**11. Маршрутни алгоритми Обхождане, наводняване,метод на Берън. Маршрутни алгоритми:** Основната функция на мрежовото ниво е дамаршрутизира пакетите от източника към получателя- през няколко хопа.**Маршрутен алгоритъм** е част от софтуера на мрежовото ниво, която определя по коя от изходните линии да се изпрати пристигнал пакет. За целта всеки маршрутизатор притежава **маршрутна таблица**.Ако мрежата е с пакетна комутация (дейтаграми),

решението трябва да се взима наново за всяка пристигнал пакет, тъй като оптималният маршрут може да се е променил.Ако се използва виртуален канал, решенията по маршрутизацията се взимат при създаването му.Маршрутизиращи протоколи :

Маршрутизиращите протоколи трябва даотговарят на множество изисквания.Да са достатъчно прости и лесни за конфигуриране и да осигуряват надеждна и стабилна работа на мрежата.Да реагират своевременно наотпадане на маршрутизатори или връзки между тях.

Да бъдат в състояние да открият алтернативни пътища за доставяне на пакетите, ако такива съществуват.Маршрутизиращи протоколи

Две други цели на маршрутизиращите протоколи си противоречат (на пръв поглед):- минимизиране на времето за закъснение (по-

малък престой на пакетите в междинните възли); - максимизиране на общия поток предполага буферите в маршрутизаторите да работят на максимален капацитет.Освен това максимизирането на общия поток може да влезе в противоречие с изискването

мрежовите ресурси да могат да се използват от всички потребители в мрежата.Маршрутизиращи алгоритмиМаршрутизиращите алгоритми са два вида -**неадаптивни** и **адаптивни**.При неадаптивните маршрутизацията не се извършва на базата на текущата топология на мрежата.Маршрутите между всеки два възела в мрежата се изчисляват предварително и се записват ръчно от мрежовите администратори, след което влизат в маршрутните таблици.При промяна на топологията на мрежата (например при

отпадане на възел или на връзка), администраторите ръчно трябва да променят маршрутите.Това прави неадаптивните алгоритми приложими само в малки мрежи, при които рядко настъпват промени. Неадаптивните алгоритми се наричат още **статични**.

Попълване на маршрутна таблица.Пример.Мрежата представяме като граф - върховете са възлите в мрежата, а дъгите са

комуникационните линии.Метриката в графа се определя на базата на разстоянието до крайната точка (хопове),време-закъснението за минаване на един пакет,надеждност на линията, цена, брой хопове и др. Възможно е да се комбинират една или повече от изброените характеристики.Попълване на маршрутна таблица. Хопове.Метриката се определя от това през колко маршрутизатора (хопа) ще мине пакета до крайната точка (мрежа).Адаптивни алгоритми. Скорост на сходимост (конвергенция) При адаптивните алгоритми маршрутните таблици се променят динамично, за да отразяват промени в топологията и натовареността на трафика.Важна характеристика на адаптивния алгоритъм е неговата **скорост на сходимост**:времето, което е необходимо да се преизчислят маршрутните таблици на всички маршрутизатори в мрежата при промяна в топологията или трафика. (конвергенция)Адаптивни алгоритми. Принципза оптималност Оптималните пътища между всеки два възела в мрежата се изчисляват по някои от алгоритмите за намиране на най-къс път в граф (след като е въведена метрика в графа,представящ мрежата).Всички тези алгоритми се базират на **принципа за оптималност**:всяка част от оптимален път е също оптимален път между съответните два върха.Като следствие от този принцип, оптималните пътища от един връх към всички останали образуват дърво (sink tree).Алгоритъм на Дейкстра Алгоритъмът на Дейкстра е алгоритъм за намиране на най-къс път в граф от даден връх до всички останали върхове. Важно е да се отбележи, че при

алгоритъма на Дейкстра теглата на ребрата трябва да са положителни.Резултатът от алгоритъма е дърво на оптималните пътища от дадения връх до всички останали. Т.е Shortest Path Tree (SPF).Метод на наводняването Методът на наводняването (flooding) е статичен

алгоритъм за маршрутизация.Когато един пакет пристигне по дадена линия,той се изпраща по всички останали линии.Получават се дубликати на пакетите (тъй като между два възела обикновено има повече от един път).За да се избегне този проблем, се въвежда

идентификация на пакетите - всеки пакет съдържа идентификация на възела-източник и пореден номер.Метод на наводняването

Всеки маршрутизатор поддържа по един списък за всеки възел-източник, съдържащ номерата на пакетите, които са минали през него.

