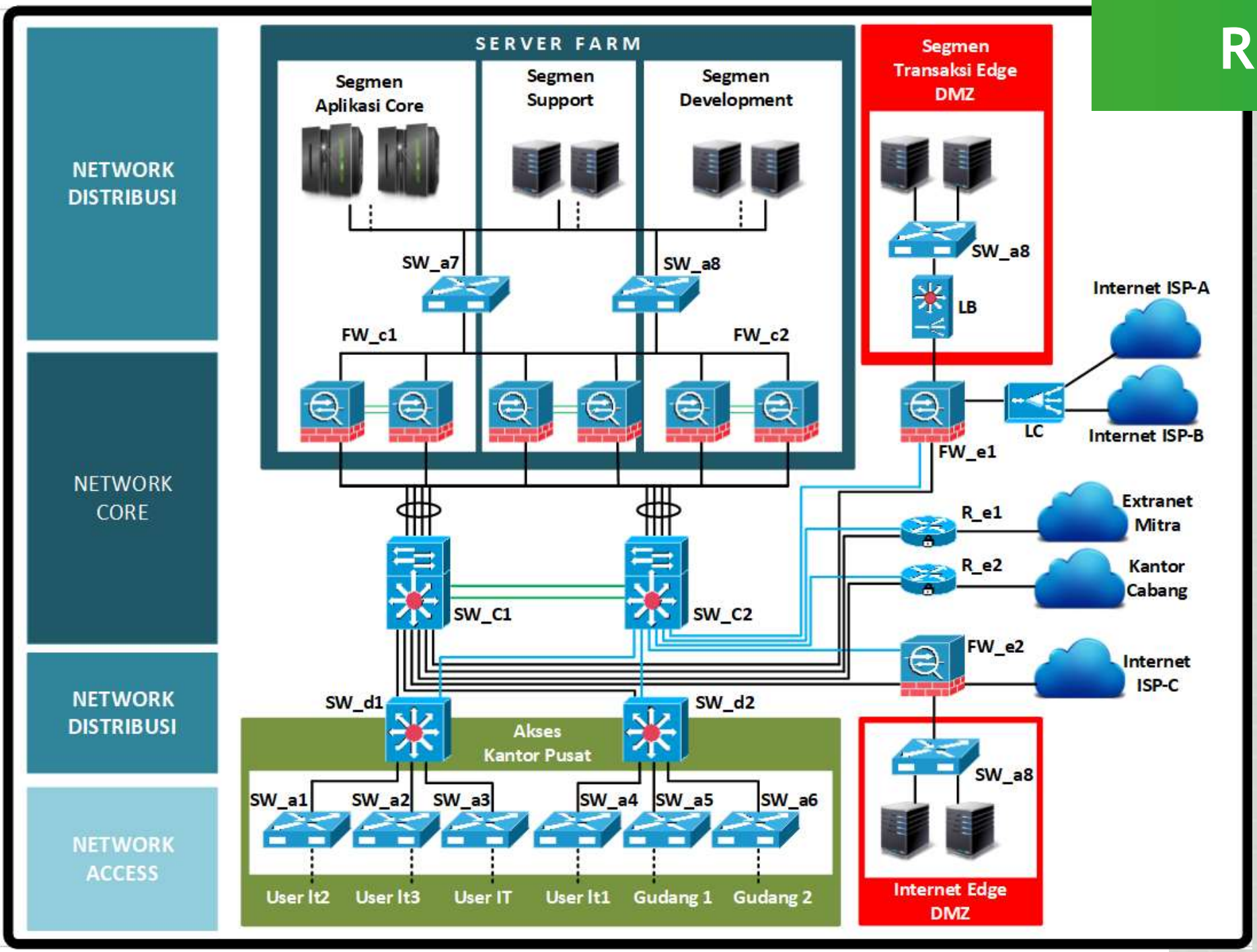
- Load trafik internet selalu mencapai level 80-90% di jam operasional kantor.
- Load trafik pada Switch Core yang menuju Server farm tidak lebih dari 70% dengan teknologi ethernet yang digunakan 1GbE
- Peak Cpu usage pada Switch Core 70%
- Pemisahan fungsi pada server Core untuk aplikasi, Database dan Web dengan SSL Sertifikat ditanam pada Web Server
- Serverfarm berada dalam 1 segemen untuk semua server: Server Core, Server Mobile API, Server Gateway, Development, AD, DNS-NTP, mail, Servicedesk, Antivirus, Monitoring, HR.
- Terdapat Cold Backup untuk server Core.
- Firewall menggunakan teknologi sebelum Next Generation
- Terdapat Perangkat yang telah EOL yaitu Swicth Core, Firewall, Switch Distribusi, Router, dan beberapa Switch Access

Hasil analisa



- Harus dibedakan akses Internet untuk akses transaksi dan User
- Segera dilakukan penggantian perangkat EOL (risiko tinggi)
- Dibutuhkan Penambahan Backup yang tergolong Critical poin
- Perlu Menambahkan Link Internet baru dengan mengimplementasikan perangkat Link Controller untuk Redudansi Internet dan penempatan SSL Sertifikat
- Pelu Menempatkan Server yang membutuhkan koneksi internet ke area DMZ sesuai area jaringan internet
  - Server support (Antivirus, DNS-NTP, Mail, Server Gateway)
  - Server Web Core dan Server Mobile API
- Pemisahan Segement pada Serverfarm dan menambahkan NGT Firewall untuk proteksi masing-masing segmen
- Dibutuhkan pembagi Beban untuk WEB Core Backup
- Perlu penggantian firewall Internet dengan NGT Firewall.

REKOMENDASI



Perangkat terdiri dari:

1. 2 Swich Core
2. 8 Switch Access
3. 4 Switch Distribusi
4. 2 Router
5. 1 Load Balancer
6. 1 Load Controller
7. 2 Firewall Core
8. 2 Firewall Edge

Segmentasi

DETIL

- Segmen Aplikasi Core
  - Server Aplikasi Core Bisnis
  - Server Database
  - Server Interface
- Segmen Support
  - Server Active Directory
  - Server DNS internal
  - Server Aplikasi HR
  - Server Aplikasi Helpdesk
  - Server Email
  - Server Monitoring & Manajement
- Segmen Development
  - Server Development Core
  - Server Development Interface
  - Server Development Support

- Segemen Transaksi Edge
  - Server Web Korporasi
  - Server API Mobile
  - Server Web Transaksi
- Segemen Internet Edge
  - Server DNS Forwader
  - Server Antivirus
  - Server NTP
  - Akses Internet User

KEUNGGULAN

- Segmentasi lebih dalam untuk area server Farm sehingga keamanan lebih terjamin
- Kualitas Layanan dapat ditinggkatkan
- Pemisahan Jalur internet user dengan internet untuk transaksi
- Pengamanan berlapis dengan pemenuhan standard iso27001
- Availability layanan dengan link controller dan link balancer
- Dengan memisahkan akses SSL sertifikat pada server dapat meningkatkan performace Server

No	Item	Jumlah	Harga Satuan	Harga
1	Switch Core Cisco Catalyst 9300 C9300-48P-E	2	250.000.000	500.000.000
2	Switch Distribusi Cisco Catalyst 3850 Ws-C3850-24t-S	4	80.000.000	320.000.000
3	Switch Access Cisco Catalyst 2960-x (WS-C2960X-48FPD-L)	4	50.000.000	200.000.000
4	Router Cisco Cisco ASR 1002-X	2	70.000.000	140.000.000
5	Firewall Core Fortigate 1000D	1	260.000.000	260.000.000
6	Firewall Edge Fortigate 100e	1	80.000.000	80.000.000
7	Load Balancer A10 Thunder 840 ADC	1	150.000.000	150.000.000
8	Link Controller f5 2200 LTM	1	250.000.000	250.000.000
9	Cabling,Soket, dll (Panduit)	1	20.000.000	20.000.000
10	Implementasi	20	5.000.000	100.000.000
	TOTAL			2.020.000.000

Q & A


Thank you

Nama:Muhammad Ichsan
Kelas:MTI Reg B angkatan 21

Upload UAS MITB
Video title : Why Internet Slows Down When it's Busy

Analisa:

Dari Video tersebut pembicara menjelaskan mengenai akses internet yang diberlakukan oleh ISP ke customer, dimana secara teknis akses yang diberikan tidak murni sebesar bandwith yang ditawarkan untuk kondisi jika semua customer menggunakan full bandwith secara bersamaan pada ISP tersebut. Hal ini disebabkan karena ISP menggunakan teknik Multiplexing dalam mendistribusi bandwith ke customer, dimana ISP sendiri meyakini bahwa tidak semua customer menggunakan full bandwith diwaktu bersamaan, sehingga bandwith yang tidak digunakan/sedikit digunakan oleh beberapa customer akan dishare ke customer lain yang mengunakan lebih tinggi diwaktu bersamaan.

Saya menganalisa bahwa teknik ini digunakan ISP dengan memanfaatkan link internet yang dimilikinya untuk memperoleh jumlah customer yang lebih banyak, jika dibandingkan ISP mendisribusikan link nya secara utuh/real ke customer. Sehingga ini akan berdampak pada keuntungan bisnis ISP itu sendiri.

Meskipun ISP sendiri dalam melakukan teknik multilexing berdasarkan penelitian terhadap pola penggunaan/pattern internet oleh customer (statistic multiplexing), tentunya peluang terjadinya internet lambat pasti terjadi, ini tentunya harus menjadi perhatian bagi ISP.

Issue berkembang dari Analisa:

Service dari internet ISP yang memberlakukan multiplexing pada bandwith customer bedasarkan statistical multiplexing jika diperhatikan lebih dalam lagi akan sangat berdampak sekali terutama jika kita sebagai customer bisnis yang menggunakan internet sebagai sumber layanan/server (sebagai aplikasi web https, api-rest, ftp server, dan lainnya) untuk client-client kita, dimana jika link internet ISP tersebut dalam posisi load tinggi, maka sudah dipastikan client-client kita akan mengalami delay, bahkan drop koneksi dan hal ini tentunya dapat merugikan, karena berdampak pada availabilty layanan, serta tidak menutup kemungkinan akan timbul banyak komplain yang berdampak pada bisnis itu sendiri.

Solusi terhadap Issue:

Dari Issue yang berkembang diatas, saya mengusulkan untuk tidak menggunakan hanya satu link internet dari satu ISP saja, melainkan menggunakan beberapa link dari ISP yang berbeda untuk layanan aplikasi kita terhadap client kita. Dimana saat ini harga link internet publik semakin murah dan ini bertujuan untuk memastikan availability dari layanan dari aplikasi kita terhadap issue sebagai disampaikan diatas tidak terjadi.

Secara best practice multi link ISP ini sendiri akan dimanajemen mengunakan teknologi pembagi jalur jaringan berupa perangkat Link Controller, dimana teknologi pada perangkat ini

Nama:Muhammad Ichsan
Kelas:MTI Reg B angkatan 21

Upload UAS MITB
Video title : Why Internet Slows Down When it's Busy

akan memanajemen Link-link ISP yang kita gunakan tersebut dan melakukan re-route koneksi untuk client berdasarkan tipe dan kualitas link pada saat client akses ke server, dengan demikian jika saat salah satu link ISP mengalami penurunan kuliatas, atau terputus, atau pun load jaringannya tinggi, maka client kita akan diarakan ke link ISP yang lain yang kualiatasnya lebih baik, sehingga client tidak akan merasakan adanya hambatan saat mengakses layanan aplikasi kita.

Nama : Muhammad Wahyudi
NIM : 192420023

Upload UAS MITB
Video title : Why Internet Slows Down When it's Busy

Analisa:
Dari Video tersebut pembicara menjelaskan mengenai akses internet yang diberlakukan oleh ISP ke customer, dimana secara teknis akses yang diberikan tidak murni sebesar bandwith yang ditawarkan untuk kondisi jika semua customer menggunakan full bandwith secara bersamaan pada ISP tersebut. Hal ini disebabkan karena ISP menggunakan teknik Multiplexing dalam mendistribusi bandwith ke customer, dimana ISP sendiri meyakini bahwa tidak semua customer menggunakan full bandwith diwaktu bersamaan, sehingga bandwith yang tidak digunakan/sedikit digunakan oleh beberapa customer akan dishare ke customer lain yang mengunakan lebih tinggi diwaktu bersamaan.

Saya menganalisa bahwa teknik ini digunakan ISP dengan memanfaatkan link internet yang dimilikinya untuk memperoleh jumlah customer yang lebih banyak, jika dibandingkan ISP mendisribusikan link nya secara utuh/real ke customer. Sehingga ini akan berdampak pada keuntungan bisnis ISP itu sendiri.

Meskipun ISP sendiri dalam melakukan teknik multilexing berdasarkan penelitian terhadap pola penggunaan/pattern internet oleh customer (statistic multiplexing), tentunya peluang terjadinya internet lambat pasti terjadi, ini tentunya harus menjadi perhatian bagi ISP.

Issue berkembang dari Analisa:
Service dari internet ISP yang memberlakukan multiplexing pada bandwith customer bedasarkan statistical multiplexing jika diperhatikan lebih dalam lagi akan sangat berdampak sekali terutama jika kita sebagai customer bisnis yang menggunakan internet sebagai sumber layanan/server (sebagai aplikasi web https, api-rest, ftp server, dan lainnya) untuk client-client kita, dimana jika link internet ISP tersebut dalam posisi load tinggi, maka sudah dipastikan client-client kita akan mengalami delay, bahkan drop koneksi dan hal ini tentunya dapat merugikan, karena berdampak pada availabilty layanan, serta tidak menutup kemungkinan akan timbul banyak komplain yang berdampak pada bisnis itu sendiri.

Solusi terhadap Issue:
Dari Issue yang berkembang diatas, saya mengusulkan untuk tidak menggunakan hanya satu link internet dari satu ISP saja, melainkan menggunakan beberapa link dari ISP yang berbeda untuk layanan aplikasi kita terhadap client kita. Dimana saat ini harga link internet publik semakin murah dan ini bertujuan untuk memastikan availability dari layanan dari aplikasi kita terhadap issue sebagai disampaikan diatas tidak terjadi.

Secara best practice multi link ISP ini sendiri akan dimanajemen mengunakan teknologi pembagi jalur jaringan berupa perangkat Link Controller, dimana teknologi pada perangkat ini akan

Nama : Muhammad Wahyudi
NIM : 192420023

Upload UAS MITB
Video title : Why Internet Slows Down When it's Busy

memanajemen Link-link ISP yang kita gunakan tersebut dan melakukan re-route koneksi untuk client berdasarkan tipe dan kualitas link pada saat client akses ke server, dengan demikian jika saat salah satu link ISP mengalami penurunan kuliatas, atau terputus, atau pun load jaringannya tinggi, maka client kita akan diarakan ke link ISP yang lain yang kualiatasnya lebih baik, sehingga client tidak akan merasakan adanya hambatan saat mengakses layanan aplikasi kita.

The TIME-WAIT state in TCP and Its Effect on Busy Servers

Theodore Faber
Joe Touch
Wei Yue
University of Southern California/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292
Phone: 310-822-1511
{faber,touch,wyue}@isi.edu

Abstract - **Hosts providing important network services such as HTTP and FTP incur a per-connection memory load from TCP that can adversely affect their connection rate and throughput. The memory requirement is directly tied to the number of connections; caching and other sharing methods will not alleviate it. We have observed HTTP throughput reductions of as much as 50% under SunOS 4.1.3 due to this loading.**

This paper advocates off-loading the memory requirements to the growing number of clients. This reduces server memory requirements as connection rate at that server grows due to increases in the number of clients and the bandwidth available on the network. Our approaches control server memory load better with growing client load than per-transaction techniques such as persistent HTTP connections. Our approaches also interoperate with persistent connections to take advantage of their other benefits.

This paper describes the causes of the memory loading, called *TIME-WAIT loading*, and defines three methods of alleviating it that scale with increasing number of clients. We present measurements of the systems and a comparison of their properties.

1. Introduction

The Transmission Control Protocol (TCP)[1] provides reliable byte-stream transport to hosts on the Internet. TCP is used by most network services that require reliable transport, including the Hypertext Transport Protocol (HTTP)[2]. TCP's method of isolating old connections from new ones results in an accumulation of state at busy servers that can reduce their throughput and connection rates. The effect on HTTP servers is of particular interest because they carry a large amount of Internet traffic.

TCP requires that the endpoint that closes a connection blocks further connections on the same host/port pair until there are no packets from that connection remaining in the network[2]. Under HTTP, this host is usually the server[2].

To temporarily block connections, one endpoint keeps a copy of the TCP control block (TCB) indicating that the connection has been terminated recently. Such a connection is in the TIME-WAIT state[1]. Connections in TIME-WAIT are moved to CLOSED and their TCB discarded after enough time has passed that all packets from the same connection have left the network. Packets leave the network by arrive at one of the endpoints and being rejected, or arriving with an expired time-to-live (TTL) field at a router and being deleted.

For endpoints that are the target of many connections, thousands of connections may be in TIME-WAIT state at any time, which introduces significant memory overhead. We refer to this condition as *TIME-WAIT loading.*

If the endpoint's TCP implementation searches all TCBs when delivering packets, TIME-WAIT loading will directly affect its performance. The presence of many TIME-WAIT TCBs can increases the demultiplexing time for active connections. We have seen throughput drop by 50% at loaded endpoints, and the effect on commercial servers has been noted elsewhere[3]. Some TCP implementations address the demultiplexing problem without addressing the memory load; we discuss them in Section 2.2.

The design of TCP places the TIME-WAIT TCB at the endpoint that closes the connection; this decision conflicts with the semantics of many application protocols. The File Transfer Protocol (FTP)[4] and HTTP both interpret the closing of the transport connection as an end-of-transaction marker. In each case, the application protocol requires that servers close the transport connection, and the transport protocol requires that servers incur a memory cost if they do. Protocols that use other methods of marking end-of-transaction, e.g., SUN RPC over TCP[5], can have the clients close connections at the expense of a more complex application protocol.

If the number of clients continues to increase, the only way to keep server TIME-WAIT memory requirements constant is to move the TIME-WAIT TCBs to clients. Aggregating more requests per connection merely reduces the growth of the memory load with respect to increasing client load; moving the load to clients distributes server memory load to the cause of that load.

As networks become faster and support more users, the connection rates at busy servers are likely to increase, resulting in more TIME-WAIT loading. Even if packet demultiplexing is done efficiently, the memory cost of TIME-WAIT loading can become a significant drain on server resources. Servers will need additional physical memory resources to support the load. For embedded servers using TCP, this translates directly to a higher cost in dollars and power.

0-7803-5420-6/99/$10.00 (c) 1999 IEEE

Distributing the TCBs across clients scales better than per-transaction load reductions like persistent HTTP connections for controlling TIME−WAIT loading. The transaction rate is being driven up by the increasing bandwidth and the growing number of users. Reducing each transaction's cost only slows the growth rate. Offloading TCBs to clients distributes the load more equitably as the number of clients grows, riding the growth curve instead of throttling it. Because Persistent connections reduce other per-connection costs, such as extra connection establishment overhead, our systems interoperate with them.

This work presents three systems to distribute the TIME−WAIT build-up to clients. We suggest avoiding TIME−WAIT loading by negotiating which endpoint will hold the TIME−WAIT TCB during connection establishment. This provides the most control over TIME−WAIT location by making the placement an explicit part of connection establishment; however, performing this negotiation and respecting it when the connection is closed requires significant changes to TCP.

In light of this, we also discuss two less invasive alternative solutions: a modification to TCP that shifts the TIME−WAIT TCB from server to client when the connection is closed; and a modification to HTTP that supports the client closing the connection and holding the TIME−WAIT TCB.

2. The TIME−WAIT State

This Section discusses the TIME−WAIT state and its use in TCP in some detail, and how the TIME−WAIT state impacts the performance of busy servers.