Ако пристигнал пакет присъства в списъка, той не се препредава, в противен случай списъкът се актуализира и пакетът се предава.

Списъците може да нарастнат много - при една голяма мрежа алгоритъмът не е приложим.Метод на наводняването За намаляване на дължината на списъка може да се добави допълнителен брояч *k*, който указва, че всички пакети до номер *k* са предадени. Пакетите с номера по-малки от *k* може да не присъстват в съответния списък.Метод на случайното обхождане Това е неадаптивен алгоритъм.Когато един пакет пристигне на някоя линия, той се изпраща по случаен начин по една от останалите линии.Този метод няма сигурна сходимост.Ако всеки възел помни историята си, т.е. кой пакет в коя посока е изпратил, то може да се предотврати повторно изпращане на един и същи пакет в една и съща посока, което ще доведе до сходимост.Такава информация, обаче, ще има твърде голям обем.Изолирана маршрутизация.Алгоритъм на Берън При изолираната маршрутизация всеки възел се разглежда сам за себе си.За всяка линия възелът поддържа по една изходяща опашка.При алгоритъма на Берън всеки пристигнал пакет се добавя към най-късата изходяща опашка в момента на пристигането.Този алгоритъм се адаптира към пиковете на предаване, но маршрутите не са оптимални.

**12. Статична и централизирана Маршрутизация** Централизирани адаптивни:алгоритми При централизираните адаптивни алгоритми в мрежата се създава един маршрутен управляващ център.Той изчислява маршрутните таблици на всички възли и им ги изпраща.За да се адаптират маршрутните таблици към текущата топология и текущия трафик, всички възли трябва да изпращат информация към маршрутния център. На базата на получените сведения, маршрутният център изчислява теглата на ребрата и след това пресмята оптималният маршрут между всеки два възела.Добре е да се поддържат алтернативни пътища между възлите.Aлгоритми Информацията от по-близките до маршрутния център възли ще пристигне по-бързо отколкото от по-далечните.Поради това периодът на обновяване на маршрутните таблици трябва да е поне два пъти по-голям от времето за преминаване на пакет от маршрутния център до най-отдалечения от него възел.Преизчислена маршрутна таблица, получена в един възел, не трябва да се използва веднага, тъй като маршрутните таблици пристигат по различно време в различните възли.Ако по някаква причина маршрутният център отпадне, мрежата остава без управление.За целта може да се дублира маршрутният център, но тогава служебният трафик би се увеличил твърде много.Пример на статична маршрутизация Нека е дадена следната мрежа.  Маршрутната таблица на възела J би могла да има по следния вид:Първата колона показва дестинацията на пакета. Нека, например, в J пристигне

пакет за A. Тогава J генерира случайно число между 0 и 1.Ако числото е <0.63, пакетът се изпраща към A, ако числото е > 0.63 и 0.84

пакетът се изпраща към I,ако числото е > 0.84, пакетът се изпраща към H.Статична маршрутизация. По-подробно.Ръчното добавяне на маршрути в конфигурацията на маршрутизатор се нарича статична маршрутизация (static routing). Подходящо е за малки мрежи.

Маршрутите се описват чрез фиксирани пътища (статични маршрути), които се въвеждат в маршрутизатора от мрежовия администратор. Така чрез статични маршрути (static routes) може да се опише цялата мрежа.