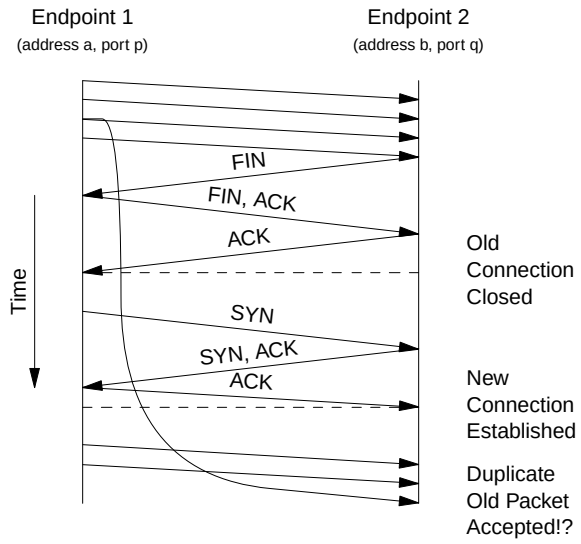


Figure 1: The Problem Addressed by the TIME−WAIT State

2.1. The Function of TIME−WAIT

The purpose of TIME−WAIT is to prevent delayed packets from one connection being accepted by a later connection. Concurrent connections are isolated by other mechanisms, primarily by addresses, ports, and sequence numbers[1].

The TIME−WAIT state avoids the situation depicted in Figure 1. Arrows represent packets, and endpoints' time lines run down the page. Packets are labelled with the header flags that are relevant to connection establishment and shutdown; unlabelled packets carry only data.

Specifically:

- A connection from (address a, port p) to (address b, port q) is terminated
- A second connection from (address a, port p) to (address b, port q) is established
- A duplicate packet from the first connection is delayed in the network and arrives at the second connection when its sequence number is in the second connection's window.

If such a packet appears, there is no way for the endpoints in the second connection to determine that the delayed packet contains data from the first connection.

This confusion can only exist if a second connection from (address a, port p) to (address b, port q) is active while duplicate packets from the first connection are still in the network. TCP avoids this condition by blocking any second connection between these address/port pairs until one can assume that all duplicates must have disappeared.

Connection blocking is implemented by holding a TIME−WAIT TCB at one endpoint and checking incoming connection requests to ensure that no new connection is established between the blocked addresses and ports. Because only a connection between the same endpoints can cause the confusion, only one endpoint needs to hold the state. The TCB is held for twice the maximum segment lifetime (MSL).

The MSL is defined as the longest period of time that a packet can remain undelivered in the network. Originally, the TTL field of an IP packet was the amount of time the packet could remain undelivered, but in practice the field has become a hop count[6]. Therefore, the MSL is an estimate rather than a guarantee. The Internet host requirements document suggests a using 2 minutes as the MSL[7], but some implementations use values as small as 30 seconds[8]. Under most conditions waiting $2 \times$ MSL is sufficient to drain duplicates, but they can and do arrive after that time. The chance of a duplicate arriving after $2 \times$ MSL is greater if MSL is smaller.

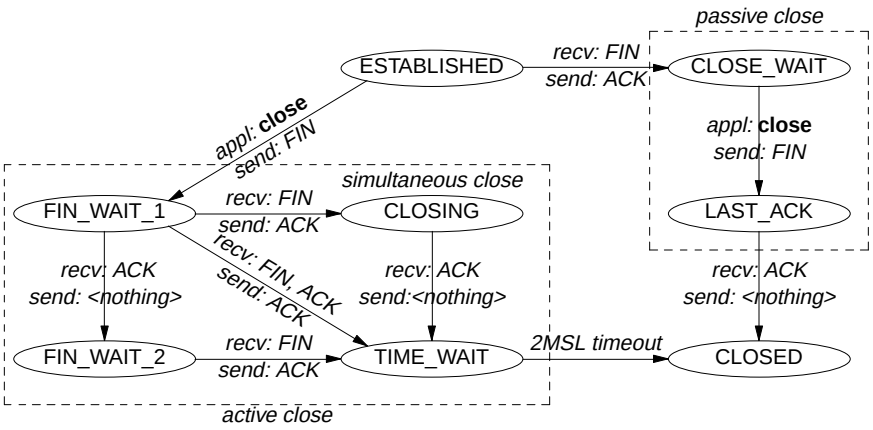


Figure 2: The TCP State Machine For Closing Connections[1]

TCP requires that the endpoint that initiates an active close of the connection eventually enters `TIME-WAIT`. Closing a connection that is in the `ESTABLISHED` state is called *actively closing*, closing from `CLOSE-WAIT` is *passively closing*. If both ends close the connection from `ESTABLISHED`, this is a simultaneous close, and both endpoints do a modified active close[1]. (See Figure 2.) Intuitively, the first endpoint to close a connection closes it actively, and the second passively; HTTP and FTP servers generally close connections actively.

Tying the `TIME-WAIT` state to the closing method simplifies the TCP state diagram because no information from the connection establishment affects connection termination.

2.2. Performance Problems at Busy Servers

Because client/server protocols are generally synchronized request/response protocols, the protocol specification usually determines which endpoint will close the transport connection. For example, FTP clients know a file has been delivered successfully if the connection on which the file was transferred closes gracefully[4]; this implies that the server closes connections.

TCP commentators encourage client/server systems to arrange for the client to close connections to avoid `TIME-WAIT` loading[8]. Many protocols, such as FTP, do not follow this convention. We discuss the reasons for this in Section 2.3.

Because application protocols do not take `TIME-WAIT` TCB distribution into account, heavily loaded servers can have thousands of connections in `TIME-WAIT` that consume memory and can slow active connections. In BSD-based TCP implementations, TCBs are kept in mbufs, the memory allocation unit of the networking subsystem[9]. There are a finite number of mbufs available in the system, and mbufs consumed by TCBs cannot be used for other purposes such as moving data. Some systems on high speed networks can run out of mbufs due to `TIME-WAIT` buildup under high connection load. A SPARCStation 20/71 under SunOS 4.1.3 on a 640 Mb/s Myrinet[10] cannot support more than 60 connections/sec because of this limit.

Demultiplexing incoming packets requires searching the endpoint's list of TCBs which will be full of of `TIME-WAIT` TCBs at a `TIME-WAIT` loaded server. In a simple implementation, the TCB list is searched linearly to pass the packet to the appropriate connection, which can be a bottleneck. The additional search overhead can cut throughput in half between two SunOS 4.1.3 SPARCStations on a Myrinet. We show an example in Section 5.

Modern TCP implementations avoid the overhead of the linear search of `TIME-WAIT` TCBs when demultiplexing packets. BSDI/OS keeps `TIME-WAIT` TCBs at the end of the list of TCBs, so that they can be used as a terminator for the linear search[11]. Looking TCBs up in a hash table reduces lookup times both for systems with many `TIME-WAIT` TCBs and for many active connections[12].

Some systems address `TIME-WAIT` loading by using a shorter MSL, hoping to age the `TIME-WAIT` TCBs out of the system sooner, which weakens the protection afforded by `TIME-WAIT`. If `TIME-WAIT` TCBs are kept at clients, they can afford to keep them for the full MSL. Using a shorter MSL at servers alleviates the memory usage problem, but can affect the integrity of communications with any host to which it connects. The size of the MSL to maintain a given memory usage level is inversely proportional to the connection rate. This implies that connections will be least protected at the most loaded servers.

Even in TCP implementations that demultiplex packets efficiently, such as those mentioned above, `TIME-WAIT` TCB accumulation consumes memory. Systems such as persistent HTTP connections can reduce the per-transaction server memory cost, but the Internet continues to grow both in terms of available bandwidth and in terms of number of users. The result is that busy servers will be seeing more

[1] The diagram layout is modelled after one appearing in [8].

connections per unit time, which translates directly into increased memory usage.

Spreading the memory requirement to the growing number of clients scales better than reducing the per-transaction cost at the server when the number of transactions is still growing. Per-transaction systems try to reduce the cost of each client, but the requirements still grow with increasing Internet traffic. Distributing the TIME−WAIT TCBs to clients takes advantage of their growing number to reduce the memory load on servers. Our systems support persistent connections in order to share their other benefits, such as avoiding extra TCP 3-way handshakes.

2.3. Server Close and Application Semantics

Using connection close to delimit transactions is a clean abstraction with unintended performance impact under TCP. HTTP and FTP servers signal the end of a transaction by closing the transport connection. As we argued above, this can TIME−WAIT load servers[2,4]. These protocols carry the majority of the TCP traffic in the wide area Internet today.

Using the TCP connection closure as part of a simple request/response protocol results in simpler protocols. In such as system, the semantics of TCP's close make it an unambiguous end-of-transaction marker.[2] Without this mechanism, protocols are forced to specify transaction length explicitly, or to provide an unambiguous end-of-transaction marker.

Providing an unambiguous end-of-transaction marker in the data stream requires that the server either knows the content length when the response begins, or edits the outgoing data stream to mask end-of-transaction markers in the data that the client must restore. Transactions must have a length field or be byte-stuffed.

If a response is generated dynamically, its length may not be known when the response starts, making the former framing methods unwieldy. Such data may be generated on demand from another program in the system that does not respect the protocol end-of-file marker. An HTTP server that supports the Common Gateway Interface (CGI) is such a system. Buffering that program's output to determine its size or remove embedded end-of-transaction markers slows the response.

3. TIME−WAIT Negotiation

This section discusses modifying TCP to negotiate the TIME−WAIT TCB holder when the connection is established. This makes the post-connection memory requirement explicit and allows either endpoint to decline the connection if the overhead is unacceptable. Furthermore it is transparent to applications using the transport, and allows them to close the transport connection as part of their protocol without incurring hidden costs.

We propose adding a TCP option, TW-Negotiate, that indicates which end of the connection will hold the TCBs. TW-Negotiate will be negotiated during the three-way handshake that is used to synchronize TCP sequence numbers. The three-way handshake has been used to negotiate other options, such as TCP window scaling[14].

The negotiation algorithm for a client/server connection:

1. Client includes the TW-Negotiate option in the <SYN> packet for the connection. TW-Negotiate contains the IP address of the end that will hold the TCB. The option's presence indicates that the client supports negotiation. Clients must send an IP address, to support the algorithm below for resolving a simultaneous open.
2. Server returns the <SYN, ACK> packet with TW-Negotiate set to its choice to keep the TIME−WAIT state. If it does not support negotiation, it sends no TW-Negotiate option.
3. The client decides if the server's choice is acceptable. If so, it acknowledges the <SYN, ACK> packet with the same value of TW-Negotiate. If not it aborts the connection with an <RST> packet. (The connection is aborted as though it failed to synchronize, and introduces no new failure modes to TCP.) Aborting the connection from this unsynchronized condition leaves no extra state at either endpoint; the server returns to LISTEN, and the client closes the connection. If the server returned no TW-Negotiate option, the connection will use current TCP semantics: the side that issues the active close will enter TIME−WAIT (or both will if they close simultaneously).

This algorithm handles any non-simultaneous connection establishment; the following handles the simultaneous case. During a simultaneous open, neither endpoint is in the server role, so neither has the TW-Negotiate value has priority. As establishment progresses, both sides will find themselves in SYN-RCVD (the state transitions are CLOSED→SYN-SENT→SYN-RCVD) Each will know two TW-Negotiate values: theirs and the other endpoint's[1]. From here, each endpoint behaves as if it were a client in step 3 of the negotiation and had received the value in Table 1 from its peer. At most one endpoint will disagree with the conclusion, and send an <RST>.

This algorithm guarantees that the endpoints will agree on which will enter TIME−WAIT when the connection is dissolved, or will fall back to TCP semantics if either side does not support negotiation.

[2] This is false in protocols that can have multiple pending requests, e.g., pipelined HTTP requests[13].

0-7803-5420-6/99/$10.00 (c) 1999 IEEE

TW-Negotiation Values Known	TW-Negotiation Value To Use
Either Contains No Option	No Option
The Same IP Address	That IP Address
Two Different IP addresses	Larger IP Address

Table 1: Negotiation Values for Simultaneous Open

As an example of a negotiation during when two endpoints simultaneously open the same connection, consider two endpoints, A and B. A's IP address is larger. Both always attempt to negotiate holding their own TIME−WAIT TCB, i.e., they send their own address in the TW-Negotiate option. The two endpoints attempt to open the connection, the <SYN>s cross in the network and both receive a <SYN> before they have received a <SYN, ACK>. The endpoints know the value of both TW-Negotiate options, but neither is in the server role above and can make the "final offer". they both act as if they had were clients and had sent a <SYN> with their preferred value, and received a <SYN, ACK> with A's address. They use A's address based on Table 1.

If having A hold the TIME−WAIT TCB is acceptable to both, they will both send an <SYN, ACK> with A's address in the header, and the connection will be established (after the final <ACK> exchange). If B is unwilling to have A hold the TCB, it will send an <RST> and the connection will never be established. A will always send a <SYN, ACK> because Table 1 has selected its preference; this will always be true of one endpoint, so only one will send the <RST>.

This system is biased toward endpoints with larger IP addresses; however, simultaneous attempts to establish connections are rare and never occur in client/server systems. Should systems evolve that exhibit frequent simultaneous connection establishment attempts and TIME−WAIT loading, the protocol can be modified to include a random number in each header and use that to pick the TCB holder.

When the a connection connection that has a negotiated TIME−WAIT holder is closed, the two endpoints will exchange <FIN> packets as usual, and the TIME−WAIT holder will enter TIME−WAIT and the other endpoint will enter CLOSED, regardless of which end closed actively and which (if either) passively.

When negotiation is added to a endpoint's operating system, most applications will use system-wide defaults for the TW-Negotiate option. These defaults will be set to minimize server load, i.e, to hold TIME−WAIT TCBs at clients. A mechanism, such as a socket option, will be provided to allow applications to override the default setting.

The negotiation algorithm allows busy servers to accept connections only from clients that are willing to incur the TIME−WAIT overhead. Application algorithms do not have to alter their use of connection close in their protocol, and incur no hidden performance penalty associated with the distribution of TIME−WAIT TCBs.

3.1. Barriers to Adoption

Although the algorithm above is simple to describe, it represents a significant change to the TCP state machine. Many TCP implementations are descended from the original BSD reference implementation of the TCP/IP stack that directly implements the TCP state machine[9]. Implementing changes to that state machine would require significant programming and testing effort. Furthermore, proofs of TCP correctness that rely on the TCP state machine would be invalidated.

The state machine changes reflect the fact that information from connection establishment affects the closure. Currently, once an endpoint has finished the three-way handshake and entered the ESTABLISHED state, it is impossible to tell what role it played in creating the connection. By using information from the connection establishment to determine the endpoints' behavior when the connection is terminated, we have created two states that represent an established connection, and which state a connection enters depends on the result of an option negotiation.

Negotiating the TIME−WAIT TCB holder when the connection is closed disrupts the state machine less, but reduces a endpoints' control over their resources. A client in a system that negotiates the holder before the connection is established cannot get the data it wants without reaching an agreement with the server about which side bears the costs of the TIME−WAIT TCB; a client in a system that negotiates the holder when the connection closes can always leave the server paying the TIME−WAIT cost. We prefer a system that makes agreement on the allocation of connection costs a prerequisite to incurring them. However, because allocating the TIME−WAIT state at connection close time is simpler, we have implemented an example of that system, which we discuss in Section 5.1.

We feel that the benefits of providing applications more control over the endpoint resources that they commit to a connection has significant advantages. The proposed system isolates that protocol behavior, which makes the solution more general than, for example, reversing the roles of active and passive closer. Finally it isolates application programs from an implementation detail of the transport protocol, allowing new application protocols to meet the needs of applications rather than being bent out of shape by transport.

4. Other Systems to Avoid TIME−WAIT TCB Loading

In this section we propose two less ambitious solutions to the server TIME−WAIT loading problem. Each solution is a small change to an existing protocol that reduces TIME−WAIT loading in HTTP and FTP servers. One system modifies TCP to exchange TIME−WAIT TCBs after a

successful close, the other modifies HTTP to encourage clients to close the underlying TCP connection. We chose to modify HTTP because it is a large component of Internet traffic.

These solutions are intended to be practical ones. As such, they are incrementally deployable and compatible with existing protocol specifications. Both the TCP and HTTP solutions realize benefits without modifying the server systems, although additional benefits accrue if both client and server implement the HTTP changes. Neither set of changes violates the current TCP or HTTP specifications, so changed systems will operate in today's Internet.

We have implemented both systems, and observed that both significantly reduce the TIME-WAIT loading on HTTP servers. We discuss the performance of both systems in Section 5. Patches which implement the systems are available from the authors.

4.1. Transport Level (TCP) Solution

The TCP solution exchanges the TIME-WAIT state between the server and client when the connection is closed. We modify the client's TCP implementation so that after a successful passive close, it sends an <RST> packet to the server and puts itself into a TIME-WAIT state. The <RST> packet removes the TCB in TIME-WAIT state from the server; the explicit transition to a TIME-WAIT state in the client preserves correct TCP behavior.

If the client <RST> is lost, both server and client remain in TIME-WAIT state, which is equivalent to a simultaneous close. If either endpoint reboots during the <RST> exchange, the behavior is the same as if an endpoint running unmodified TCP fails with connections in TIME-WAIT state: packets will not be erroneously accepted if the endpoint recovers and refuses connections until a $2 \times$ MSL period has elapsed[1,7]. The behavior of an active close is unaffected.

Using an <RST> packet means that this system only works with TCP stacks that accept <RST> packets that arrive for a connection in TIME-WAIT state. Such stacks are susceptible to TIME-WAIT assassination[15], which can lead to connections becoming desynchronized or destroyed. TIME-WAIT assassination is the accidental or malicious deletion of a TIME-WAIT TCB at an endpoint, which can lead to confusion as shown in Figure 1.

Our system assassinates TIME-WAIT states at the server and replaces them at the client, which does not change TCP's behavior. Adding our system to a server that is susceptible to TIME-WAIT assassination does not make it more vulnerable, but a server that implements the changes in[15] to prevent assassinations will not benefit the system described in this section. Interactions between a server that prevents TIME-WAIT assassination and a client that implement our changes do not compromise TIME-WAIT guarantees.

Our system modifies the TCP state machine by changing the arc from LAST-ACK to CLOSED to an arc from LAST-ACK to TIME-WAIT and sending an <RST> when the arc is traversed. (See Figure 2 for the relevant section of the TCP state diagram.) To reduce TIME-WAIT loading from FTP or HTTP, these modifications need to be made only to clients.

Hosts that act primarily as clients may be configured with the new behavior for all connections; clients that serve as both client and server, such as HTTP proxies, may be configured to support both the new and old behaviors. Supporting both swapping and non-swapping close is straightforward, although it requires a more extensive modification of the TCP state machine.

To allow both behaviors on the same host we split the LAST-ACK state into two states, one that represents the current behavior (LAST-ACK) and one which represents the modified behavior (LAST-ACK-SWAP).[3] If the client invokes close while in CLOSE-WAIT, current TCP semantics apply; if the client invokes close_swap in the same state, the <RST>-sending behavior applies. Close and close_swap are indistinguishable if invoked from ESTABLISHED.

The state machine in Figure 3 implements both behaviors. Compare it with the earlier Figure 2, which shows the state machine for TCP's connection closing behavior. The details of active and simultaneous closing are unchanged from Figure 2, and are omitted for clarity.

Adding close_swap does not require adding a system call. One implementation of close_swap adds a per-connection flag that changes the default behavior when set. When a connection with the flag set is closed, the close system call calls close_swap instead. Endpoints that are primarily clients set their default close behavior to be close_swap, endpoints that are primarily servers will default to close.

The performance of this system is limited by how efficiently the endpoint processes <RST> packets. Endpoints that incur a high overhead to handling <RST>s, or delay processing them are not good candidates for this approach.

This system also changes the meaning of the <RST> packet. An <RST> packet currently indicates an unusual condition or error in the connection; this system proposes making it part of standard connection closing procedure.

The TCP solution described in Section 3 does not suffer from the drawbacks associated with using <RST> packets, either in terms of exposing systems to incorrect TCP semantics or in terms of additional processing time for <RST> packets.

[3] These states may both be reported as LAST-ACK to monitoring tools for backward compatibility.