Този подход не е отказоустойчив. В случай на промени или прекъсвания на връзки и/или устройства не се получава автоматично пренасочване на трафика.Статична маршрутизация: В някои случаи статичните маршрути са даже за предпочитане, даже влияят положително на производителността.Това са мрежите с един единствен изход (stub networks) и маршрутите по подразбиране (default routes).Stub Networks:Статичните маршрути най-добре се вписват в Stub мрежите.stub network е като “задънена улица” –само с един изход към външния свят.На фигурата по-долу двете клонови локални мрежи - Tokyo и London са с по един изход към главната квартира.Stub Networks Не е необходимо да пускаме динамичен протокол за маршрутизация по WAN линиите между NewYork и клоновете.Статични маршрути отвеждат трафика до “клоновите” LAN-ве.Докато клоновите маршрутизатори е необходимо да “знаят” само дали пакета е насочен към мрежа извън техните локални.Т.е да го “докарат” до NewYork, който има по-богата маршрутна таблица.Конфигурации: По-долу са дадени статичните маршрути в NewYork и Tokyo.Редове 10 и 11 са статичните маршрути в NewYork към съответните “клонови LAN-ве”.Ред 19 е по подразбиране (default route) в Tokyo (маршрут до мрежа 0.0.0.0), който сочи обратно пътя към NewYork.Конфигурацията на London е подобна на Tokyo.Конфигурации: static routes: NewYork Configuration 10) ip route 192.168.2.0 255.255.255.0 192.168.4.1 11) ip route 192.168.3.0 255.255.255.0 192.168.6.2 !!! В конфигурацията освен мрежата и маската е зададен и IP адреса следващия възел по пътя (Next Hop) Конфигурации: default route Tokyo Configuration 19) ip route 0.0.0.0 0.0.0.0 192.168.4.2 Частен случай: интерфейсни маршрути Статичен маршрут, на който е зададен само изходящия интерфейс на рутера,се нарича още интерфейсен.В този случай все едно, че мрежата на крайната точка е директно закачена за интерфейс на рутер.Интерфейсни маршрути NewYork#conf t NewYork(config)#no ip route 192.168.2.0 255.255.255.0 192.168.4.1 NewYork(config)#no ip route 192.168.3.0 255.255.255.0 192.168.6.2 NewYork(config)#ip route 192.168.2.0 255.255.255.0 s 1 NewYork(config)#ip route 192.168.3.0

255.255.255.0 s 0 Интерфейсни маршрути:NewYork#sh ip route C 192.168.4.0/24 is directly connected, Serial1 C 192.168.6.0/24 is directly connected, Serial0 C 192.168.1.0/24 is directly connected, Ethernet0 S 192.168.2.0/24 is directly connected, Serial1 S 192.168.3.0/24 is directly connected, Serial0 Default Route:Маршрут по подрабиране (default route) се използва, когато в таблицата няма друг маршрут до целта.Това е маршрут до мрежата 0.0.0.0. Изходът от командата **show ip route** показва на първия ред на маршрутната таблица: **Gateway of last resort is 192.168.4.1 to network 0.0.0.0** Пакет, който не намери съвпадение в маршрутната таблица,поема към “gateway of last resort”.Това е рутер с по-подробна информация за маршрутите.Ако няма default route и адреса на получателя не бъде открит в таблицата, пакетът се изхвърля и на IP адреса на източника се въща ICMP съобщение: ‘Destination or Network Unreachable’.Терминът Gateway:‘Gateway’ е по-стария термин за маршрутизатор (рутер). През него можеше да се изпратят пакети към мрежа с различна преносна среда и канални протоколи.В днешно време показва IP адреса на устройсвото, от което се излиза от локалната мрежа към “външния свят”. И обикновено е с два интерфейса – вътрешен и външен.Докато маршрутизаторът има повече от два интерфейса и изпълнява по-сложни функции по маршрутизацията.Default Route:Маршрутът по подразбиране е много полезен в “stub” мрежите.Благодарение на него не се налага да се използва динамичен протокол за маршрутизация, което не е и необходимо в такива мрежи.Така се спестява пропускателна способност на линиите, “цикли” на CPU на рутера и памет.Default Route. Конфигуриране.Tokyo(config)#ip route 0.0.0.0 0.0.0.0 192.168.4.2 Tokyo#sh ip route Gateway of last resort is 192.168.4.2 to network 0.0.0.0C 192.168.4.0/24 is directly connected, Serial0 C 192.168.2.0/24 is directly connected, Ethernet0 S\* 0.0.0.0/0 [1/0] via 192.168.4.213.

**13 Разпределена маршрутизация с дистантен вектор** Малко история:Маршрутизиращ протокол с дистантен вектор (distance vector protocol) е използван отначало в ARPANET.По-късно в Интернет намира широко приложение като RIP (Routing Information Protocol).Основите на тези алгоритми са поставени от Белман (1957 г.), Форд и Фолкерсън (1962 г.).Затова са известни като алгоритми Белман-Форд или Форд-Фолкерсън.Само на Cisco Systems – IGRP и EIGRP.Основни принципи:Distance Vector – рутерите се анонсират(рекламират) като вектори:Посока - адреса на следвашия възел (next hop) и изходящия интерфейс и Разстояние (метрика), напр. брой възли до дестинацията (hop count).Основни принципи:Маршрутизаторите (рутери) в тези случаи не знаят целия път до крайната точка (дестинация).DV използва само:1. Посока или интерфейс, по който да се отправи пакета. 2. Разстоянието до дестинацията.Общи положения:При маршрутизацията с дистантен вектор (distance vector routing) всеки маршрутизатор изгражда и поддържа маршрутна таблица, в която всеки ред съдържа адрес на местоназначение,адрес на следващата стъпка към това местоназначение по най-добрия известен до момента път и дължината на този път (метрика).Периодически маршрутизаторите изпращат на съседите си цялата или част от маршрутната таблица. Предимства и недостатъци DV алгоритмите: - не товарят процесора и паметта;- лесни са за реализация и поддръжка;Но - Периодическите update-и отнемат пропускателна способност от потребителите.Метрика:Предполага се, че всеки маршрутизатор знае метриката на връзките до своите съседи.Ако метриката е брой стъпки или маршрутизатори до местоназначението (хопове), разстоянието до всеки съсед е 1.Ако метриката е натоварване на възела,разстоянието до всеки съсед е броя на пакетите в изходящата опашка към този пакет.Ако метриката е времезакъснение,маршрутизаторът периодично изпраща “ехо” пакети до съседните му маршрутизатори и измерва закъснението на техния отговор.Недостатък Сериозен недостатък на маршрутизиращите алгоритми с дистантен вектор е **ниската им скорост насходимост**.Добрите новини се разпространяват бързо в мрежата,но лошите новини обикновено изискват твърде голямброй периодични съобщения за да достигнат до всички маршрутизатори.Добавяне на обект:Нека маршрутизаторът *A* в началото не е включен в мрежата.Всички останали маршрутизатори знаят това – в маршрутната им таблица към направлението *A* е записано (достатъчно голямо число, трябва да е поне с единица повече от диаметъра на мрежата). Това е отразено на първия ред по-горе.След включването на *A* останалите маршрутизатори научават за това събитие чрез няколко обмена на своите вектори на разстоянията, всеки от които се извършва едновременно между всички съседни маршрутизатори.Добавяне на обект:При първата обмяна *B* научава от *А* за път с дължина 0 до *A* и записва в своята таблица, че *A* е на разстояние 1.В този момент останалите маршрутизатори все още не са научили за включването на *A*. Това е отразено на втория ред по-горе.При следващия обмен *C* научава, че от *B*съществува път до *A* с дължина 1 и записва в своя вектор път до *A* през *B* на разстояние 2 и т.н.Добавяне на обект:По-общо в мрежа с диаметър *k* хопа са необходими най-много *k* размени на съобщения за разпространяване на новината за появил се по-добър път.Иключване на обект:Нека всички маршрутизатори в началото са включени в мрежата.Да предположим, че *A* спира да работи или се прекъсва връзката от *A* до *B*, което от гледна точка на *B* е същото.При първия обмен *B* не получава информация от *A*, но получава информация от *C*, че има път до *A* с дължина 2. *B* не знае, че пътя от *C* до *A* минава през него - от негова гледна точка би могъл да съществува друг независим път от *C* до *A*, затова *B* записва в таблицата си в реда за *A* път с дължина 3 и следваща стъпка *C*.*D* и *E* не променят маршрутните си таблици при първия обмен на векторите на разстоянията.На следващия обмен *C* научава за два възможни пътя до *A*, и двата с дължина 4, единият през *B*, другият през *D.C* избира и записва в маршрутната си таблица единия от тях в зависимост от реда на обработването на съобщенията от *B* и *D*.Изключване на обект:Count to Infinity:Резултатът от продължаващия обмен е отразен в следващите редове по-горе.Той ще продължи, докато стойностите по направленията към *A* и в четирите машрутизатора не достигнат .Този проблем се нарича **броене до безкрайност (count to infinity)**.Split horizon:Едно частично негово решение е т.н.**разделяне на хоризонта** (**split horizon**).При него се въвежда ново правило - ако в маршрутната таблица на *X* в реда за *Y* е записана следваща стъпка *Z*, то *X* не изпраща към *Z* информация за маршрута към *Y*.Частично решение - **split horizon:**В горния пример на втория обмен на вектори, *C* не изпраща към *B*информация за маршрута към *A*, тъй като маршрутът от *C* към *A* минава през *B*.Въвеждането на разделяне на хоризонта не решава напълно проблема броене до безкрайност.Да разгледаме следния пример.**split horizon:**В началото *A* и *I* имат пътища с дължина 2 стъпки до *K* през *J*, а *J* има път с дължина 1 до *K*. Да предположим, че връзката между *J* и *K* отпадне.**split horizon:**Тогава на първия последвал обмен *J* няма да получи информация за нов път до *K* през *A* или *I* по правилото за разделяне на хоризонта и правилно ще заключи, че *K* е недостижим.На следващия обмен *А* и *I* научават, че няма път до *K* през *J*, но *A* научава за път до *K* през *I* с дължина 3 и *I* научава за път до *К* през *A* с дължина 3.Така въпреки разделянето на хоризонта, *A* и *I* ще броят до безкрайност.Triggered Updates:Незабавното изпращане на извънредни съобщения за обновяване (Triggered Updates) намалява вероятността за поява на цикли.Когато един маршрутизатор промени метриката за даден маршрут, той трябва веднага да изпрати извънредно съобщение.Например, връзката J-K отпада.Triggered Updates J ще съобщи веднага на A и I, че: d(J-K) =  A и I записват в маршрутните си таблици, че K е недостижим и веднага съобщават това на своите съседи.При следващо редовно (периодично)обноввяване няма да възникне “броене до безкрайност” между А и I. В таблиците им няма да има остаряла информация за K.Hold down Triggered Updates не достига едновременно до всички маршрутизатори.Маршрутизатор, още неполучил съобщението,би изпратил редовно периодично съобщение за обновяване с неточна информация до друг маршрутизатор, който вече има актуалната информация за променен маршрут.Hold down:Защита срещу това: hold down таймер.След изпращане или получаване на triggered update маршрутизаторът стартира hold down таймер.До неговото нулиране не приемат съобщения за обновяване на променения маршрут.