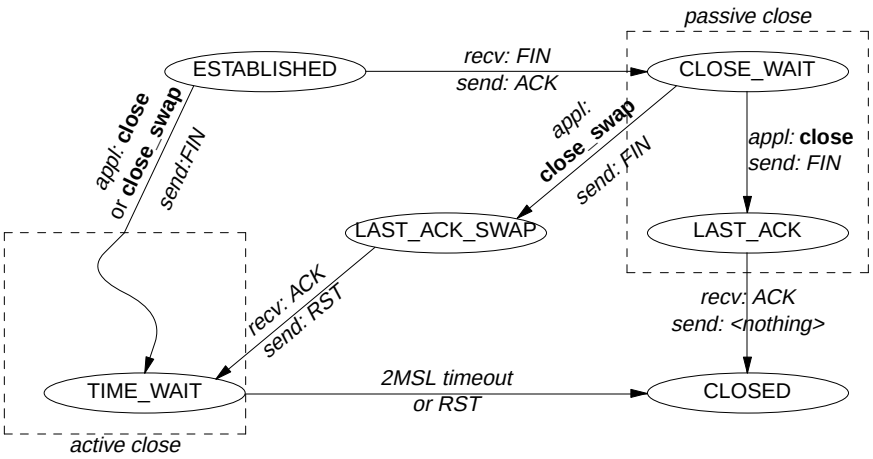


Figure 3: Modified TCP State machine for swapping `TIME-WAIT` states

4.2. Application Level Solution for HTTP

The systems in Section 3 and Section 4.1 both involve changes to the transport protocol which is used by many applications. Directly modifying an application protocol that is loading servers may control the loading problem and minimize the effect on other applications.

This section describes modifications to HTTP that alleviate the contribution of that protocol to `TIME-WAIT` loading. We chose to modify HTTP because it is a major source of client/server Internet traffic.

Early versions of HTTP relied on the closing of the TCP connection to indicate the end of a transaction. Among the changes is HTTP 1.1[2] is the support of persistent connections, a technique that allows clients to pass multiple transactions over the same TCP connection. In order to support persistent connections, the end-of-connection and end-of-transaction indications have been decoupled. This decoupling allows us to modify HTTP to allow clients to actively close connections and therefore hold the `TIME-WAIT` state.

We modify HTTP 1.1 to include a notification from the client that the connection is closed. This notification takes the form of an extension request, called `CLIENT_CLOSE`. An extension request is a new HTTP command, like `PUT` or `POST`, that is not explicitly defined in the HTTP specification[2] A `CLIENT_CLOSE` request requires no reply. It terminates a series of requests on a persistent connection, and indicates to the server that the client has closed the TCP connection. A client will close the TCP connection immediately after sending the `CLIENT_CLOSE` request to the server.

A `CLIENT_CLOSE` request differs from including a `Connection: close` in the header of a request because a request that includes `Connection: close` still requires a reply from the server, and the server will (actively) close the connection[2]. A `CLIENT_CLOSE` request indicates that the client has severed the TCP connection, and that the server should close its end without replying.

`CLIENT_CLOSE` is a minor extension to the HTTP protocol. Current HTTP clients conduct an HTTP transaction by opening the TCP connection, making a series of requests with a `Connection: close` line in the final request header, and collecting the responses. The server closes the connection after sending the final byte of the final request. Modified clients open a connection to the server, make a series of requests, collect the responses, and send a `CLIENT_CLOSE` request to the server after the end of the last response. The client closes the connection immediately after sending the `CLIENT_CLOSE`.

Modified clients are compatible with the HTTP 1.1 specification[2]. A server that does not understand `CLIENT_CLOSE` will see a conventional HTTP exchange, followed by a request that it does not implement, and a closed connection when it tries to send the required error response. A conformant server must be able to handle the client closing the TCP connection at any point. The client has gotten its data, closed the connection and holds the `TIME-WAIT` TCB.

We intend to extend `CLIENT_CLOSE` to include a mechanism for the server to request that the client close the connection. This is analogous to the current `Connection: close` but is initiated by the server and implemented by the client. Under HTTP 1.1, loaded servers are allowed to close persistent connections to reduce their load, but they will incur a `TIME-WAIT` TCB by doing so. Allowing servers to request that the client disconnect sheds the `TIME-WAIT` load at the server as well.

Modifying servers to recognize `CLIENT_CLOSE` can make parts of their implementation easier. Mogul et al. note that discriminating between a persistent connection that is temporarily idle and one that is closed can be difficult for servers because many operating systems do not notify the server that the client has closed the connection until the server tries to read from it[2]. `CLIENT_CLOSE` marks closing connections, which simplifies the server code that detects and closes connections that clients have closed.

Having a client decide when to initiate a CLIENT_CLOSE is somewhat complex. It has to consider user browsing patterns, state of local resources, and the state of server resources. The last may be the trickiest to incorporate. As mentioned above, servers may choose to terminate persistent connections in order to reuse the resources allocated to that connection. Servers need a mechanism to communicate their commitment level to a connection, so that clients and servers are more likely to decide to terminate the same ones[16].

The CLIENT_CLOSE request has been implemented directly in the apache-1.2.4 server[17] and test programs from the WebSTONE performance suite[18]. Patches are available from the authors.

Adopting the HTTP solution is effective if HTTP connections are the a major source of TIME-WAIT loading; however, if another protocol begins loading servers with TIME-WAIT states, that protocol will have to be modified as well. Currently, we believe HTTP causes the bulk of TIME-WAIT loading.

The CLIENT_CLOSE system requires changes only on the client side, although making servers aware of CLIENT_CLOSE may enhance the system's effectiveness. The system conforms to the HTTP 1.1 specification and requires no changes to other protocols. to our knowledge, it creates no new security vulnerabilities.

5. Experiments

In this section we present experiments that demonstrate TIME-WAIT loading and show that our solutions reduce its effects. The proposed solutions have been implemented under SunOS 4.1.3 and initial evaluations of their performance have been made using both custom benchmark programs and the WebSTONE benchmark[18]. The tests were run on workstations connected to the 640 Mb/sec Myrinet LAN.

We performed two experiments. The first experiment shows that TCB load degrades server performance and that our modifications reduce that degradation. The second illustrates that both our TCP and HTTP solutions improve server performance under the WebSTONE benchmark, which simulates typical HTTP traffic. The last experiment shows that our modifications enable a server to support HTTP loads that it cannot in their default configurations.

5.1. Demonstration of Worst-Case Server Loading

The first experiment was designed to determine if TCB load reduces server throughput and if our modifications alleviate that effect. This experiment used four Sparc 20/71's connected by a Myrinet using a user-level data transfer program over TCP. The throughput is the average of each of two client workstations doing a simultaneous bulk transfer to the server. We varied the number of TIME-WAIT TCBs at the server by adding TIME-WAIT states.

The procedure was:

1. Two client machines establish connections to the server
2. The server is loaded with TIME-WAIT TCBs state by a fourth workstation. This workstation established and shut down connections as fast as possible until the server was loaded.
3. The two bulk transport connections transfer data. (Throughput timing begins when the data transfer begins, not when the connection is established. TIME-WAIT TCBs may expire during the transfer.)
4. Between runs, the server was idled until all TIME-WAIT TCBs timed out.

The results are plotted in Figure 4. Each point is the average of ten runs, error bars are standard deviations. The "Modified TCP" curve represents a SunOS system with the TCP modifications from Section 4.1. "Unmodified TCP" represents an unmodified SunOS system. All other parameters remained unchanged.

The experimental procedure is designed to isolate a worst case at the server. The client connections are established first to put them at the end of the list of TCBs in the server kernel, which will maximize the time needed to find them using SunOS's linear search. Two clients are used to neutralize the simple caching behavior in the SunOS kernel, which consists of keeping a single pointer to the most recently accessed TCB. Two distinct clients are used to allow for bursts from the two clients to interleave; two client programs on the same endpoint send bursts in lock-step, which reduces the cost of the TCB list scans.

The experiment shows that under worst case conditions, TCB load can reduce throughput by as much as 50%, and that

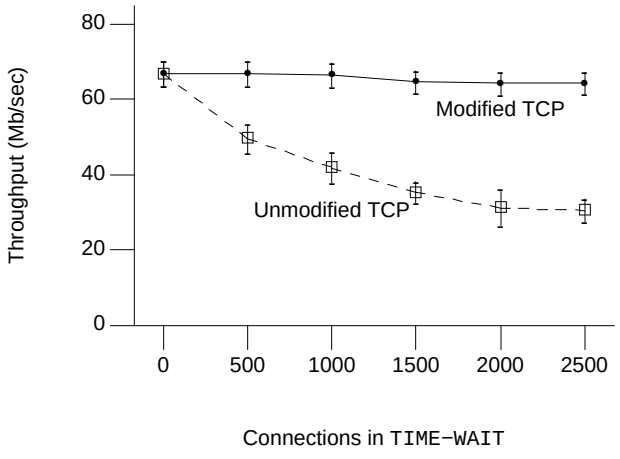


Figure 4: Worst-Case Server Loading

our TCP modifications improve performance under those conditions.

While it is useful that our modifications perform well in the worst case, it is important to asses the worth of the modifications under expected conditions. The previous experiment constructed a worst case scenario; the following experiment uses WebSTONE to test our modifications under more typical HTTP load.

5.2. HTTP Load Experiments

WebSTONE is a standard benchmark used to measure web server performance in terms of connection rate and per connection throughput. To measure server performance, several workstations make HTTP requests of a server and monitor the response time and throughput. A central process collects and combines the information from the individual web clients. We modified the benchmark to measure the amount of memory consumed by TCBs on the server machine. We used WebSTONE version 2 for these experiments. The same workstations and network from Section 5.1 are used in these experiments.

WebSTONE models a heavy load that simulates HTTP traffic. Two workstations run multiple web clients which continuously request files ranging from 9KB to 5MB from the server. Each workstation runs 20 web clients. TCP modifications are as described in Section 4.1 and HTTP modifications are as described in Section 4.2. Results are shown in Table 2.

Both modifications show marked improvements in throughput, connection rate and memory use. TCP modifications increase connection rate by 25% and HTTP modifications increase connection rate by 50%. Server memory requirements are reduced regardless of the HTTP/TCP demultiplexing implementation.

When clients request smaller files, unmodified systems fail completely because they run out of memory; systems using our modifications can support much higher connection rates than unmodified systems. Table 3 reports data from a typical WebSTONE run using 8 clients on 4 workstations connecting to a dedicated server. All clients request only 500 byte files.

System Type	Throughput (Mb/sec)	Conn. per second	TCB Memory (Kbytes)
Unmodified	20.97	49.09	722.7
TCP Mods.	26.40	62.02	23.1
HTTP Mods.	31.73	74.70	23.4

Table 2: `TIME-WAIT` Loading Under WebSTONE

System Type	Throughput (Mb/sec)	Conn. per second	TCB Memory (Kbytes)
Unmodified	fails	fails	fails
TCP Mods.	1.14	223.8	16.1
HTTP Mods.	1.14	222.4	16.1

Table 3: `TIME-WAIT` Loading Under WebSTONE With Small Files

The experiments support the hypothesis that the proposed solutions reduce the memory load on servers. The worst-case experiment shows that the system with a modified TCP performs much better in the worst case, and that server bandwidth loss can be considerable. The WebSTONE experiments shows that both systems reduce memory usage, and that this leads to performance gains. Finally modified systems are able to handle workloads that unmodified systems cannot.

This is a challenging test environment because the TCB load of the server workstations is spread across only two clients rather than the hundreds that would share the load in a real system. The clients suffer some performance degradation due to the accumulating TCBs, much as the server does in the unmodified system.

5.3. `TIME-WAIT` Avoidance and Persistent Connections

The systems proposed are compatible with the per-connection schemes such as persistent connections. To show that our systems improve the memory performance of those systems, we ran WebSTONE experiments using persistent connections and our systems. The experiment used the same network as the experiments described in Section 5.2; two workstations acted as clients, and one as a web server. Each client used the same request pattern as the results in Table 2. Each client issued 5 HTTP requests, waited until they all arrived, and sent 5 more. Each connection served 10 requests in two

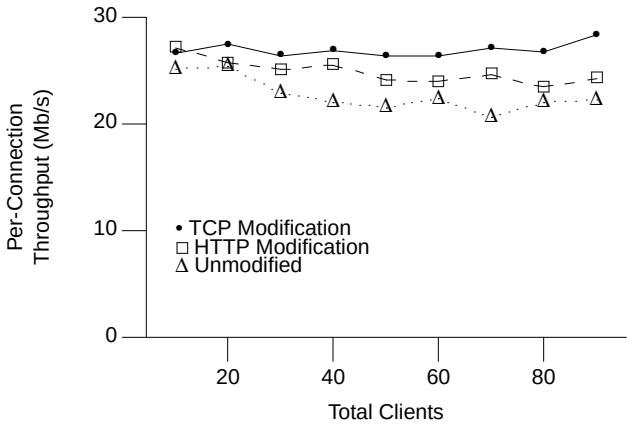


Figure 5: Throughput vs. Clients (Persistent Connections)

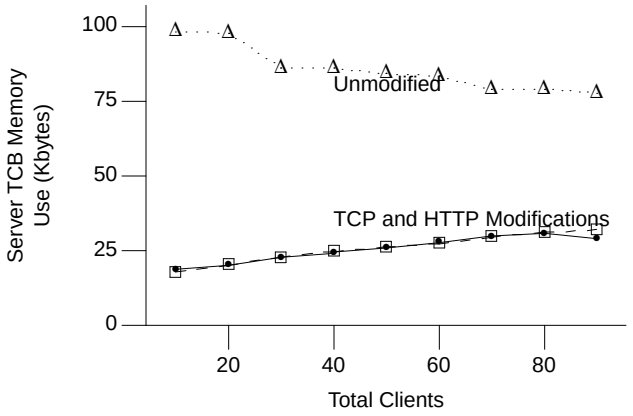


Figure 6: Memory Use vs. Clients (Persistent Connections)

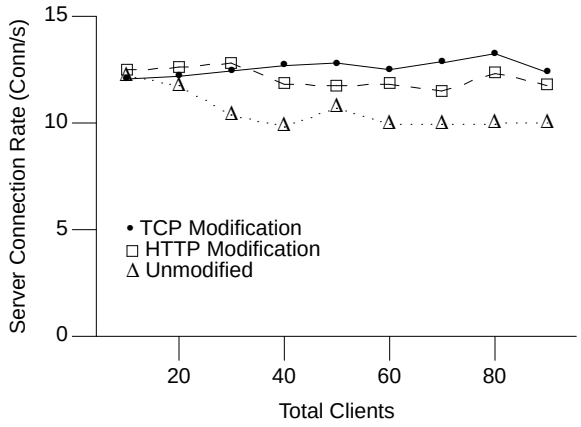


Figure 7: Connection Rate vs. Clients (Persistent Connections)

5-request bursts.

Figure 5 shows how per-connection average client throughput varies with increasing number of clients. Connection throughputs are comparable to those in Table 2, with the difference due primarily to the longer life of these connections. For example, congestion windows will open farther. Our `TIME-WAIT` avoidance methods increase the per-connection throughput as client load increases.

Figure 6 shows that in addition to a modest increase in per-connection throughput, our systems provide significant reduction in the memory used for TCB blocks at servers. That figure plots the number of TCBs in use by the server versus the client load.

It appears from Figure 6 that a simple persistent connection system is showing improved memory performance with increasing client load, but this is not the case. Figure 7 shows that the connection rate decreases with increasing client load in the unmodified system, due to the additional overhead. The memory usage follows the connection rate in the unmodified system. Because Figure 6 includes active TCBs as well as `TIME-WAIT` TCBs, our systems show a linear increase in used TCBs.

6. Conclusions

We have described how TCP interacts with certain application protocols to load servers with `TIME-WAIT` TCBs, and shown examples of this behavior. This interaction is a direct result of the the simplifying assumption that an endpoint's role in closing the connection is independent of its role in establishing the connection.

We have proposed negotiating which endpoint holds the `TIME-WAIT` state during connection establishment, and proposed a negotiation protocol. This system extends TCP functionality to allocate `TIME-WAIT` TCBs to the proper end of the connection without interfering with the semantics of the application protocols or leaving known vulnerabilities in TCP. However, because our proposal involves significant changes to the TCP stack we expect some resistance to its adoption.

We have also implemented and tested a simpler TCP solution and an HTTP solution to show shift the `TIME-WAIT` load from client to server. We have presented experimental evidence that `TIME-WAIT` loading can affect server performance under SunOS 4.1.3. Under these conditions, throughput can be reduced by as much as 50%.

Using WebSTONE, we have shown that HTTP clients and servers using one of our systems exhibit higher throughputs under SunOS 4.1.3. Clients and servers using our system can support higher connection rates than unmodified systems, in certain configurations.

We have shown that our systems combined with persistent HTTP connections use less memory than an otherwise unmodified SunOS 4.1.3 system using persistent connections for a given client load. Our systems interoperate with persistent connections.

Although there are other systems that address the throughput problems we discuss here[11,12], our systems attack the memory loading problem directly. This can reduce the cost of deploying a server, which can be especially important to an embedded or battery powered server.

Table 4 compares the three systems proposed here.

We believe that TCP should eventually be modified to support `TIME-WAIT` negotiation. The coming upgrade from IP version 4 to version 6 represents an opportunity to revisit TCP implementation and design as well. This would be an opportune time to include `TIME-WAIT` state negotiation.

We have also proposed two effective, practical systems for reducing TW load at servers until the more ambitious proposal can be implemented. Adopting one of them would also reduce `TIME-WAIT` loading at servers.

0-7803-5420-6/99/$10.00 (c) 1999 IEEE

	TCP With TIME−WAIT Negotiation	TCP With Client <RST>	CLIENT_CLOSE HTTP Extension
Reduces TIME−WAIT Loading	Yes	Yes	Yes
Compatible With Current Protocols	Yes	Yes	Yes
Changes Are Effective If Only The Client Is Modified	No	Yes	Yes
Allows System To Prevent TIME−WAIT Assassination	Yes	No	Yes
No Changes To Transport Protocol	No	No	Yes
No Changes To Application Protocols	Yes	Yes	No
Adds No Packet Exchanges To Modified Protocol	Yes	No	No
TIME−WAIT Allocation Is A Requirement of Connection Establishment	Yes	No	No

Table 4: Summary of Proposed Systems

References

1. Jon Postel, ed., "Transmission Control Protocol," *RFC-793/STD-7* (September, 1981).
2. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee, "Hypertext Transport Protocol - HTTP/1.1," *RFC-2068* (January, 1997).
3. Robert G. Moskowitz, "Why in the World Is the Web So Slow," *Network Computing*, pp. 22-24 (March 15, 1996).
4. J. Postel and J. K. Reynolds, "File Transfer Protocol," *RFC-959*, USC/Information Sciences Institute (October, 1985).
5. Sun Microsystems, Inc., "Remote Procedure Call Specification," *RFC-1057* (June 1, 1988).
6. Jon Postel, ed., "Internet Protocol," *RFC-791/STD-5* (September 1981).
7. Internet Engineering Task Force, R. Braden, ed., "Requirements for Internet Hosts − Communications Layers," *RFC-1122* (October 1989).
8. W. Richard Stevens, *TCP/IP Illustrated, Volume 1, The Protocols*, Addison-Wesley, Reading, MA, et al. (1994).
9. Gary R. Wright and W. Richard Stevens, *TCP/IP Illustrated, Volume 2 The Implementation*, Addison-Wesley, Reading, MA, et al. (1995).
10. Myricom, Inc., Nannette J. Boden, Danny Cohen, Robert E. Felderman, Alan E Kulawik, Charles L. Seitz, Jakov N. Selovic, and Wen-King Su, "Myrinet: A Gigabit-per-second Local Area Network," *IEEE Micro*, pp. 29-36, IEEE (February 1995).
11. Mike Karels and David Borman, *Personal Communication* (July 1997).
12. Paul E. McKenney and Ken F. Dove, "Efficient Demultiplexing of Incoming TCP Packets," *Proceedings of SIGCOMM 1992*, vol. 22, no. 4, pp. 269-279, Baltimore, MD (August 17-20, 1992).
13. Hendrik Frystyk Nielsen, James Gettys, Anselm Baird-Smith, Eric Prud'hommeaux, Håkon Wium Lie, and Chris Lilley, "Network Performance Effects of HTTP/1.1, CSS1, and PNG," *Proceedings of the SIGCOMM Symposium on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 155-166, Cannes, France (14-18 September 1997).
14. Van Jacobson, Robert Braden, and D. Borman, "TCP Extensions for High Performance," *RFC-1323* (May 1992).
15. R. Braden, "TIME−WAIT Assassination Hazards in TCP," *RFC-1337*, USC/Information Sciences Institute (May 1992).
16. James Gettys, *Personal Communication* (December 1997).
17. Roy T. Fielding and Gail Kaiser, "Collaborative Work: The Apache Server Project," *IEEE Internet Computing*, vol. 1, no. 4, pp. 88-90, IEEE (July/August 1997), available electronically from `http://www.computer.org/internet/ic1997/pdf/w4088.pdf`.
18. Gene Trent and Mark Sake, "WebSTONE: The First Generation in HTTP Server Benchmarking," *white paper*, Silicon Graphics International (February 1995), available electronically from `http://www.sgi.com/Products/WebFORCE/WebStone/paper.html`.

NOVITA ANGGRAINI

192420025

COMPUTER NETWORDK AND COMMUNICATIONS

UAS (MTIK111)

⇨ **SOAL**

ISP idak selalu benar mereka bertaruh bahwa semua pelanggan mereka tidak akan menggunakan semua bandwidth mereka sepanjang waktu. Sehingga ISP memberikan layanan kecepatan internet broadband -contohnya hingga 100 Mbps- kepada pelanggan dengan cara berbagi koneksi kepada pengguna lain. Hal ini mengakibatkan kecepatan internet menurun atau melambat apabila semakin banyak pengguna yang menggunakan layanan internet secara bersamaan pada jaringan yang sama.

Sebagai contoh ketika ada 10 pengguna internet dalam jaringan dengan kecepatan hingga 100 Mbps, mengakses layanan internet hingga 1 Gbps maka setiap pengguna maksimal hanya bisa mencapai kecepatan 10 Mbps. Jika dalam jaringan tersebut pada waktu sibuk pengguna layanan internetnya bertambah menjadi 50 maka kecepatan internet pun akan menurun dan maksimal yang akan di dapat hanya sampai 2 Mbps.

Ketika semakin banyaknya pengguna yang mengakses layanan internet untuk streaming. Downloading maupun browsing hal ini tidak hanya membuat kecepatan menjadi turun atau melambat tetapi dapat menyebabkan antrian data semakin lama dan akhirnya koneksi internet menjadi terputus.
Hal lain yang menyebabkan terjadinya penurunan kecepatan internet adalah data internet itu. sendiri Jika dibandingkan dengan telefon, data telefon hanyalah suara antara pengguna yang satu dengan pengguna yang lain. Sedangkan pada internet merupakan sekumpulan paket data yang dikinmkan dan didistribusikan langsung berupa paket data tersebut.

⇨ **JAWABAN**

Membahas tentang perilaku TCP dari server Web yang sibuk. Dari penelitian (Faber, Touch, & Yue, 1999) saya mencoba memahami bahwa penyebab pemuatan memori, yang disebut *TIME -WAIT loading*, dapat di tentukan tiga metode untuk meringankannya.

Permasalahan awal adalah ketika, jumlah klien terus meningkat, satu-satunya cara untuk menyimpan *TIME-WAIT server* untuk kebutuhan memori konstan untuk meminta *TIME - WAIT TCB* ke klien. Agregasi lebih banyak permintaan per koneksi hanya mengurangi pertumbuhan beban memori bertambah dengan peningkatan beban klien; memindah beban ke *clients distributed server memory load* ke penyebab beban itu.

Membuat jaringan menjadi lebih cepat dan mendukung lebih banyak pengguna, jaringan tingkat koneksi di server yang sedang sibuk meningkat, hasilnya lebih

banyak TIME – WAIT dimuat. Bahkan jika paket *demultiplexing* dilakukan secara efisien, biaya memori TIME - WAIT itu memuat dapat menjadi sumber daya yang signifikan pada sumber daya server. Server akan membutuhkan sumber daya memori fisik tambahan mendukung beban. Untuk server memerlukan adanya TCP, ini diterjemahkan/dihubungkan langsung ke biaya yang lebih tinggi dalam dolar dan daya (listrik).

Mendistribusikan TCB pada skala klien lebih baik daripada pertransaksi memuat reduksi seperti koneksi HTTP persisten untuk mengendalikan *TIME – WAIT loading*. Kurs transaksi didorong oleh peningkatan *bandwidth* dan pertumbuhan jumlah pengguna. Mengurangi setiap biaya transaksi hanya memperlambat tingkat pertumbuhan. Pembongkaran TCB ke klien mendistribusikan memuat lebih adil ketika jumlah klien tumbuh, naik kurva pertumbuhan bukannya mencekiknya. Karena koneksi yang persisten mengurangi biaya per-koneksi lainnya, seperti koneksi tambahan *overhead* pendirian, sistem yang dibuat beroperasi dengan mereka.

Dalam penelitian (Faber dkk., 1999, hlm.) menyajikan tiga sistem untuk mendistribusikan *TIME - WAIT build-up* ke klien. Di sarankan untuk menghindari *TIME - WAIT loading* dengan menegosiasikan titik akhir mana yang akan tahan *TIME − WAIT TCB* selama pembuatan koneksi. Ini memberikan kontrol paling besar atas lokasi TIME – WAIT dengan menjadikan penempatan sebagai bagian eksplisit dari pembentukan koneksi; Namun, melakukan *negotiation dan respecting* ketika koneksi ditutup membutuhkan perubahan signifikan ke TCP.

Mengingat hal ini, penelitian ini juga membahas dua alternatif yang kurang invasif: modifikasi TCP yang menggeser TIME − WAIT TCB dari server ke klien ketika koneksi ditutup; dan modifikasi HTTP yang mendukung klien menutup koneksi dan menahan TIME − WAIT TCB.

- ⇨ **HASIL**

Hasilnya dalam penelitian (Faber dkk., 1999) mengusulkan negosiasi titik akhir yang mana memegang status TIME - WAIT selama pembentukan koneksi, dan diusulkan protokol negosiasi.

Sistem ini memperluas fungsionalitas TCP untuk mengalokasikan WAKTU − TUNGGU TCB ke akhir koneksi yang tepat tanpa mengganggu semantik dari protokol aplikasi atau meninggalkan kelemahan yang diketahui dalam TCP. Namun, karena penelitian ini melibatkan signifikan perubahan pada TCP *stack* mereka mengharapkan beberapa penolakan terhadap adopsinya. Penelitian ini juga menerapkan dan menguji solusi TCP yang lebih sederhana dan solusi HTTP untuk menunjukkan pengalihan TIME – WAIT memuat dari klien ke server.

Disana juga telah mengeksperimen, bahwa WAIT - TIME loading dapat mempengaruhi kinerja server di bawah SunOS 4.1.3. Dalam kondisi ini, *throughput* (hasil) dapat dikurangi sebanyak 50%. Menggunakan WebSTONE, penelitian ini telah menunjukkan bahwa klien HTTP dan server yang menggunakan salah satu sistem, menunjukkan *throughput* (hasil) yang lebih tinggi di bawah SunOS 4.1.3.

Klien dan server menggunakan sistem yang dibuat mendukung tingkat koneksi yang lebih tinggi daripada sistem yang tidak dimodifikasi, di konfigurasi tertentu.

Dengan dikombinasikan dengan persisten, Koneksi HTTP menggunakan lebih sedikit memori daripada yang lain sistem SunOS 4.1.3 yang tidak dimodifikasi menggunakan koneksi persisten untuk beban klien yang diberikan. Sistem yang dibuat beroperasi dengan persisten koneksi.

Meskipun ada sistem lain yang menangani throughput masalah yang kita diskusikan di sini adalah sistem yang menyerang masalah pemuatan memori secara langsung. Ini dapat mengurangi biaya menyebarkan server, yang bisa sangat penting bagi sebuah server tertanam atau baterai. **Gambar 1** membandingkan tiga sistem yang diusulkan di sini.

	TCP With TIME-WAIT Negotiation	TCP With Client <RST>	CLIENT_CLOSE HTTP Extension
Reduces TIME-WAIT Loading	Yes	Yes	Yes
Compatible With Current Protocols	Yes	Yes	Yes
Changes Are Effective If Only The Client Is Modified	No	Yes	Yes
Allows System To Prevent TIME-WAIT Assassination	Yes	No	Yes
No Changes To Transport Protocol	No	No	Yes
No Changes To Application Protocols	Yes	Yes	No
Adds No Packet Exchanges To Modified Protocol	Yes	No	No
TIME-WAIT Allocation Is A Requirement of Connection Establishment	Yes	No	No

Gambar 1. Summary of Proposed Systems

Dari penelitian ini, membuat saya yakin bahwa TCP pada akhirnya harus dimodifikasi untuk mendukung TIME−WAIT negotiation. *Upgrade* yang datang dari IP versi 4 hingga versi 6 merupakan peluang untuk mengunjungi kembali Implementasi dan desain TCP juga. Ini akan menjadi waktu yang tepat untuk memasukkan TIME−WAIT *state negotiation*.

Penelitian ini juga mengusulkan dua sistem praktis dan efektif untuk mengurangi beban TW di server hingga proposal yang lebih ambisius dapat diimplementasikan. Mengadopsi salah satu dari mereka juga akan melakukannya mengurangi TIME − TUNGGU memuat di server, dimana perbandingan dapat dilihat pada **Gambar 1** diatas.

⇨ DAFTAR PUSTAKA

Faber, T., Touch, J., & Yue, W. (1999). The TIME-WAIT state in TCP and its effect on busy servers. *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future Is Now (Cat. No.99CH36320)*, 1573–1583 vol.3. https://doi.org/10.1109/INFCOM.1999.752180

192420033 – RIKA SEFTIANA

UJIAN AKHIR SEMESTER (UAS)
COMPUTER NETWORK AND BUSINESS INFORMATION

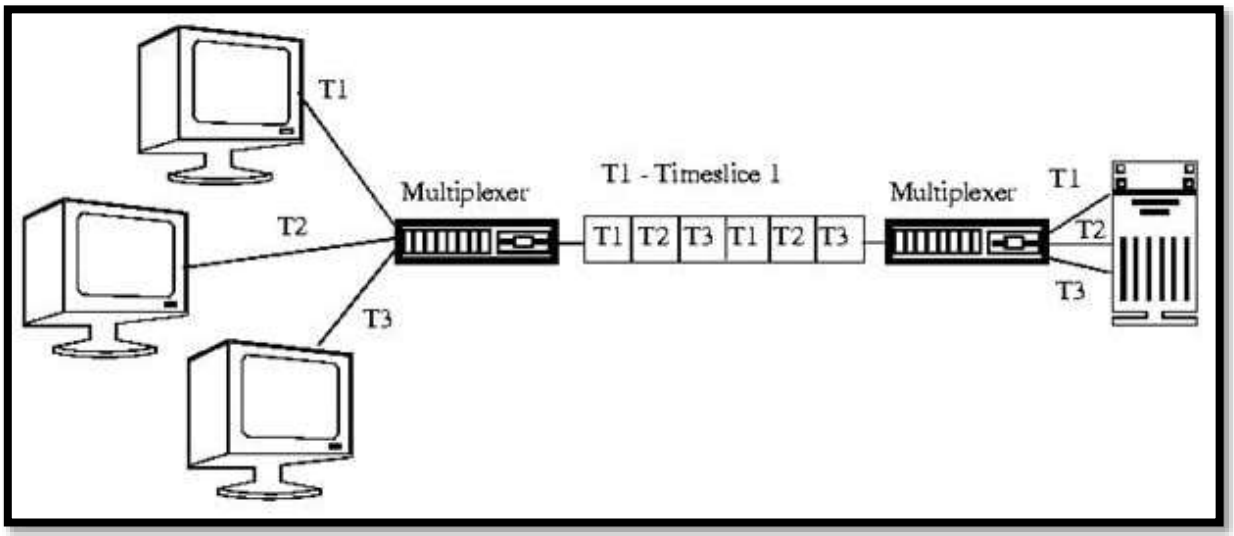
STATISTIK MULTIPLEXING

Statistik Multiplexing adalah teknik multiplexing yang memungkinkan informasi dari sejumlah saluran untuk digabungkan untuk transmisi melalui satu saluran. Multiplexing statistik adalah jenis berbagi tautan komunikasi, sangat mirip dengan alokasi bandwidth dinamis (DBA). Dalam multiplexing statistik, saluran komunikasi dibagi menjadi sejumlah saluran digital bitrate variabel atau aliran data. Berbagi tautan disesuaikan dengan tuntutan lalu lintas sesaat dari aliran data yang ditransfer melalui setiap saluran. Ini merupakan alternatif untuk membuat pembagian tautan yang tetap, seperti dalam multiplexing pembagian waktu umum (TDM) dan multiplexing pembagian frekuensi (FDM). Ketika dilakukan dengan benar, multiplexing statistik dapat memberikan peningkatan pemanfaatan tautan, yang disebut gain multiplexing statistik.

Multiplexing statistik difasilitasi melalui mode paket atau komunikasi berorientasi paket, yang antara lain digunakan dalam jaringan komputer packet switched. Setiap aliran dibagi ke dalam paket-paket yang biasanya dikirimkan secara serempak dengan cara yang pertama datang pertama dilayani. Dengan cara alternatif, paket-paket dapat dikirimkan sesuai dengan beberapa disiplin penjadwalan untuk antrian yang adil atau dibedakan dan / atau kualitas layanan yang terjamin. Multiplexing statistik dari saluran analog, misalnya saluran nirkabel, juga difasilitasi melalui skema berikut:

- Pembagian frekuensi orthogonal random-hopping frekuensi ganda (RFH-OFDMA)
- Code multiple division-division multiple access (CDMA), di mana jumlah kode penyebaran atau faktor penyebaran yang berbeda dapat ditetapkan untuk pengguna yang berbeda.

Multiplexing statistik biasanya menyiratkan layanan "sesuai permintaan" dan bukan layanan yang mengalokasikan sumber daya untuk setiap aliran data. Skema multiplexing statistik tidak mengontrol transmisi data pengguna.



Gambar Statistik Multiplexing

Cara Kerja Multiplexing Statistik secara dinamis mengalokasikan bandwidth ke setiap saluran berdasarkan kebutuhan. Ini berbeda dengan teknik time-division multiplexing (TDM), di mana perangkat senyap menggunakan sebagian dari aliran data multiplexing, mengisinya dengan paket kosong. Multiplexing statistik mengalokasikan bandwidth hanya untuk saluran yang saat ini mentransmisikan. Ini mengemas data dari saluran aktif ke dalam paket dan secara dinamis memasukkannya ke saluran keluaran, biasanya berdasarkan FIFO (masuk pertama, keluar pertama), tetapi juga dapat mengalokasikan bandwidth ekstra ke saluran input tertentu.

Perangkat multiplexing statistik biasanya mendukung fitur-fitur lain, seperti berikut ini:

- Kemampuan deteksi dan koreksi kesalahan store-and-forward:
  Mengidentifikasi saluran mana yang mengirim setiap paket data dan memperbaiki kesalahan yang terjadi
- Kompresi data:
  Meningkatkan jumlah data yang dapat dikirim per paket

Multiplexing statistik kadang-kadang disebut sebagai time-division multiplexing (STDM) atau Statistical Package Multiplexing (SPM), tetapi istilah yang lebih pendek digunakan lebih sering. Multiplexer yang mampu secara multiplexing secara statistik beberapa aliran data bersama-sama kadang-kadang disebut statmux. Jika Anda memiliki statmux di setiap ujung jalur digital, statmux penerima dapat mengidentifikasi saluran dari setiap paket yang dikirim oleh statmux pengiriman dan demultiplex aliran data ke saluran data aslinya.

Dalam penyiaran audio dan video digital, misalnya, multiplexer statistik adalah perangkat penghimpun konten yang memungkinkan penyiar untuk memberikan jumlah terbesar dari layanan audio atau video untuk bandwidth tertentu dengan berbagi kumpulan bandwidth tetap di antara beberapa layanan atau aliran bitrate yang bervariasi. . Multiplexer mengalokasikan untuk setiap layanan bandwidth yang diperlukan untuk kebutuhan real-time sehingga layanan dengan adegan kompleks menerima bandwidth lebih banyak daripada layanan dengan layanan yang kurang kompleks. Teknik berbagi bandwidth ini menghasilkan kualitas video terbaik dengan bandwidth agregat serendah mungkin. Contoh multiplexer statistik termasuk lini produk BNPXr Imagine Communications (RGB Networks), Harmonic Inc. ProStream, Electra dan keluarga produk VOS atau Motorola (Terayon) DM64

Nama : Rudi Setiawan
Nim : 192420029
Kelas : MTI Reg B angkatan 21

Why Internet Slows Down When it's Busy

Analisa:

Dari Video tersebut pembicara menjelaskan mengenai akses internet yang diberlakukan oleh ISP ke customer, dimana secara teknis akses yang diberikan tidak murni sebesar bandwith yang ditawarkan untuk kondisi jika semua customer menggunakan full bandwith secara bersamaan pada ISP tersebut. Hal ini disebabkan karena ISP menggunakan teknik Multiplexing dalam mendistribusi bandwith ke customer, dimana ISP sendiri meyakini bahwa tidak semua customer menggunakan full bandwith diwaktu bersamaan, sehingga bandwith yang tidak digunakan/sedikit digunakan oleh beberapa customer akan dishare ke customer lain yang mengunakan lebih tinggi diwaktu bersamaan.

Saya menganalisa bahwa teknik ini digunakan ISP dengan memanfaatkan link internet yang dimilikinya untuk memperoleh jumlah customer yang lebih banyak, jika dibandingkan ISP mendisribusikan link nya secara utuh/real ke customer. Sehingga ini akan berdampak pada keuntungan bisnis ISP itu sendiri.

Meskipun ISP sendiri dalam melakukan teknik multilexing berdasarkan penelitian terhadap pola penggunaan/pattern internet oleh customer (statistic multiplexing), tentunya peluang terjadinya internet lambat pasti terjadi, ini tentunya harus menjadi perhatian bagi ISP.

Issue berkembang dari Analisa:

Service dari internet ISP yang memberlakukan multiplexing pada bandwith customer bedasarkan statistical multiplexing jika diperhatikan lebih dalam lagi akan sangat berdampak sekali terutama jika kita sebagai customer bisnis yang menggunakan internet sebagai sumber layanan/server (sebagai aplikasi web https, api-rest, ftp server, dan lainnya) untuk client-client kita, dimana jika link internet ISP tersebut dalam posisi load tinggi, maka sudah dipastikan client-client kita akan mengalami delay, bahkan drop koneksi dan hal ini tentunya dapat merugikan, karena berdampak pada availabilty layanan, serta tidak menutup kemungkinan akan timbul banyak komplain yang berdampak pada bisnis itu sendiri.

Solusi terhadap Issue:

Dari Issue yang berkembang diatas, saya mengusulkan untuk tidak menggunakan hanya satu link internet dari satu ISP saja, melainkan menggunakan beberapa link dari ISP yang berbeda untuk layanan aplikasi kita terhadap client kita. Dimana saat ini harga link internet publik semakin murah dan ini bertujuan untuk memastikan availability dari layanan dari aplikasi kita terhadap issue sebagai disampaikan diatas tidak terjadi.

Secara best practice multi link ISP ini sendiri akan dimanajemen mengunakan teknologi pembagi jalur jaringan berupa perangkat Link Controller, dimana teknologi pada perangkat ini akan memanajemen Link-link ISP yang kita gunakan tersebut dan melakukan re-route koneksi untuk client berdasarkan tipe dan kualitas link pada saat client akses ke server, dengan demikian jika saat salah satu link ISP mengalami penurunan kuliatas, atau terputus, atau pun load jaringannya tinggi, maka client kita akan diarakan ke link ISP yang lain yang kualiatasnya lebih baik, sehingga client tidak akan merasakan adanya hambatan saat mengakses layanan aplikasi kita.

Nama : Sapardi
Nim : 192420026
Kelas : MTI Reg B angkatan 21

Why Internet Slows Down When it's Busy

Analisa:

Dari Video tersebut pembicara menjelaskan mengenai akses internet yang diberlakukan oleh ISP ke customer, dimana secara teknis akses yang diberikan tidak murni sebesar bandwith yang ditawarkan untuk kondisi jika semua customer menggunakan full bandwith secara bersamaan pada ISP tersebut. Hal ini disebabkan karena ISP menggunakan teknik Multiplexing dalam mendistribusi bandwith ke customer, dimana ISP sendiri meyakini bahwa tidak semua customer menggunakan full bandwith diwaktu bersamaan, sehingga bandwith yang tidak digunakan/sedikit digunakan oleh beberapa customer akan dishare ke customer lain yang mengunakan lebih tinggi diwaktu bersamaan.