**14. Маршрутизация със следене състоянието на връзката.** Йерархична маршрутизация:Link State vs. distance vector: Информацията в един distance vector (DV) е сравнима с пътния знак.Докато link state (**LS**) протоколите са пътна карта.LS рутерът има пълна картина на мрежата и и по-трудно ще вземе неправилно решение.DV маршрутизират по слухове.Link State. Особености:LS – подава на съседите си информация за директно свързаните връзки и състоянието им (затова е link state).Тази информация се подава от рутер на рутер, всеки я копира, без да я променя.Всеки рутер има идентична информация за мрежата.Всеки рутер сам за себе си изчислява най-добрите пътища до съответните префикси.Алгоритъм на Dijkstra:LS протоколите - shortest path first или distributed database, се базират на алгоритъма на E. W. Dijkstra - shortest path.Такива протоколи са:- Open Shortest Path First (OSPF) за IP; - Intermediate System to Intermediate System (IS-IS) на ISO за CLNS и IP; - Optimized Link State Routing Protocol (OLSR - rfc3626).CLNS:CLNS (Connectionless Network Service).OSI Network Layer услуга (за разлика от CONS - Connection-Oriented NS) не изисква да се установи канал.CLNS маршрутизира съобщения до дестинацията независимо от други – message switching.В OSI CLNS е услуга, осигурена от CLNP (Connectionless Network Protocol) и използвана от TP4 (Transport Protocol Class 4).CLNS:Но CLNP не се използва Internet, вместо него IP.Все пак много телеком операторите още използват CLNP.Защото IS-IS (OSI layer 3 protocol) е признат от ITU-T като протокол за управление на SDH (Synchronous Digital Hierarchy).Около нас – академичната мрежа в Румъния.Пет основни действия:При маршрутизацията със следене състоянието на връзката (link state routing) всеки маршрутизатор трябва да извършва следните пет основни действия:1. Откриване на съседните маршрутизатори и техните мрежови адреси. 2. Измерване на стойностите на връзките до съседните маршрутизатори. 3. Конструиране на пакети с информация за състоянието на връзките.Пет основни действия:4. Изпращане на тези пакети до всички останали маршрутизатори.5. Изчисляване на най-късия път до всеки маршрутизатор в мрежата.Резултат от тези действия:В резултат на тези пет действия се събира и разпространява до всички маршрутизатори информация за цялата топология на мрежата.Откриване на съседните маршрутизатори: Sлед включването на един маршрутизатор неговата първа задача е да научи кои са съседите му.Това се постига чрез изпращане на “ехо” пакет по всяка от изходящите линии на маршрутизатора. От своя страна, всеки от съседите отговаря като съобщава името си. Това име трябва да бъде уникално в мрежата.(Например, IP адрес на някой от интерфейсите.)Откриване на съседните маршрутизатори:Ако два или повече маршрутизатора са свързани в мрежа с общодостъпно предаване (например Ethernet), откриването на съседите е малко по-сложно. Един възможен начин за представяне на връзките между тях е да се въведе допълнителен възел, който да отговаря на общата среда за предаване. Измерване на цените на връзките:Всеки маршрутизатор трябва да може да определи време-закъснението до своите съседи.Най-простият начин е маршрутизаторът да изпрати "ехо" пакет към всеки свой съсед на който трябва директно да се отговори. Времето от изпращането на "ехо" пакета до получаване на отговора се дели на две и по този начин се получава времето-закъснение до съответния съсед.За по-точно измерване, този процес може да се повтори няколко пъти и да се вземе средната стойност. Този метод предполага, че връзките са симетрични,което не винаги е вярно.Измерване на цените на връзките:Друг въпрос е дали при измерването да се взима предвид натовареността на възлите.Разликата се постига в зависимост от това кога маршрутизаторът стартира измерването:когато пакетът постъпва в съответната изходяща опашка или когато пакетът се придвижи в началото на опашката.Включването на натовареността на възлите има предимства и недостатъци.Предимството е, че от две линии, които имат еднаква скорост за по-къса ще се счита по-ненатоварената линия. Това ще доведе до по-голяма ефективност.Измерване на цените на връзките:Недостатъкът може да се илюстрира със следния пример.Нека една мрежа е разделена на две части, които са свързани чрез две линии *А* и *B*.Да предположим, че в даден момент по-голямата част от трафика между двете части на мрежата минава по линия *А*.Тогава при следващото изчисляване на маршрутните таблици трафикът ще се насочи към по-добрата линия *B*.Този процес ще се повтаря циклично и ще доведе до нестабилност в работата на мрежата.link state packets:След като събере необходимата информация за състоянието на връзките си, следващата задача на маршрутизатора е да конструира пакет, който съдържа тази информация.