Saya menganalisa bahwa teknik ini digunakan ISP dengan memanfaatkan link internet yang dimilikinya untuk memperoleh jumlah customer yang lebih banyak, jika dibandingkan ISP mendisribusikan link nya secara utuh/real ke customer. Sehingga ini akan berdampak pada keuntungan bisnis ISP itu sendiri.

Meskipun ISP sendiri dalam melakukan teknik multilexing berdasarkan penelitian terhadap pola penggunaan/pattern internet oleh customer (statistic multiplexing), tentunya peluang terjadinya internet lambat pasti terjadi, ini tentunya harus menjadi perhatian bagi ISP.

Issue berkembang dari Analisa:

Service dari internet ISP yang memberlakukan multiplexing pada bandwith customer bedasarkan statistical multiplexing jika diperhatikan lebih dalam lagi akan sangat berdampak sekali terutama jika kita sebagai customer bisnis yang menggunakan internet sebagai sumber layanan/server (sebagai aplikasi web https, api-rest, ftp server, dan lainnya) untuk client-client kita, dimana jika link internet ISP tersebut dalam posisi load tinggi, maka sudah dipastikan client-client kita akan mengalami delay, bahkan drop koneksi dan hal ini tentunya dapat merugikan, karena berdampak pada availabilty layanan, serta tidak menutup kemungkinan akan timbul banyak komplain yang berdampak pada bisnis itu sendiri.

Solusi terhadap Issue:

Dari Issue yang berkembang diatas, saya mengusulkan untuk tidak menggunakan hanya satu link internet dari satu ISP saja, melainkan menggunakan beberapa link dari ISP yang berbeda untuk layanan aplikasi kita terhadap client kita. Dimana saat ini harga link internet publik semakin murah dan ini bertujuan untuk memastikan availability dari layanan dari aplikasi kita terhadap issue sebagai disampaikan diatas tidak terjadi.

Secara best practice multi link ISP ini sendiri akan dimanajemen mengunakan teknologi pembagi jalur jaringan berupa perangkat Link Controller, dimana teknologi pada perangkat ini akan memanajemen Link-link ISP yang kita gunakan tersebut dan melakukan re-route koneksi untuk client berdasarkan tipe dan kualitas link pada saat client akses ke server, dengan demikian jika saat salah satu link ISP mengalami penurunan kuliatas, atau terputus, atau pun load jaringannya tinggi, maka client kita akan diarakan ke link ISP yang lain yang kualiatasnya lebih baik, sehingga client tidak akan merasakan adanya hambatan saat mengakses layanan aplikasi kita.

Nama : sulistiyani / 182420044

Mk : Computer Network and Communications

mengapa internet melambat ketika banyak yang akses ?

metode yang dapat kami berikan untuk menyelesaikan case tersebut :

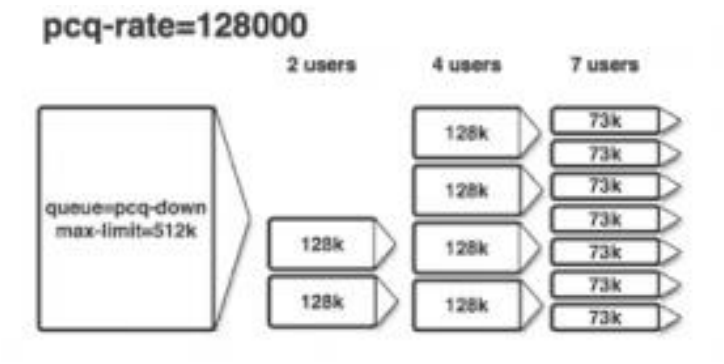
Queue Tree & PCQ

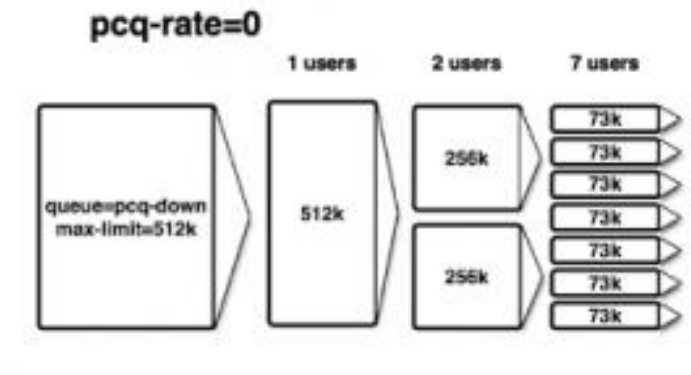
- Queue Tree berfungsi untuk mengimplementasikan fungsi yang lebih komplex dalam limit bandwidth, dimana penggunaan packet mark nya memiliki fungsi yang lebih baik. Digunakan untuk membatasi satu arah koneksi saja baik itu download maupun upload. Secara umum Queue Tree ini tidak terlihat berbeda dari Simple Queue. Perbedaan yang bisa kita lihat langsung yaitu hanya dari sisi cara pakai atau penggunaannya saja. Dimana Queue Simple secara khusus memang dirancang untuk kemudahan konfigurasi sementara Queue Tree dirancang untuk melaksanakan tugas antrian yang lebih kompleks dan butuh pemahaman yang baik tentang aliran trafik.
- PCQ (Per Connection Queuing)

  Digunakan untuk mengenali arah arus dan digunakan karena dapat membagai bandwidth secara adil, merata dan masif. PCQ digunakan bersamaan dengan fitur Queue, baik Simple Queue maupun Queue Tree.

  PCQ Classifier berfungsi mengklasifikasikan arah koneksi, Misalnya jika Classifier yang digunakan adalah *src-address* pada Local interface, maka aliran pcq akan menjadi koneksi upload. Begitu juga dgn *dst-address* akan menjadi pcq download.

  PCQ rate berfungsi untuk membatasi bandwidth maksimum yang bisa didapatkan. Dengan memasukkan angka pada rate ini (default: 0) maka maksimal download yang akan didapatkan per IP akan dibatasi mis. 128k (kbps).





Limit berfungsi untuk membatasi jumlah koneksi paralel yang diperkenankan bagi tiap IP. artinya bila kita meletakkan nilai 50, maka cuma 50 koneksi simultan yang bisa didapat oleh 1 IP address (baik itu source / destination).

Total Limit adalah total keseluruhan koneksi paralel yang diperkenankan untuk seluruh ip addresss (baik itu source ataupun destination).

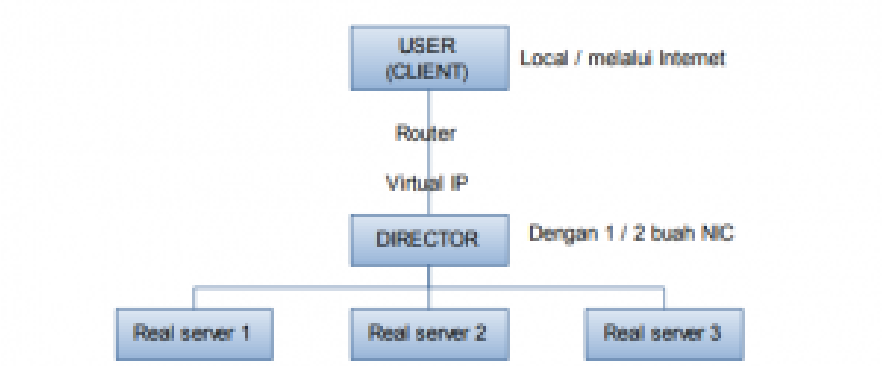
2. LVS Linux Virtual Server dengan algoritma Rond robin

- Selain itu dengan load balancer kita dapat mengarahkan client ke web server yang lain jika terjadi web server overload atau web server sedang down, hal ini sangat bermanfaat, baik dari sisi kecepatan akses maupun efisiensi waktu pada saat user mengakses sebuah web. Untuk menganalisis algoritma scheduling di linux virtual server (LVS), dilakukan perancangan, implementasi dan pengukuran waktu respon.
- Pada rancangan model sistem LVS (Linux Virtual server) dilakukan pengujian dengan tahap sebagai berikut:
    1) Client melakukan requests ke server melalui load balancer dengan tool httperf dengan jumlah koneksi user antara 10.000/s – 50.000/s.

2) Komputer load balancer menerima requests dari client kemudian melakukan scheduling dan rewriting packets. Pada proses ini load balancer mengirimkan request ke real server yang sedang aktif dengan menggunakan algoritma round robin.
3) Real server menerima request dari client melalui komputer load balancer dan mengirimkan reply ke load balancer.
4) Komputer load balancer menerima reply dari real server yang sedang aktif, kemudian meneruskan reply ke komputer client yang meminta request. Client menerima replies dari load balancer. Pada proses ini komputer client dapat mengetahui respon time dari sebuah web server.

- Linux Virtual Server atau disingkat LVS merupakan suatu teknologi clustering yang dapat digunakan untuk membangun suatu server dengan menggunakan kumpulan dari beberapa buah realserver. LVS merupakan implementasi dari komputer cluster dengan metoda High Availibility. LVS mengimbangi berbagai bentuk dari service jaringan pada banyak mesin dengan memanipulasi paket sebagaimana diproses TCP/IP stack. Satu dari banyak peran yang paling umum dari Linux Virtual Server adalah bertindak sebagai server yang berada pada garis terdepan dari kelompok server web.

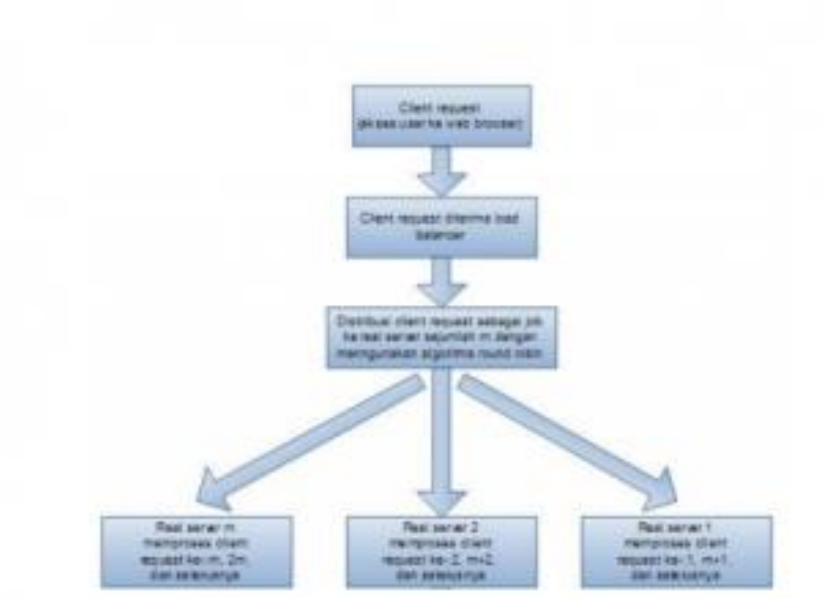
  Seperti ditunjukkan pada Gambar 1 Linux Virtual Server atau LVS ini terdiri dari sebuah Director dan beberapa realserver yang bekerja bersama dan memberikan servis terhadap permintaan user. Permintaan User diterima oleh Director yang seolah olah berfungsi sebagai IP Router yang akan meneruskan paket permintaan user tersebut pada real server yang siap memberikan servis yang diminta. Dengan demikian virtual server akan terdiri dari beberapa komputer Yang mempunyai image yang sama tetapi ditempatkan pada IP yang berbeda. User dapat mengakses virtual server tersebut dengan bantuan suatu Director, yang bertugas untuk melakukan pemetaan IP dari server dan komputer lainnya yang berperan sebagai virtual server.



LVS Director adalah modifikasi dari sistem Linux yang bertanggung jawab untuk mendistribusikan permintaan user/client terhadap realserver pada kelompok server. Realserver melakukan pekerjaan untuk memenuhi permintaan serta memberikan atau membuat laporan balik kepada user/client. LVS Director memelihara rekaman daripada sejumlah permintaan yang telah ditangani oleh masing-masing realserver dan menggunakan informasi ini ketika memutuskan server mana yang akan ditugaskan untuk menangani suatu permintaan berikutnya. LVS juga dapat memiliki Director cadangan yang akan menggantikan bilamana suatu saat Director utama mengalami suatu kegagalan sehingga membentuk suatu LVS failover. Masing-masing realserver dapat bekerja dengan menggunakan berbagai sistem operasi dan aplikasi yang mendukung TCP/IP dan Ethernet. Pembatasan dan pemilihan sistem operasi pada realserver serta jenis servis yang didukung oleh realserver dilakukan pada saat proses konfigurasi LVS dijalankan.

Algoritma Penjadwalan (Scheduling) Round Robin

Mekanisme penjadwalan pada LVS dikerjakan oleh sebuah patch kernel yang disebut modul IP Virtual Server atau IPVS modules. Modul ini mengaktifkan layer 4 yaitu transport layer switching yang dirancang dapat bekerja dengan baik pada multi server dalam IP address tunggal (virtual IP address). IPVS membuat IPVS table pada kernel untuk menelusuri dan merutekan paket ke real server secara efisien. Tabel ini digunakan oleh load balancer yang sedang aktif (yang pasif adalah backup-nya) untuk meneruskan client request dari virtual IP address ke real server. IPVS table secara rutin diperbarui menggunakan software ipvsadm.



Pada penjadwalan tipe round-robin, manager mendistribusikan client request sama rata ke seluruh real server tanpa memperdulikan kapasitas server ataupun beban request. Jika ada tiga real server (A,B,C), maka request 1 akan diberikan manager kepada server A, request 2 ke server B, request 3 ke server C dan request 4 kembali ke server A.

Mekanisme ini dapat dilakukan jika seluruh real server menggunakan spesifikasi komputer yang sama. Konsep dasar dari algoritma ini adalah dengan menggunakan time-sharing. Pada dasarnya algoritma ini sama dengan FCFS, hanya saja bersifat preemptive. Setiap proses mendapatkan waktu CPU yang disebut dengan waktu quantum (quantum time) untuk membatasi waktu proses, biasanya 1-100 milidetik. Setelah waktu habis, proses ditunda dan ditambahkan pada ready queue. Algoritma Round Robin merupakan algoritma yang paling sederhana dan banyak digunakan oleh perangkat load balancing . Algoritma ini membagi beban secara bergiliran dan berurutan dari satu server ke server lain sehingga membentuk putaran. Penjadwalan ini merupakan:

1. Penjadwalan preemptive, bukan di-preempt oleh proses lain, tapi terutama oleh penjadwal berdasarkan lama waktu berjalannya proses, disebut preempt by time.

2. Penjadwal tanpa prioritas. Semua proses dianggap penting dan diberi sejumlah waktu proses yang disebut kwanta (quantum) atau time slice dimana proses itu berjalan.

Ketentuan algoritma round robin adalah sebagai berikut:

1. Jika kwanta dan proses belum selesai maka proses menjadi runnable dan pemroses dialihkan ke proses lain.

2. Jika kwanta belum habis dan proses menunggu suatu kejadian (selesainya operasi I/O), maka proses menjadi blocked dan pemroses dialihkan ke proses lain.

3. Jika kwanta belum habis tapi proses telah selesai, maka proses diakhiri dan pemroses dialihkan ke proses lain. Algoritma penjadwalan ini dapat diimplementasi sebagai berikut: – Mengelola senarai proses read (runnable) sesuai urutan kedatangan.

Ambil proses yang berada di ujung depan antrian menjadi running. – Bila kwanta belum habis dan proses selesai maka ambil proses di ujung depan antrian proses ready. – Jika kwanta habis dan proses belum selesai maka tempatkan proses running ke ekor antrian proses ready dan ambil proses di ujung depan antrian proses ready.

Tentukan masalah dan solusi dari video "Why internet slows down when it's busy – Computerphile"

Yudy Pranata (192420001)
Mahasiswa Magister Teknik Informatika Universitas Bina Darma
Jl. Jend. A. Yani No. 12, Plaju, Palembang 30264
yudypranata26@gmail.com

Pada video "*Why internet slows down when it's busy – Computerphile*" yang telah diberikan, terdapat penjelasan yang menyebutkan bahwa penyedia jasa jaringan internet selalu menjanjikan kapasitas besar dan cepat dalam jaringan untuk menarik banyak peminat. Seperti didalam video dengan contoh penyedia jasa jaringan internet memiliki total kapasitas jaringan server sebesar 1 (satu) Gb/sec dan mereka memberikan "janji" kecepatan dan kapasitas jaringan internet kepada pengguna (konsumen) 100 Mb/sec. Tetapi yang sebenarnya mereka tidak bisa menjanjikan bahwa kapasitas tersebut dengan penuh dapat digunakan, karena pada saat menggunakan internet kapasitas jaringan yang diberikan tetap mengacu pada jumlah pengguna (konsumen) dan seberapa besar menggunakan kapasitas jaringan tersebut pada saat yang bersamaan.

Solusi

Berdasarkan pada penelitian *TIME-WAIT loading* [1] dalam percobaan dilakukan pada sebuah *workstation* yang terhubung ke Myrinet LAN sebesar 640 Mb/sec [2]. Dengan adanya 3 (tiga) penelitian yang pertama membuktikan bahwa beban TCB menurunkan kinerja server dan modifikasi yang digunakan akan menurunkan degradasi tersebut. Penelitian kedua menggambarkan dari kedua solusi TCP [3] dan HTTP [4] dapat meningkatkan kinerja server dibawah tolak ukur WebSTONE [5]. Dan penelitian yang terakhir menggambarkan bahwa modifikasi yang dibuat memungkinkan server untuk mendukung beban HTTP yang tidak dapat digunakan pada konfigurasi standar. Dapat dijelaskana bagaimana TCP berinteraksi dengan protokol aplikasi tertentu untuk memuat server dengan *TIME-WAIT* TCB dan menunjukkan dari tindakan ini. Interaksi tersebut adalah hasil langsung dari penyederhanaan asumsi bahwa titik akhir dalam penutupan koneksi tidak bergantung untuk selanjutnya membangun kembali koneksi tersebut. Dengan mangajukan usulan pada titik akhir yang dapat menahan *TIME-WAIT* selama pembentukan kembali koneksi. System tersebut akan memperluan fungsi TCP untuk mengalokasikan *TIME-WAIT* TCB ke ujung koneksi yang tepat tanpa mengganggu protokol aplikasi atau meninggalkan kelemahan pada TCP. Pada tahap implementasi dan menguji solusi pada TCP yang lebih sederhana dan solusi pada HTTP untuk menunjukkan beban

TIME-WAIT dari klien ke server dan dalam kondisi tersebut dapat dipastikan bahwa beban *TIME-WAIT* dapat mempengaruhi kinerja server dibawah SunOS 4.1.3 yang hasilnya dpaat berkurang sebanyak 50 %. Sementara itu, dengan menggunakan WebSTONE dapat dibuktikan bahwa HTTP pada klien dan server yang menggunakan system mereka memiliki hasil yang lebih besar dibandingkan dengan SunOS 4.1.3. Klien dan server yang telah menggunakan system konfigurasi mereka dapat mendukung tingkat koneksi yang lebih tinggi dari pada system yang belum dimodifikasi pada konfigurasi tertentu. Mereka juga telah menunjukkan bahwa sistemnya dapat dikombinasikan dengan koneksi HTTP yang persisten menggunakan lebih sedikit memori daripada system SunOS 4.1.3 yang tidak dimodifikasi menggunakan koneksi untuk beban klien tertentu.

	TCP with TIME-WAIT Negotation	TCP with Clien <RST>	CLIENT_CLOSE HTTP Extension
Reduces TIME-WAIT loading	Yes	Yes	Yes
Compatible with Current Protocols	Yes	Yes	Yes
Changes are Effective if only the client is modified	No	Yes	Yes
Allows system to prevent TIME-WAIT assassination	Yes	No	Yes
No changes to Transport Protocol	No	No	Yes
No changes to Application Protocols	Yes	Yes	No
Adds no packet exchanges to Modified Protocol	Yes	No	No
TIME-WAIT allocation is a requirement of connection establishment	Yes	No	No

Table 4: Summary of Proposed Systems [1]

Referensi

1. Faber Theodore, Joseph D. Touch, and W. Yue. The TIME-WAIT state in TCP and its effect on busy servers. In *Proceedings of IEEE Infocom* (March 1999).
2. Myricom, Inc., Nannette J. Boden, Danny Cohen, Robert E. Felderman, Alan E Kulawik, Charles L. Seitz, Jakov N. Selovic, and Wen-King Su, “Myrinet: A Gigabit-per-second Local Area Network.” *IEEE Micro,* pp.29-36, IEEE (February 1995).
3. Jon Postel, ed., “Transmission Control Protocol,” *RFC-793/STD-7* (September, 1981).
4. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee, “Hypertext Transport Protocol – HTTP/1.1,” *RFC-2068* (January, 1997).
5. Gene Trent and Mark Sake, “WebSTONE: The First Generation in HTTP Server Benchmarking,” *white paper*, Silicon Graphics International (February, 1995).

Nama : A.Firdaus

Nim : 1924200423

Kelas : MTIR2

Tugas

Pertanyaan : Mengapa Internet melambat ketika jaringan sedang sibuk dari studi kasus diberikan melalui video yang diunggah oleh Dr Richard Mortier

A. Latar Belakang :

Menurut jurnal Ilmu Komputer Universitas California :

Dalam kasus tersebut dimana pertumbuhan internet semakin pesat dan banyak transisi perpindahan akses peneliti menganalsis dengan dua metode yaitu single Connection Behavior dan Parallel Connection Behavior

1. Single Connection Behavior :
   a. Loss Recovery : Bagaimanakah Pemulihan dan efektifitas pemulihan data dalam menghindari saat timeout
   b. Reciever Bottleneck : Berapa Seringkah Kinerja TCP menerima halaman dengan waktu yang terbatas
   c. Ack compression : Seberapa sering terjadi act compression saat kehilangan data
2. Parallel Connection Behavior :
   a. Bagaimana clien melihat jumlah koneksi saat server dibuka
   b. Congestion control: bagaimana reaksi koneksi pararel menerima saat kehilangan data yang disebabkan koneksi timeout
   c. Loss behavior : Bagaimana data yang didistribusikan saat koneksi sedang padat

B. Analisa :

DidalamPenelitan tersebut dijelaskan bagaimana cara mengalisa tentang koneksi TCP saat koneksi sedang sibuk dan seberapa baik pemulihan yang

terjadi. Ada beberapa analisa tentang bagaimana cara memperbaiki data yang diterima saat melakukan koneksi jaringan antara lain :

1. **Single Connection Behavior**

Total connections	16501	
With packet re-ordering	9703	6
With receiver window as	2339	14
Total packets	78216	
During slow-start	66620	85
# of slow-start pkts lost	3545	5
During congestion	11595	15
# of congestion	8218	7
Total retransmissions	8571	
Fast retransmissions	3753	44
Slow-start retransmissions	5981	7
Coarse timeouts	4220	49
Avoidable with SACKs	1871	4
Avoidable with enhanced	1042	25

Pada tabel diatas saat melakukan single connection dimana ukuran data yang panggil lebih besar daripada data yang diterima

2. **Parallel Connection Behavior**

Analisa Koneksi yang terbuang saat melakukan koneksi secara bersamaan

3. **Key Results of Trace Analysis**

Dari penelitian yang dikembangkan dimana koneksi tunggal lebih baik dalam pemulihan pengriman data ketimbang dengan koneksi yang bersamaan

C. Pembahasan

Ada beberapa metode yand dapat dilakukan antara lain :

1. **Application-level Multiplexing**

Solusi tingkat aplikasi menghindari penggunaan beberapa koneksi TCP paralel, dan masalah yang dihasilkan, dengan multiplexing beberapa

aliran data ke koneksi TCP tunggal. Karena TCP hanya menyediakan abstraksi byte-stream tunggal

Tapi ada beberapa kekurangan antara lain :

a. Mereka membutuhkan tambahan dan diperlukan server dan klien yang baik.
b. Mereka tidak mengizinkan multiplexing transfer yang dilakukan oleh lebih dari satu aplikasi.
c. Multiplexing melalui koneksi TCP tunggal memperkenalkan kopling yang tidak diinginkan antara transfer data yang secara logis independen

2. TCP-INT Implementation

Menggambarkan penerapan kontrol kemacetan terintegrasi dan skema pemulihan kehilangan yang hanya mengubah TCP pada pengirim. Untuk setiap host yang terkait dengan satu mesin, tumpukan TCP / IP membuat struktur (Gambar 9) untuk menyimpan informasi tentang komunikasi apa pun. Struktur baru ini memungkinkan kontrol kemacetan bersama yang diinginkan dan pemulihan kerugian dengan menyediakan satu titik koordinasi untuk semua koneksi ke host tertentu. Struktur chost berisi variabel-variabel standar TCP yang terkait dengan pemeliharaan jendela kongesti TCP (cwnd, ssthresh dan count). Struktur ini juga memperkenalkan beberapa variabel baru untuk membantu dalam kontrol kemacetan (ownd, decr_ts) dan variabel lain untuk mendukung pemulihan kerugian terintegrasi (pkts []). Di subbagian berikut, kami menjelaskan bagaimana berbagai rutinitas TCP menggunakan dan memperbarui informasi baru ini. Rutin pengiriman data baru: Ketika suatu koneksi ingin mengirim suatu paket, ia memeriksa untuk melihat apakah jumlah byte yang sudah ada dalam "pipa" dari pengirim (ownd) lebih besar daripada ukuran "pipa" yang diinginkan (cwnd). Jika tidak, koneksi akan menyiapkan paket yang akan dikirim dengan menambahkan entri pada ekor daftar paket yang beredar. Entri ini berisi ukuran nomor urut dan stempel waktu dari paket yang dikirimkan. Ketika paket dikirim, koneksi menambah kepemilikan

dengan ukuran paket. Kami menggunakan penjadwalan round-robin di seluruh koneksi meskipun itu bukan persyaratan penting. Rutin recv ack baru: Ketika ack baru tiba, pengirim meningkatkan variabel cwnd yang sesuai. Juga pada saat kedatangan ACK baru, pengirim menghapus paket dari daftar pkts [] yang telah mencapai

3. **Integrated Congestion Control/Loss Recovery**

Motivasi untuk kontrol kemacetan terintegrasi dan pemulihan kerugian adalah untuk memungkinkan aplikasi menggunakan koneksi TCP terpisah untuk setiap transfer (seperti yang mereka lakukan hari ini), tetapi untuk menghindari masalah yang disebutkan dalam Bagian 3.2 dengan membuat modifikasi yang sesuai untuk tumpukan jaringan. Kami membagi fungsionalitas TCP ke dalam dua kategori: yang berkaitan dengan abstraksi TCP byte-stream yang andal, dan yang berkaitan dengan kontrol kemacetan dan pemulihan kerugian berbasis data. Yang terakhir ini dilakukan secara terintegrasi di set koneksi paralel. Kami menyebutnya versi modifikasi TCP TCP-Int. Dengan membuka n koneksi TCP terpisah untuk transfer, aplikasi memiliki n-stream stream yang andal dan independen untuk digunakan. Kontrol aliran untuk setiap koneksi terjadi secara independen dari yang lain, sehingga pengiriman data ke aplikasi penerima juga terjadi secara independen untuk setiap koneksi. Pada saat yang sama, kontrol kemacetan terintegrasi di seluruh koneksi TCP. Ada satu jendela kemacetan untuk set koneksi TCP antara klien dan server yang menentukan jumlah total data luar biasa yang dimiliki oleh set koneksi dalam jaringan. Ketika terjadi kerugian pada salah satu koneksi,

4. **Simulation Results: One Client Host Case**

Cara ini menjelaskan hasil dari simulasi ns yang dirancang untuk menguji kontrol kemacetan dengan pengenal integer dan pemulihan kehilangan pada koneksi TCP simultan. Tes pertama menggunakan topologi pada Gambar 7. Ukuran buffer router diatur ke 3 paket. Ini cukup kecil untuk memaksa transfer untuk memiliki jendela kemacetan kecil dan sering mengalami kerugian. Sekali lagi, topologi dan parameter dipilih untuk menciptakan kembali situasi yang sering terjadi di jejak kami, dan

tidak meniru jaringan yang sebenarnya. Dalam tes ini, node pengirim melakukan 4 transfer TCP ke penerima. Transfer mulai dari 0, 2, 4 dan 6 detik dan semua berakhir pada 10 detik. Pilihan aktual dari nilai 0, 2, 4, dan 6 tidak penting, hanya saja waktu mulai setiap koneksi terhuyung-huyung dalam waktu. Gambar 10 menunjukkan plot urutan untuk pengujian menggunakan pengirim berbasis SACK. Ini menunjukkan bahwa biasanya hanya satu koneksi berperforma memuaskan pada satu waktu. Misalnya, pada waktu 2 detik, koneksi yang memulai mengalami beberapa kerugian awal dan dipaksa untuk memulihkannya melalui timeout kasar. Faktanya, koneksi ini tidak mengirim sejumlah besar data hingga 4 detik kemudian (pada waktu 6 detik). Selama periode 10 detik, koneksi dimulai pada waktu 2 detik. dan waktu 6 detik. akun untuk fraksi sangat kecil (<10%) dari total byte yang ditransfer. Ketidakadilan dan kinerja yang tidak dapat diprediksi (karena timeout kasar) tidak diinginkan dari sudut pandang aplikasi karena koneksi yang membawa data penting dapat diperlambat sementara yang lain yang membawa data yang kurang penting lebih baik.

5. **Simulation Results: Multiple Client Hosts Case**

Simulasi ini menggunakan topologi jaringan dimana saat waktu mulai koneksi di waktu 0, transfer TCP tunggal dimulai setengah dari bandwith dapat diterima kembali. berfungsi untuk mengimplementasikan fungsi yang lebih komplex dalam limit bandwidth, dimana penggunaan packet mark nya memiliki fungsi yang lebih baik. Digunakan untuk membatasi satu arah koneksi saja baik itu download maupun upload. Secara umum Queue Tree ini tidak terlihat berbeda dari Simple Queue. Perbedaan yang bisa kita lihat langsung yaitu hanya dari sisi cara pakai atau penggunaannya saja. Dimana Queue Simple secara khusus memang dirancang untuk kemudahan konfigurasi sementara Queue Tree dirancang untuk melaksanakan tugas antrian yang lebih kompleks dan butuh pemahaman yang baik tentang aliran trafik

WHY INTERNET SLOWS DOWN WHEN ITS BUSY – COMPUTERPHILE

Magister Teknik Informatika Universitas Bina Darma

Nama : Ade saputra

Nim: 192420027

RINGKASAN VIDEO

ISP tidak selalu benar, mereka bertaruh bahwa semua pelanggan mereka tidak akan menggunakan semua bandwidth mereka sepanjang waktu. Sehingga ISP memberikan layanan kecepatan internet broadband -contohnya hingga 100 Mbps- kepada pelanggang dengan cara berbagi koneksi kepada pengguna lain. Hal ini mengakibatkan kecepatan internet menurun atau melambat apabila semakin banyak pengguna yang menggunakan layanan internet secara bersamaan pada jaringan yang sama.

Sebagai contoh ketika ada 10 pengguna internet dalam jaringan dengan kecepatan hingga 100 Mbps, mengakses layanan internet hingga 1 Gbps maka setiap pengguna maksimal hanya bisa mencapai kecepatan 10 Mbps. Jika dalam jaringan tersebut pada waktu sibuk pengguna layanan internetnya bertambah menjadi 50 maka kecepatan internet pun akan menurun dan maksimal yang akan di dapat hanya sampai 2 Mbps.

Ketika semakin banyaknya pengguna yang mengakses layanan internet untuk streaming, downloading maupun browsing, hal ini tidak hanya membuat kecepatan menjadi turun atau melambat tetapi dapat menyebabkan antrian data semakin lama dan akhirnya koneksi internet menjadi terputus.

Hal lain yang menyebabkan terjadinya penurunan kecepatan internet adalah data internet itu sendiri. Jika dibandingkan dengan telefon, data telefon hanyalah suara antara pengguna yang satu dengan pengguna yang lain. Sedangkan pada internet merupakan sekumpulan paket data yang dikirimkan dan didistribusikan langsung berupa paket data tersebut.

SARAN Gunakan layanan koneksi internet dedicated. Berbeda halnya dengan koneksi Broadband, koneksi Dedicated menawarkan kualitas layanan yang lebih

baik dengan jaminan yang signifikan. Berikut ini adalah sifat-sifat dari Internet Dedicated 1. Ratio 1:1 Pernahkah Anda menemui dimana hasil speedtest menunjukkan angka yang besar namun kenyataan koneksi Anda lambat? Hal ini dikarenakan bandwidth yang ditampilkan merupakan bandwidth ketika mengakses backbone ISP sehingga angka yang ditampilkan pada speedtest bukan sepenuhnya yang Anda dapatkan pada network. Berbeda dengan broadband yang menekankan penggunaan kata ‘hingga’, koneksi dedicated memberikan jaminan bahwa Anda akan selalu mendapatkan bandwidth sesuai dengan besar paket berlangganan Anda selama 24 jam penuh. 2. Bandwidth Simetris Dedicated internet menawarkan bandwidth simetris yang menjamin kecepatan download dan upload Anda setara dan sesuai dengan paket berlangganan Anda. Contohnya Anda berlangganan paket internet dedicated sebesar 10 Mbps, maka secara konstan Anda akan selalu mendapatkan kecepatan upload 10 Mbps dan donwload sebesar 10 Mbps. 3. Jaminan SLA

Pada Umumnya apabila Anda menggunakan koneksi dedicated, maka Anda akan mendapatkan jaminan SLA (Service Level Agreement) seperti kuota waktu down, latensi dan sebagainya yang semakin menjamin ketersediaan koneksi internet Anda. Kompensasi yang ditawarkan apabila SLA tidak dipenuhi dapat bervariasi seperti misalnya pengembalian sebagaian biaya langganan

Ade Saputra & Rudy Seftiawan
Nim : 192420027 & 192420029
Kelas : MTI

Jaringan PAKET CV CEMERLANG

Sistem Layanan Jaringan Profesional

Rencana Desain Warnet

Profil

Profil Perusahaan

VISI

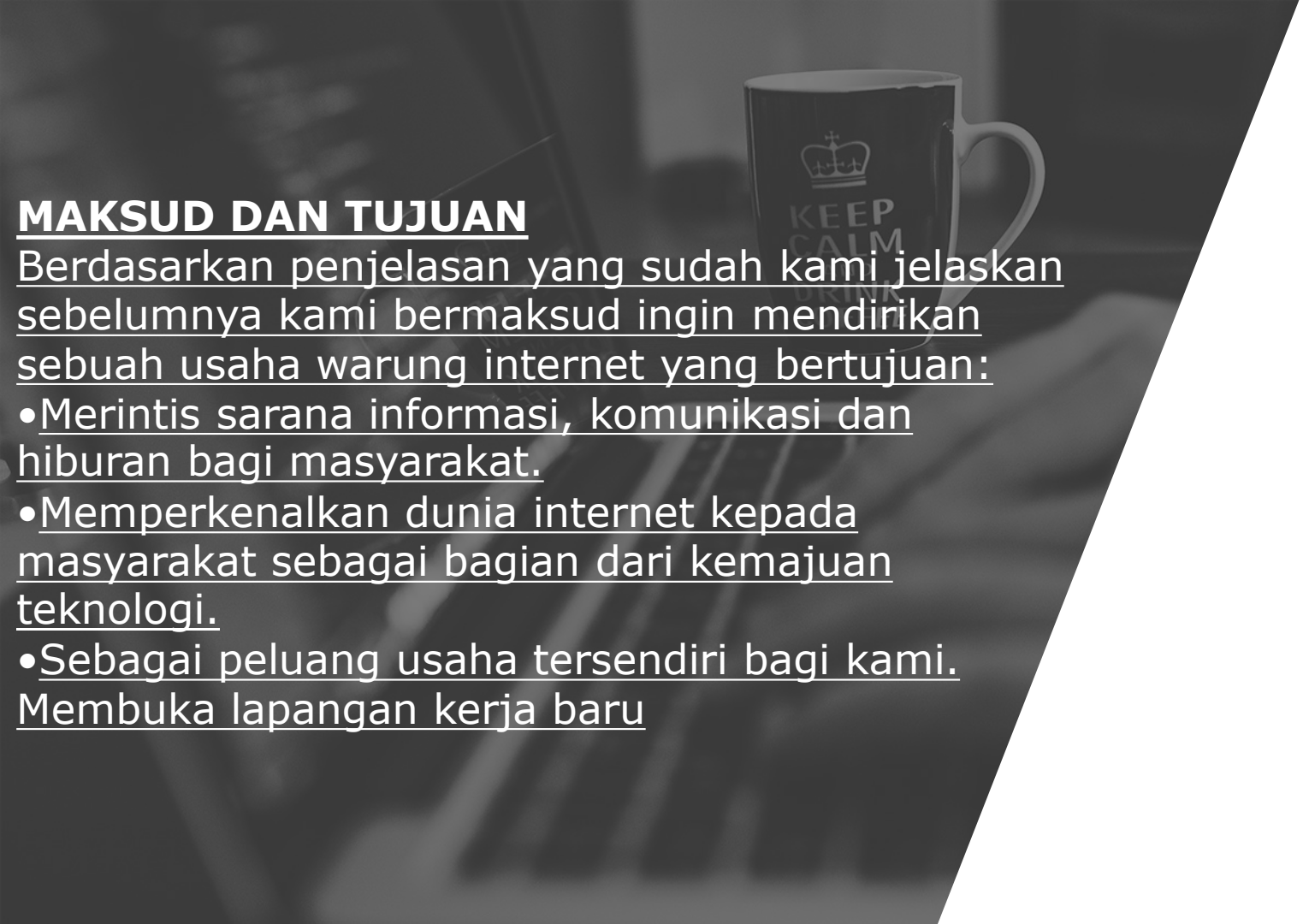
“Untuk menjadi perusahaan layanan teknologi yang terbaik, memberikan nilai b erkelanjutan kepada para pemangku kepentingan Proses Bisnis

MISI

- “Membangun hubungan mitra tepercaya melalui penyediaan solusi terbaik”
- “Menjalankan filosofi perusahaan kami dengan standar kualitas tinggi Sumber Daya Manusia, untuk memastikan pertumbuhan yang berkelanjutan dari perus ahaan kami”

CV CEMERLANG

Sistem Layanan Jarigan Nasional



MAKSUD DAN TUJUAN

Berdasarkan penjelasan yang sudah kami jelaskan sebelumnya kami bermaksud ingin mendirikan sebuah usaha warung internet yang bertujuan:

- Merintis sarana informasi, komunikasi dan hiburan bagi masyarakat.
- Memperkenalkan dunia internet kepada masyarakat sebagai bagian dari kemajuan teknologi.
- Sebagai peluang usaha tersendiri bagi kami. Membuka lapangan kerja baru

A. Latar Belakang

Dengan semakin meningkatnya kebutuhan masyarakat akan inf ormasi dan komunikasi, menuntut akan adanya kemudahan dal am mengaksesnya oleh karena itu kami berinisiatif untuk memb angun dan mengembangkan bisnis internet. Bisnis ini selalu dib utuhkan karena informasi yang dijualnya.

Sebagai sarana komunikasi, serta sebagai media hiburan. Apal agi dengan kehadiran jejaring social yang saat ini sangat di gan drungi masyarakat, kenyataan ini tentu akan meningkatkan par a pengguna internet Maka dari itu kami dari CV Cemerlang me nawar kan kepada an Bapak Usman di jl jaya 7 lematang 8.

.