Пакетът трябва да съдържа уникалното име на подателя, пореден номер, срок на годност и списък със съседите на подателя, като за всеки съсед е указана цената на връзката до него. Определянето на момента, в който трябва да бъдат подготвени и изпратени пакетите, е важна задача. link state packets:Един възможен начин е това да става през определени равни интервали от време.Друга по-добра възможност е пакетите да се подготвят и изпращат само при промяна в топологията на мрежата - след отпадане или поява на нов съсед или промяна в цената на някоя връзка.Нека да разгледаме следната примерна мрежа.Ребрата имат етикети със съответното време-закъснение.Разпространяване на **link state packets:**Най-съществената част на алгоритъма е надеждното доставяне на пакетите с информацията за състоянието на връзката до всички маршрутизатори.За разпространението на пакетите се използва методът на наводняването (flooding). При него всеки пакет се изпраща по всички линии,освен линията по която е пристигнал.Разпространяване на **link state packets:**Обработката на всеки пристигнал пакет започва с проверка дали пакетът има по-голям пореден номер в сравнение с най-големия пореден номер, който е пристигнал до този момент от този източник.Ако номерът е по-голям, информацията от пакета се записва в таблицата с информация за състояние на връзките и пакетът се предава по останалите линии.Ако номерът е по-малък или равен, пакетът се отхвърля.Пореден номер:Този алгоритъм има някои проблеми.Ако поредният номер не е достатъчно голям, той може да се превърти.Затова се използват 32-битови поредни номера. Ако пристига 1 пакет/s, то за превъртане на номера ще са необходими около 137 години.Ако пристига 1 пакет/10 s (на практика) -1361 години.Поле за срок на годност:В полето за срок на годност маршрутизаторът-подател указва продължителността на интервала от време в секунди, през който пренасяната от него информация трябва да се счита за валидна.Всеки маршрутизатор, който получи даден пакет намалява с единица стойността на това поле преди да го предаде към своите съседи.Освен това, след като маршрутизаторът запише данните от пакета в своята таблица, той продължава да намалява срока на годност на тези данни на всяка следваща секунда.Ако срока на годност стане 0, данните се изтриват.По този начин се премахва опасността остаряла информация за състоянието на връзките да се разпространява и използва прекалено дълго време от маршрутизаторите.Таблица с информация за състоянието на връзките:Таблицата с информация за състоянието на връзките, която се използва от маршрутизатор *B* примера в слайд 17 изглежда примерно по следния начин: Всеки ред от таблицата съответства на пристигнал, но все още необработен пакет. Полето *Source* е източникът на пакета, *Seq* е поредният му номер, *Age* е срокът на годност.С всеки пакет се свързват флагове за изпращане (*send flags*) флагове за потвърждение (*ACK flags*) за всяка от изходните линии на маршрутизатора *B*. Флаговете за изпращане указват по кои линии трябва да се изпрати пакета.Флаговете за потвърждение указват по кои линии да се изпрати потвърждение за получаването на пакета.Ако в *B* пристигне дубликат на някой от пакетите в таблицата, то съответните флагове трябва да се актуализират.Например, ако пристигне дубликат на пакета със състоянието на *C* от *F*,преди този пакет да бъде препратен към *F*, то флаговете за изпращане на пакета ще се променят на 100, а флаговете за потвърждение - на 011.Изчисляване на новите маршрути: След като един маршрутизатор получи пълна информация за състоянието на връзките на всички останали маршрутизатори, той може да приложи алгоритъма на Дейкстра.Всъщност всяка връзка се представя два пъти,по веднъж за всяка посока. Двете цени могат да се усреднят или да се използват отделно.Изчислените маршрути се записват в маршрутните таблици.Необходимата памет за съхраняване на информацията за състоянието на връзките за мрежа с ***n*** маршрутизатори, всеки от които има по ***k*** съседи е пропорционална на ***nk***.Така големите по размер мрежи изискват използване на маршрутизатори с голям обем памет. Йерархична маршрутизация С увеличаването на размерите на мрежата нараства обемът на маршрутните таблици, което изисква повече памет и процесорно време за тяхната обработка.Това налага въвеждането на йерархично маршрутизиране, при което мрежата се разделя на **области**.Маршрутизаторите в една област знаят всичко за вътрешната структура на своята област, но не знаят вътрешната структура на останалите области.За по-големи мрежи може да е необходима йерархия с повече от две нива.Пример на мрежа с йерархична маршрутизация на две нива:Като пример да разгледаме следната мрежа с йерархична маршрутизация на две нива.Метриката е в хопове.

Пример Съкратената таблица за маршрутизатора *1A* съдържа 7 реда и има следния вид: В нея са запазени маршрутите към направленията в "област 1".Маршрутите към направленията в останалите области са обобщени в един ред, като се използва маршрутизатора от съответната област, който е най-близо до маршрутизатора *1A*.Спестяването на памет от маршрутните таблици има отрицателен ефект - някои от пътищата увеличават своята дължина.Например, най-късият път от *1A* до *5C* минава през "област 2", но при йерархично маршрутизиране ще се използва пътят през "област 3", тъй като това е по-добре за повечето маршрутизатори от "област 5".Ако *n* е броят на маршрутизаторите в една мрежа, може да се покаже, че оптималният брой области, всяка с по равен брой маршрутизатори, е най-близкото цяло до .*n*

**15. Маршрутен протокол RIP** Общи положения **RIP** (routing information protocol) e широко използван маршрутизиращ протокол с вектор на разстоянието (distance vector).Той е подходящ предимно за малки мрежи, в които относително рядко настъпват промени в топологията.Всеки ред в маршрутната таблица на RIP маршрутизаторите съдържа информация за направлението,следваща стъпка към това направление и метрика.Метриката обозначава разстоянието в стъпки до местоназначението, т.е.метриката използвана от RIP протокола е брой хопове. Максималният брой хопове в една RIP мрежа е 15.RIP таймери:RIP на всеки 30 секунди изпраща копие на маршрутната таблица към съседните маршрутизатори.Таймерът за невалиден маршрут (hold down time) е 180 s.Определя интервала от време, след който даден аршрут се счита за невалиден, ако маршрутизаторът не е получил съобщения за него.Когато даден път бъде отбелязан като невалиден, се изпращат съобщения с тази информация към съседните маршрутизатори и се преустановява използването му. Тези съобщения се изпращат до изтичането на таймера за изтриване на маршрут (flush timer).След което пътят се изтрива окончателно от маршрутната таблица.Формат на RIP пакетите:Първата версия на RIPv1 не поддържа subnet маски, т.е. VLSM, респ. CIDR.Втора версия на протокола - RIPv2,поддържа VLSM, респ. CIDR. Форматът на пакетите на версия RIPv2 е следния:Първите три полета Command, Version и Routing domain представляват заглавната част на пакета, а останалите шест полета съдържат данни за маршрути и комбинация