Tujuan Penawaran Bisnis Warnet

Service Informasi
Kebutuhan User dalam mendapatkan Informasi Yang Cepat dan Hiburan

Pendidikan
Kebutuhan sumber Ilmu ter baru

Peluang Usaha
Sangat Profit dan
Di Butuh kan

Ekonomi
Mendapatkan profit
Cukup Besar

Service Yang Komplitt
Fasilatas Cepat dan Nyaman

Kepatuhan
Memblok situs situs yang berbahaya

Total Seluruh Biaya Modal Yang Digunakan

Server : Rp. 5.224.000,-
Client : Rp. 17.472.000,-
Jaringan : Rp. 2.155.000,-
Lainnya : Rp. 1.615.000,-
Total : Rp. 27.876.000,-

ANGGARAN BIAYA

BIAYA ALAT & SARANA UNTUK SERVER/Operator (Rakitan) 1 Server

Alat dan Bahan	Harga
Motherboard Ecs H61H2-MV HDMI + Gigabit Lan intel socket (1155)	Rp. 570.000,-
Processor Intel Core i3-3220 Processor (3M Ca che, 3.30 GHz) (1155)	Rp. 800.000,-
Ram Venomrx 8Gb Pc 1333 Venomrx DDR3	Rp. 566.000,-
HDD Sata Hitachi Sata III 1Tb 7200 Rpm 3.5I nch Int 3,5 I	Rp. 580.000,-
HDD SSD Intel 335 Series 80 Gb SSD	Rp. 898.000,-
Power Supply Dazumba DZ 500W Box	Rp. 540.000,-
Casing Mid Tower Dazumba D-Vito 683	Rp. 325.000,-
Speaker Simbadda 5100n Aktif	Rp. 450.000,-
UPS Prolink Pro 700v 650Va, True AVR, Wid e Input Voltag	Rp. 495.000,-
Total	Rp. 5.224.000,-

CV CEMERLANG

Sistem Layanan Jarigan Nasional

BIAYA ALAT & SARANA UNTUK CLIENT (Rakitan) 7 Unit Komputer

Alat dan Bahan	Harga
Motherboard Gigabyte GA-G41M-Combo (775)	Rp. 4.424.000,-
Processor Intel E5700 Dual Core (Tray + Fan) (3.0Ghz, C2 Mb, Fsb) (775)	Rp. 805.000,-
Ram Vgen 2Gb Pc 12800/1600 Vgen DDR3	Rp. 1.309.000,-
VGA Pixelview GT 440 2Gb 128bit DDR3 Nvidia PCI Exp.	Rp. 3.724.000,-
Power Supply Dazumba DZ 380W OEM	Rp. 910.000,-
Casing Mini Atx Dazumba DE-210	Rp. 1.155.000,-
Advance 15.6 Inch LM 1670 Lumine LED+Speaker	Rp. 3.850.000,-
Keyboard Logitech Classic Keyboard PS/2 K100 Keyboard	Rp. 581.000,-
Mouse Logitech B100 Optical USB Black Mouse	Rp. 343.000,-
Mouse Pad Oem Mouse Pad Standard/Bantal Mouse Pad	Rp. 56.000,-
Headset Komic Hed103	Rp. 315.000,-
Total	Rp. 17.472.000,-

CV CEMERLANG

BIAYA ALAT & SARANA UNTUK JARINGAN WARNET

Alat dan Bahan	**Harga**
NIC Gigabit (Lan Card) Untuk PC client dan server (11 Unit)	Rp. 1.320.000,-
Router MikroTik Seri RB951-2n	Rp. 530.000,-
TP-Link TL-SG1016D : 16 Port Gigabyte	Rp. 870.000,-
Kabel UTP STURDY (100m)	Rp. 110.000,-
RG-45 (100 Unit)	Rp. 45.000,-
Total	Rp. 2.875.000,-

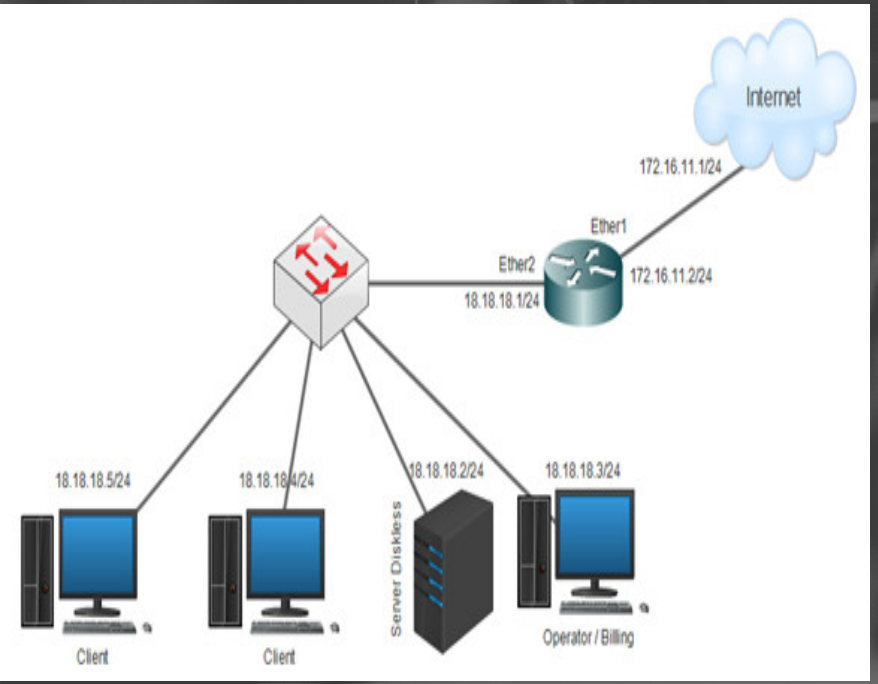
BIAYA ALAT & SARANA LAINNYA (KEPERLUAN WARNET)

Alat dan Bahan	Harga
Meja Bersekat (7 unit)	Rp. 700.000,-
Meja Operator	Rp. 225.000,-
Karpet Biru Standard Ukuran 2x3m (2 Unit)	Rp. 100.000,-
Kipas angin GMC Wall Fan 509 (3 Unit)	Rp. 360.000,-
Kenmaster Stop Kontak F1-44 Switch (4 lubang) (6 Unit)	Rp. 180.000,-
Crimping Tools	Rp. 50.000,-
Total	Rp. 1.615.000,-

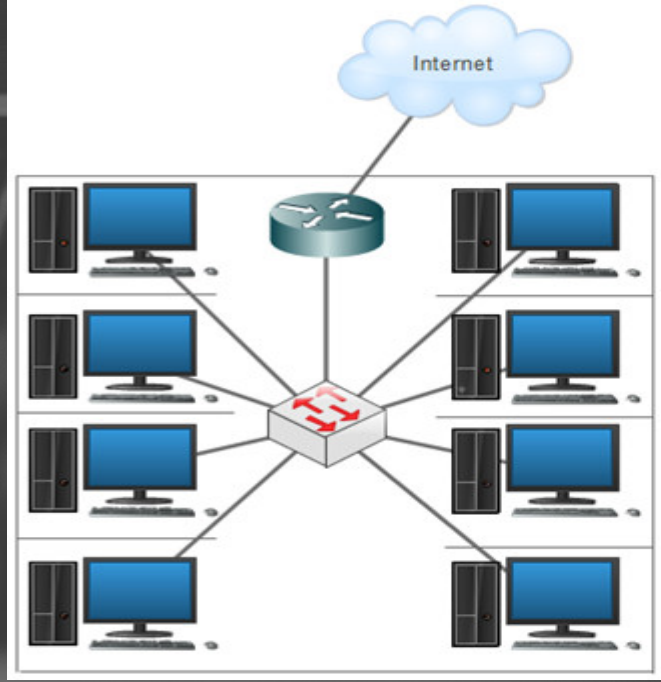
CV CEMERLANG

Desaing Jaringan Yang Di Tawarkan

Konfigurasi IP Warnet – Server, Operator dan Client



Topologi Warnet



IP Publik : 172.16.11.0/24
IP Lokal : 18.18.18.0/24

HARGA RENTAL WARNET

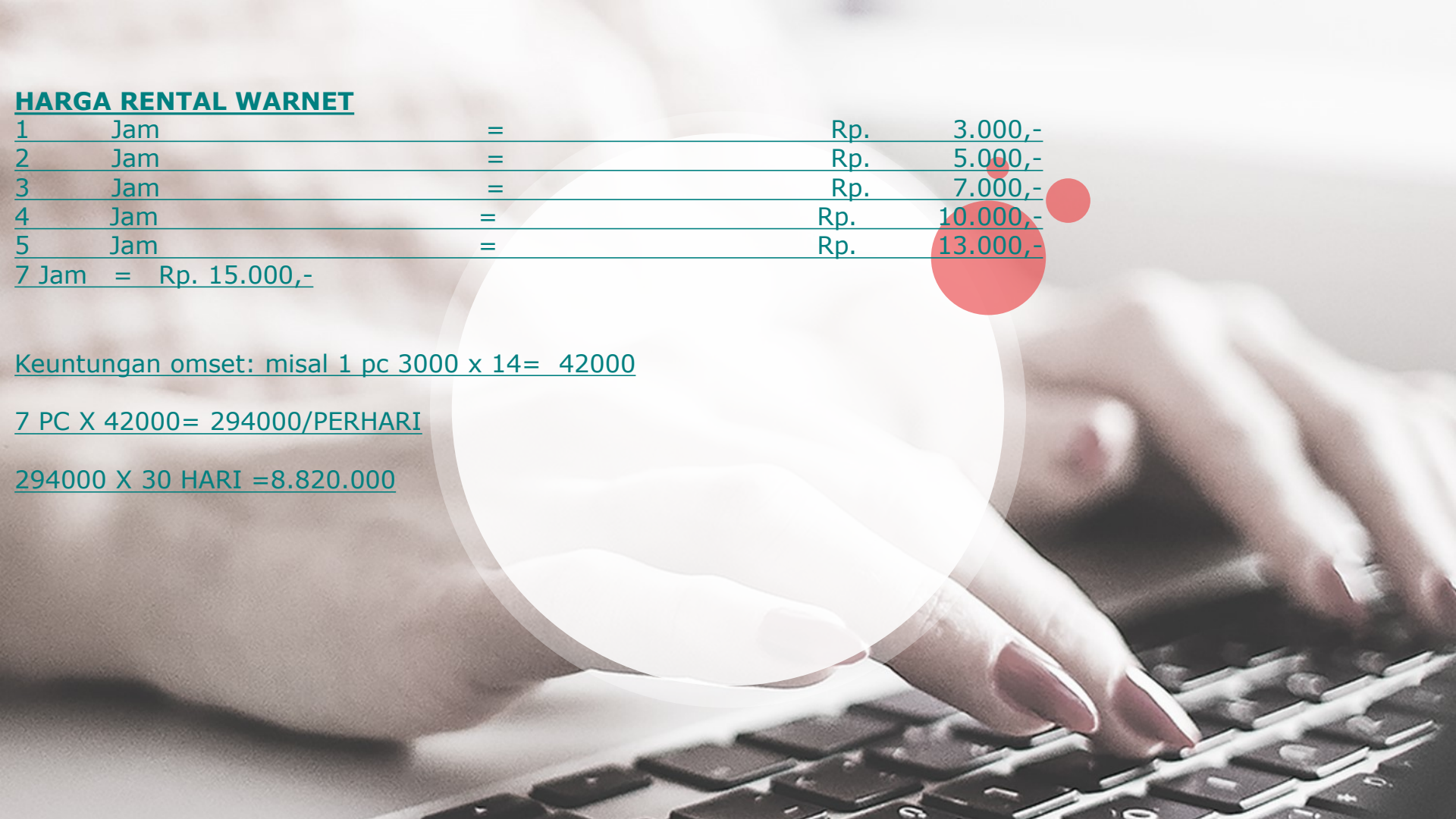
1	Jam	=	Rp.	3.000,-
2	Jam	=	Rp.	5.000,-
3	Jam	=	Rp.	7.000,-
4	Jam	=	Rp.	10.000,-
5	Jam	=	Rp.	13.000,-

7 Jam = Rp. 15.000,-

Keuntungan omset: misal 1 pc 3000 x 14= 42000

7 PC X 42000= 294000/PERHARI

294000 X 30 HARI =8.820.000




Thank you

Bhagaskara 192420028
Computer Network and Communication

Metode yang dapat saya usulkan untuk menyelesaikan case “Why Internet Slows Down When its Busy” tersebut adalah sebagai berikut:

1. **Queue Tree & PCQ**

   Queue Tree berfungsi untuk mengimplementasikan fungsi yang lebih komplex dalam limit bandwidth, dimana penggunaan packet mark nya memiliki fungsi yang lebih baik. Digunakan untuk membatasi satu arah koneksi saja baik itu download maupun upload. Secara umum Queue Tree ini tidak terlihat berbeda dari Simple Queue. Perbedaan yang bisa kita lihat langsung yaitu hanya dari sisi cara pakai atau penggunaannya saja. Dimana Queue Simple secara khusus memang dirancang untuk kemudahan konfigurasi sementara Queue Tree dirancang untuk melaksanakan tugas antrian yang lebih kompleks dan butuh pemahaman yang baik tentang aliran trafik.

   PCQ (Per Connection Queuing) Digunakan untuk mengenali arah arus dan digunakan karena dapat membagai bandwidth secara adil, merata dan masif. PCQ digunakan bersamaan dengan fitur Queue, baik Simple Queue maupun Queue Tree.

   PCQ Classifier berfungsi mengklasifikasikan arah koneksi, Misalnya jika Classifier yang digunakan adalah src-address pada Local interface, maka aliran pcq akan menjadi koneksi upload. Begitu juga dgn dst-address akan menjadi pcq download.

   PCQ rate berfungsi untuk membatasi bandwidth maksimum yang bisa didapatkan. Dengan memasukkan angka pada rate ini (default: 0) maka maksimal download yang akan didapatkan per IP akan dibatasi mis. 128k (kbps).

2. **LVS Linux Virtual Server dengan algoritma Rond robin**

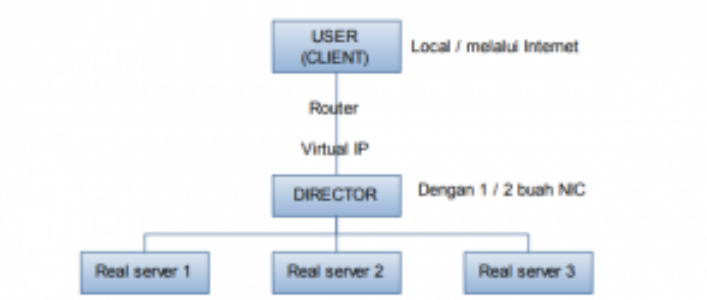
   Selain itu dengan load balancer kita dapat mengarahkan client ke web server yang lain jika terjadi web server overload atau web server sedang down, hal ini sangat bermanfaat, baik dari sisi kecepatan akses maupun efisiensi waktu pada saat user mengakses sebuah web. Untuk menganalisis algoritma scheduling di linux virtual server (LVS), dilakukan perancangan, implementasi dan pengukuran waktu respon.

   Pada rancangan model sistem LVS (Linux Virtual server) dilakukan pengujian dengan tahap sebagai berikut:

   1. Client melakukan requests ke server melalui load balancer dengan tool httperf dengan jumlah koneksi user antara 10.000/s – 50.000/s.
   2. Komputer load balancer menerima requests dari client kemudian melakukan scheduling dan rewriting packets. Pada proses ini load balancer mengirimkan request ke real server yang sedang aktif dengan menggunakan algoritma round robin.
   3. Real server menerima request dari client melalui komputer load balancer dan mengirimkan reply ke load balancer.

4. Komputer load balancer menerima reply dari real server yang sedang aktif, kemudian meneruskan reply ke komputer client yang meminta request. Client menerima replies dari load balancer. Pada proses ini komputer client dapat mengetahui respon time dari sebuah web server.

Linux Virtual Server atau disingkat LVS merupakan suatu teknologi clustering yang dapat digunakan untuk membangun suatu server dengan menggunakan kumpulan dari beberapa buah realserver. LVS merupakan implementasi dari komputer cluster dengan metoda High Availibility. LVS mengimbangi berbagai bentuk dari service jaringan pada banyak mesin dengan memanipulasi paket sebagaimana diproses TCP/IP stack. Satu dari banyak peran yang paling umum dari Linux Virtual Server adalah bertindak sebagai server yang berada pada garis terdepan dari kelompok server web. Seperti ditunjukkan pada Gambar 1 Linux Virtual Server atau LVS ini terdiri dari sebuah Director dan beberapa realserver yang bekerja bersama dan memberikan servis terhadap permintaan user. Permintaan User diterima oleh Director yang seolah olah berfungsi sebagai IP Router yang akan meneruskan paket permintaan user tersebut pada real server yang siap memberikan servis yang diminta. Dengan demikian virtual server akan terdiri dari beberapa komputer Yang mempunyai image yang sama tetapi ditempatkan pada IP yang berbeda. User dapat mengakses virtual server tersebut dengan bantuan suatu Director, yang bertugas untuk melakukan pemetaan IP dari server dan komputer lainnya yang berperan sebagai virtual server.



LVS Director adalah modifikasi dari sistem Linux yang bertanggung jawab untuk mendistribusikan permintaan user/client terhadap realserver pada kelompok server. Realserver melakukan pekerjaan untuk memenuhi permintaan serta memberikan atau membuat laporan balik kepada user/client. LVS Director memelihara rekaman daripada sejumlah permintaan yang telah ditangani oleh masing-masing realserver dan menggunakan informasi ini ketika memutuskan server mana yang akan ditugaskan untuk menangani suatu permintaan berikutnya. LVS juga dapat memiliki Director cadangan yang akan menggantikan bilamana suatu saat Director utama mengalami suatu kegagalan sehingga membentuk suatu LVS failover. Masing-masing realserver dapat bekerja dengan menggunakan berbagai sistem operasi dan aplikasi yang mendukung TCP/IP dan Ethernet. Pembatasan dan pemilihan sistem operasi pada realserver serta jenis servis yang didukung oleh realserver dilakukan pada saat proses konfigurasi LVS dijalankan.

ALGORITMA PENJADWALAN (SCHEDULING) ROUND ROBIN Mekanisme penjadwalan pada LVS dikerjakan oleh sebuah patch kernel yang disebut modul IP Virtual Server atau IPVS modules. Modul ini mengaktifkan layer 4 yaitu transport layer switching yang dirancang dapat bekerja dengan baik pada multi server dalam IP address tunggal (virtual IP address). IPVS membuat IPVS table pada kernel untuk

menelusuri dan merutekan paket ke real server secara efisien. Tabel ini digunakan oleh load balancer yang sedang aktif (yang pasif adalah backup-nya) untuk meneruskan client request dari virtual IP address ke real server. IPVS table secara rutin diperbarui menggunakan software ipvsadm.

Nama : Daniel Kukuh Pribadi

NIM : 192420024

Kelompok : MTIB

Analisa:

Dari Video dapat dianalisa bahwa ISP yang digunakkan memakai teknik Multiplexing dalam hal pendistribusian bandwith kepada pelangan. Maksudnya adalah, penyedia layanan ISP mengasumsikan bahwa setiap pengguna jasa layanannya tidak akan menggunakan secara penuh semua bandwith yang diperuntukan untuknya, maka dari itu penyedia layanan akan memakai "sisa" bandwith terebut untuk dibagi dan digunakan oleh pelanggan yang lain yang menggunakan bandwith yang lebih tinggi pada waktu yang bersamaan. Penggunaan teknik multiplexing tersebut, merupakan hasil analisa dari penyedia layanan ISP mengenai pola pengguan jasa layanan internet dari pelanggan, yang tentunya tidak seratus persen akurat. Hal ini dilakukan tentunya dengan motif ekonomi, yaitu menjaring semakin banyak pelanggan dari “sisa” bandwith yang tidak digunakan oleh pelanggan yang telah ada. Dampaknya tentu saja, pada penuruna kwalitas kecepatan internet itu sendiri, yang diakibatkan oleh keterbatasan bandwidths karena hilangnya “bandwidth tolerence” yang diambil oleh penyedia jasa layanan ISP.

Issue:

Permasalahan yang timbul dari “pencurian” bandwidth oleh penyedia jasa layana ISP tersebut, tentunya timbulnya kelambatan internet pada pelanggan-pelanggan yang memerlukan bandwidth besar, seperti pada penggunaan korporasi yang membpunyai aplikasi web, vpn, ftp server, dan lain sebagainya. Masalah akan jelas terlihat ketika beban pada klien tersebut pada load maksimal, tentunya layanan internet akan mengalai penurunan kecepatan atau delay.

Pada gambarannya, mungkin teknik multiplexing yang digunakan terkesan memberikan keuntungna jumlah pelanggan pada penyedia jasa layanan ISP. Namun sebenarnya merupakna bom waktu yang diaktifkan oleh ketidakpuasan pelanggan pada model pelayanan seperti ini. Dampak langsung kepada konsumen berupa penurunan kecepatan internet, tentunya akan mengurangi tingkat kepuasan pengguna, sehingga berakibat hengkangnya konsumen kepada competitor lain.

Solusi:

Dengan berkembangnya internet, maka penyedia jasa layanan ISP pun mulai banyak. Dengan paket penawaran masing-masing dan juga track record dari masing-masing perusahaan tersebut, maka pemilihan penyedia jasa layanan ISP menjadi krusial. Perusahaan dengan kepercayaan public yang besar, cukup layak untuk dipertimbangkan sebagai mitra. Pun jika menginginkan suatu jaminan yang lebh konkrit, memilih untuk memiliki 2 atau lebih mitra penyedia layanan ISP adalah hal yang baik untuk dilakukan sebagai backup plan dalam kondisi penting.

Paparan Study Case Network Topology

Palembang, 20 Desember 2019

By. Hendra Yada Putra & Ichan Syaputra (MTI-Reg B Angk 21)

Agenda

- Pendahuluan
- Topologi exisiting
- Assessment
- Rekomendasi
- Budget

Sekenario

1. Perusahaan Jasa Pengantaran Barang sejak 2012
2. Memiliki 1 Kantor pusat dan 12 Kantor cabang
3. Jumlah pegawai 300 orang
4. Bisnis menggunakan Channel Kantor dan internet
5. Mengadopsi teknologi Digital kedalam bisnis perusahaan pada tahun 2016, transaksi perusahaan meningkat pesat hampir 500%.
6. Peningkatan transaksi didominasi pada Channel Internet, dimana transaksi melalui kantor menurun tidak lebih dari 20% terhadap total transaksi.
7. Peningkatan transaksi diiringi dengan peningkatan keluhan dari customer dan internal staff.
    1. apikasi sering dikeluhkan lambat,
    2. SLA Layanan setiap bulannya tidak tercapai
    3. Virus dan Malware
8. Telah melakukan Tunning terhadap Aplikasi, Server, dan internet, namun keluhan masih tinggi.
9. Dari sisi Compliant. perusahaan juga diharuskan untuk mengikuti standar ISO27001.
10. Manajemen meminta untuk diadakan assessment dari sisi Jaringan

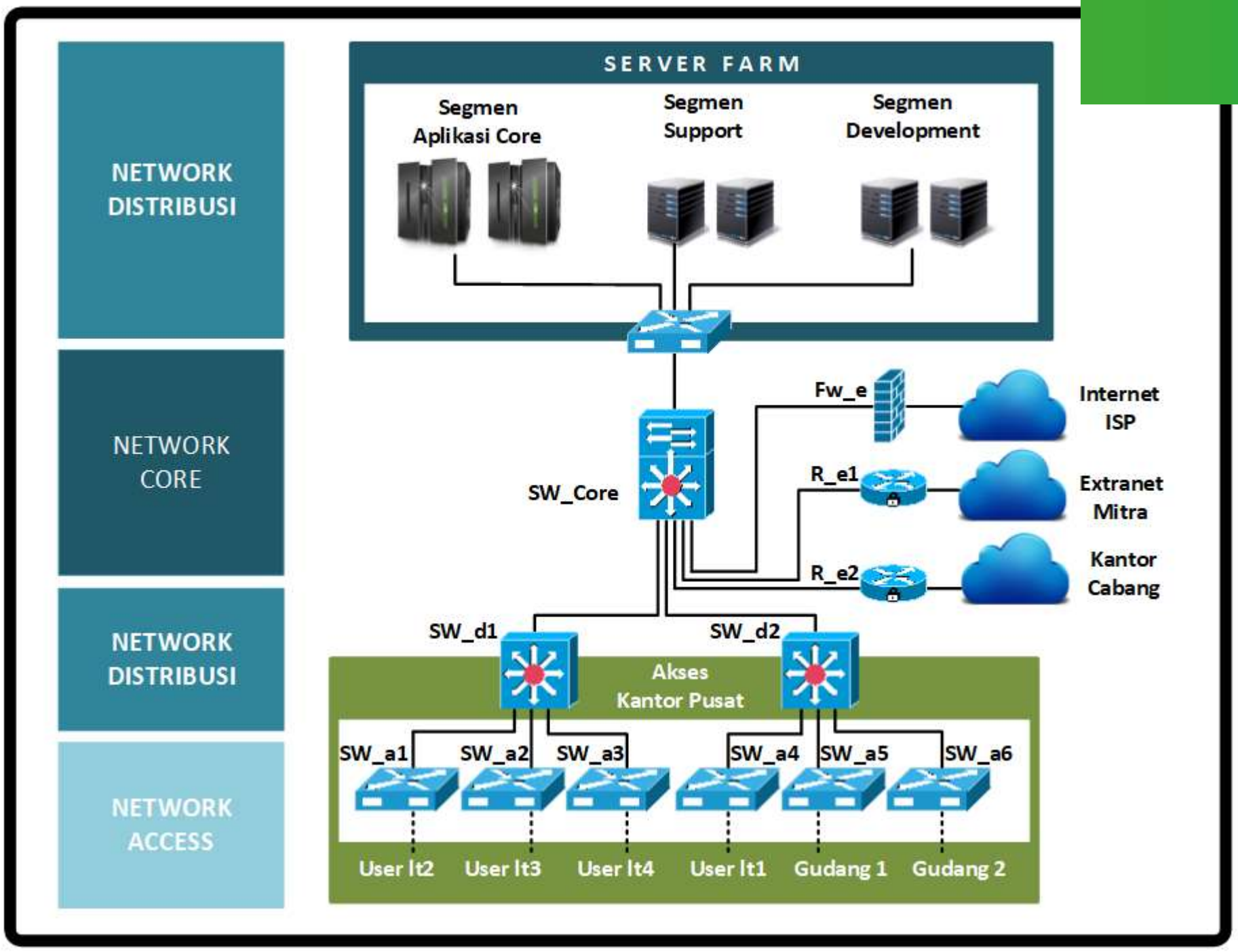
Latar Belakang

"Pemenuhan Tugas Mata Kuliah Computer Network and Communication"

Tujuan

"Memaparkan Rekomendasi Topologi Network beserta Perangkat pendukungnya untuk Perusahaan kelas Menengah Keatas, dengan Konsentrasi Peningkatan Availbaility, Performance dan keamamanan"

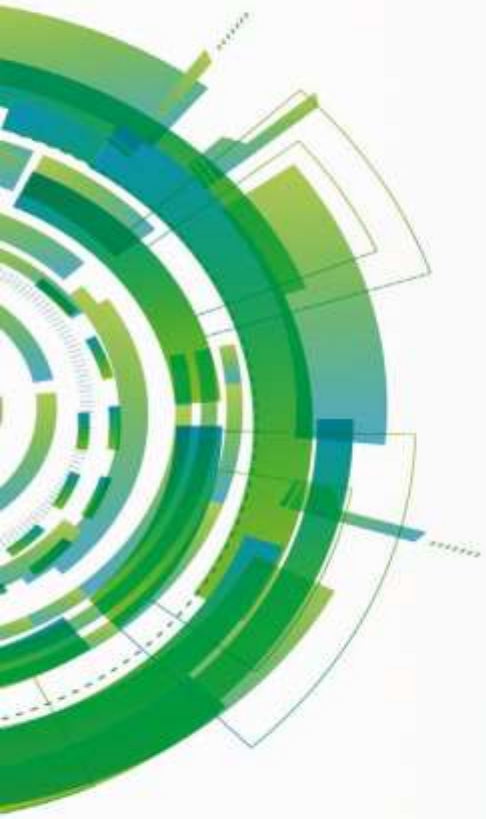
TOPOLOGI



Perangkat terdiri dari:

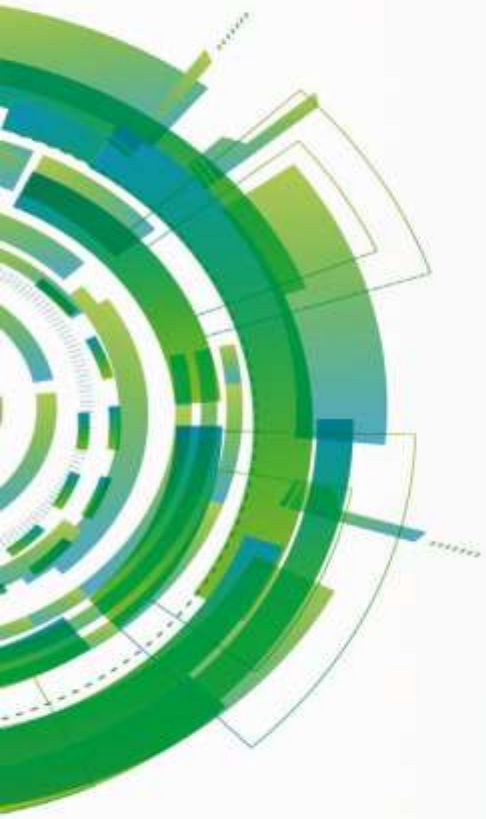
1. 1 Swich Core
2. 7 Switch Access
3. 2 Switch Distribusi
4. 2 Router
5. 1 Firewall Edge

Pengumpulan data



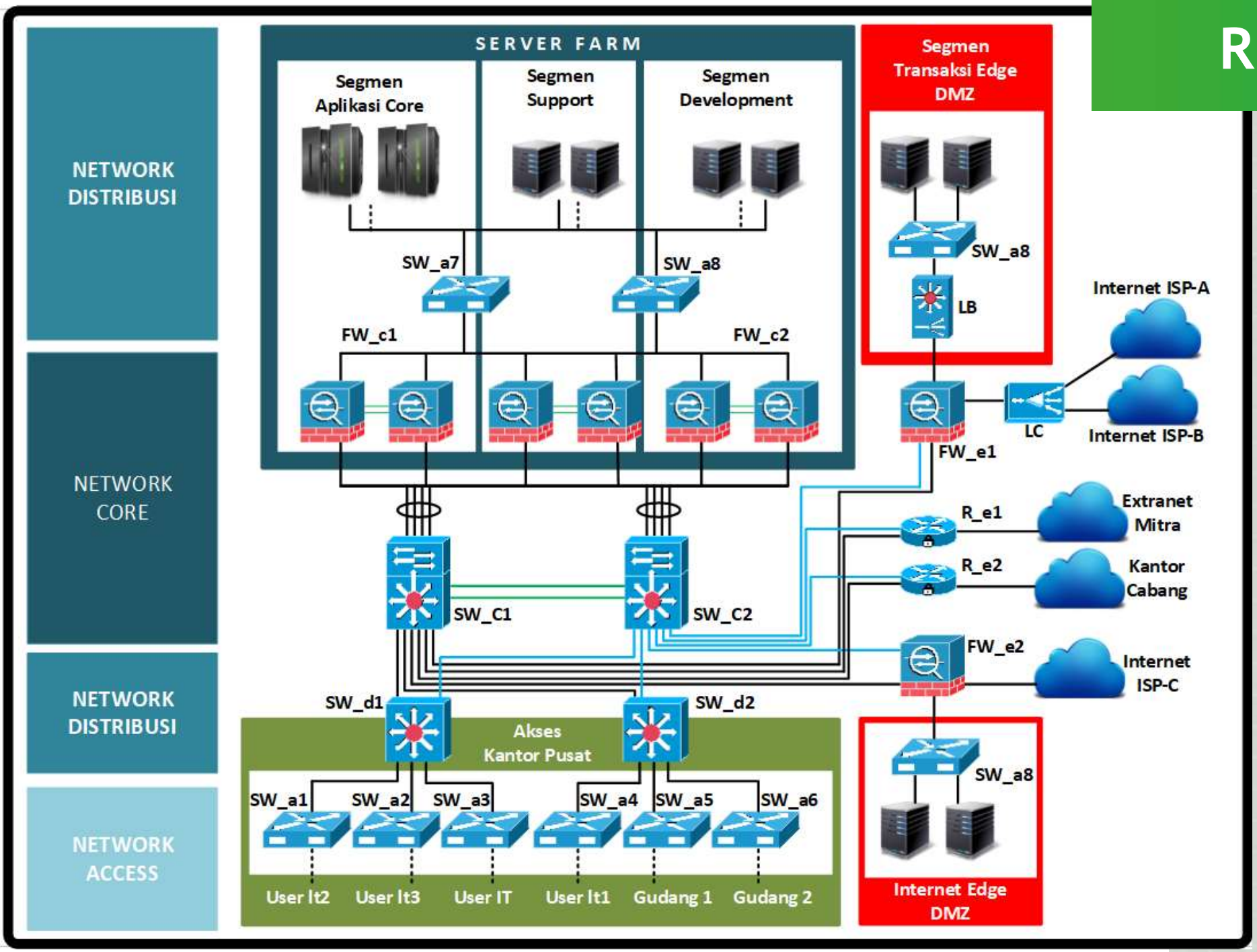
- Load trafik internet selalu mencapai level 80-90% di jam operasional kantor.
- Load trafik pada Switch Core yang menuju Server farm tidak lebih dari 70% dengan teknologi ethernet yang digunakan 1GbE
- Peak Cpu usage pada Switch Core 70%
- Pemisahan fungsi pada server Core untuk aplikasi, Database dan Web dengan SSL Sertifikat ditanam pada Web Server
- Serverfarm berada dalam 1 segemen untuk semua server: Server Core, Server Mobile API, Server Gateway, Development, AD, DNS-NTP, mail, Servicedesk, Antivirus, Monitoring, HR.
- Terdapat Cold Backup untuk server Core.
- Firewall menggunakan teknologi sebelum Next Generation
- Terdapat Perangkat yang telah EOL yaitu Swicth Core, Firewall, Switch Distribusi, Router, dan beberapa Switch Access

Hasil analisa



- Harus dibedakan akses Internet untuk akses transaksi dan User
- Segera dilakukan penggantian perangkat EOL (risiko tinggi)
- Dibutuhkan Penambahan Backup yang tergolong Critical poin
- Perlu Menambahkan Link Internet baru dengan mengimplementasikan perangkat Link Controller untuk Redudansi Internet dan penempatan SSL Sertifikat
- Pelu Menempatkan Server yang membutuhkan koneksi internet ke area DMZ sesuai area jaringan internet
  - Server support (Antivirus, DNS-NTP, Mail, Server Gateway)
  - Server Web Core dan Server Mobile API
- Pemisahan Segement pada Serverfarm dan menambahkan NGT Firewall untuk proteksi masing-masing segmen
- Dibutuhkan pembagi Beban untuk WEB Core Backup
- Perlu penggantian firewall Internet dengan NGT Firewall.

REKOMENDASI



Perangkat terdiri dari:

1. 2 Swich Core
2. 8 Switch Access
3. 4 Switch Distribusi
4. 2 Router
5. 1 Load Balancer
6. 1 Load Controller
7. 2 Firewall Core
8. 2 Firewall Edge

Segmentasi

DETIL

- Segmen Aplikasi Core
  - Server Aplikasi Core Bisnis
  - Server Database
  - Server Interface
- Segmen Support
  - Server Active Directory
  - Server DNS internal
  - Server Aplikasi HR
  - Server Aplikasi Helpdesk
  - Server Email
  - Server Monitoring & Manajement
- Segmen Development
  - Server Development Core
  - Server Development Interface
  - Server Development Support

- Segemen Transaksi Edge
  - Server Web Korporasi
  - Server API Mobile
  - Server Web Transaksi
- Segemen Internet Edge
  - Server DNS Forwader
  - Server Antivirus
  - Server NTP
  - Akses Internet User

KEUNGGULAN

- Segmentasi lebih dalam untuk area server Farm sehingga keamanan lebih terjamin
- Kualitas Layanan dapat ditinggkatkan
- Pemisahan Jalur internet user dengan internet untuk transaksi
- Pengamanan berlapis dengan pemenuhan standard iso27001
- Availability layanan dengan link controller dan link balancer
- Dengan memisahkan akses SSL sertifikat pada server dapat meningkatkan performace Server

No	Item	Jumlah	Harga Satuan	Harga
1	Switch Core Cisco Catalyst 9300 C9300-48P-E	2	250.000.000	500.000.000
2	Switch Distribusi Cisco Catalyst 3850 Ws-C3850-24t-S	4	80.000.000	320.000.000
3	Switch Access Cisco Catalyst 2960-x (WS-C2960X-48FPD-L)	4	50.000.000	200.000.000
4	Router Cisco Cisco ASR 1002-X	2	70.000.000	140.000.000
5	Firewall Core Fortigate 1000D	1	260.000.000	260.000.000
6	Firewall Edge Fortigate 100e	1	80.000.000	80.000.000
7	Load Balancer A10 Thunder 840 ADC	1	150.000.000	150.000.000
8	Link Controller f5 2200 LTM	1	250.000.000	250.000.000
9	Cabling,Soket, dll (Panduit)	1	20.000.000	20.000.000
10	Implementasi	20	5.000.000	100.000.000
	TOTAL			2.020.000.000

Q & A


Thank you

Nama:Hendra Yada Putra
Kelas:MTI Reg B angkatan 21

Upload UAS MITB
Video title : Why Internet Slows Down When it's Busy

Analisa:
Dari Video tersebut pembicara menjelaskan mengenai akses internet yang diberlakukan oleh ISP ke customer, dimana secara teknis akses yang diberikan tidak murni sebesar bandwith yang ditawarkan untuk kondisi jika semua customer menggunakan full bandwith secara bersamaan pada ISP tersebut. Hal ini disebabkan karena ISP menggunakan teknik Multiplexing dalam mendistribusi bandwith ke customer, dimana ISP sendiri meyakini bahwa tidak semua customer menggunakan full bandwith diwaktu bersamaan, sehingga bandwith yang tidak digunakan/sedikit digunakan oleh beberapa customer akan dishare ke customer lain yang mengunakan lebih tinggi diwaktu bersamaan.

Saya menganalisa bahwa teknik ini digunakan ISP dengan memanfaatkan link internet yang dimilikinya untuk memperoleh jumlah customer yang lebih banyak, jika dibandingkan ISP mendisribusikan link nya secara utuh/real ke customer. Sehingga ini akan berdampak pada keuntungan bisnis ISP itu sendiri.

Meskipun ISP sendiri dalam melakukan teknik multilexing berdasarkan penelitian terhadap pola penggunaan/pattern internet oleh customer (statistic multiplexing), tentunya peluang terjadinya internet lambat pasti terjadi, ini tentunya harus menjadi perhatian bagi ISP.

Issue berkembang dari Analisa:
Service dari internet ISP yang memberlakukan multiplexing pada bandwith customer bedasarkan statistical multiplexing jika diperhatikan lebih dalam lagi akan sangat berdampak sekali terutama jika kita sebagai customer bisnis yang menggunakan internet sebagai sumber layanan/server (sebagai aplikasi web https, api-rest, ftp server, dan lainnya) untuk client-client kita, dimana jika link internet ISP tersebut dalam posisi load tinggi, maka sudah dipastikan client-client kita akan mengalami delay, bahkan drop koneksi dan hal ini tentunya dapat merugikan, karena berdampak pada availabilty layanan, serta tidak menutup kemungkinan akan timbul banyak komplain yang berdampak pada bisnis itu sendiri.

Solusi terhadap Issue:
Dari Issue yang berkembang diatas, saya mengusulkan untuk tidak menggunakan hanya satu link internet dari satu ISP saja, melainkan menggunakan beberapa link dari ISP yang berbeda untuk layanan aplikasi kita terhadap client kita. Dimana saat ini harga link internet publik semakin murah dan ini bertujuan untuk memastikan availability dari layanan dari aplikasi kita terhadap issue sebagai disampaikan diatas tidak terjadi.

Secara best practice multi link ISP ini sendiri akan dimanajemen mengunakan teknologi pembagi jalur jaringan berupa perangkat Link Controller, dimana teknologi pada perangkat ini akan

Nama:Hendra Yada Putra
Kelas:MTI Reg B angkatan 21

Upload UAS MITB
Video title : Why Internet Slows Down When it's Busy

memanajemen Link-link ISP yang kita gunakan tersebut dan melakukan re-route koneksi untuk client berdasarkan tipe dan kualitas link pada saat client akses ke server, dengan demikian jika saat salah satu link ISP mengalami penurunan kuliatas, atau terputus, atau pun load jaringannya tinggi, maka client kita akan diarakan ke link ISP yang lain yang kualiatasnya lebih baik, sehingga client tidak akan merasakan adanya hambatan saat mengakses layanan aplikasi kita.