от тях може да се повтаря до 25 пъти в един RIPv2 пакет.За пренасяне на информацията от по-големи маршутни таблици се използват няколко RIPv2 пакета.Полето Command указва дали пакетът съдържа заявка или отговор.Полето Version указва версията на протокола, за RIPv2 тази стойност е 2.Полетaта Routing domain и Route Tag не се използват и се запълват с нули.Полето Address family е равно на 2, ако

следва IP адрес. Ако имаме заявка за цялата маршрутна таблица, е 0.Сходимост на RIP:При промяна в топологията на мрежата се налага всички маршрутизатори да преизчислят своите вектори на разстоянията и да достигнат до непротиворечиво описание на новата топология.За увеличаване на скоростта на сходимост на RIP се използват различни методи, например разделяне на хоризонта (split horizon).Тези методи намаляват вероятността за поява на цикли в маршрутите, но не могат да гарантират отсъствието им.Count to infinity

Максималният брой хопове в RIP е 15.Всяко местоназначение, което е на разстояние над 15 хопа се приема за недостижимо.Това прави невъзможно прилагането на RIP в мрежи с повече от 15 рутера.Но ограничава ситуацията “броене до безкрайност” (Count to infinity), при която могат да се получат цикли в маршрутите.Версии на RIP:RIPv1 (RFC 1058) прилага само classful маршрутизация.Т.е периодичните updates не носят subnet информация.Не е възможно да имаме подмрежи от един и същи клас с различни маски. С други думи, всички подмрежи от даден kлас трябва да бъдат с еднакви маски.RIPv2 е разработен през 1994 г. и има възможност да носи subnet информация, да поддържа CIDR.За поддържане на обратна съвместимост с версия 1 запазено е ограничението от 15 хопа. За сигурност е въведена аутентикация с явен текст, подобрена с MD5 (RFC 2082).За да не се товарят хостове, които не са участници в RIP, RIPv2 “мултикаства”обновленията на адрес 224.0.0.9, за разлика от RIPv1, който е broadcast.RIPng (RFC 2080) е разширение на RIPv2 за поддържане на IPv6.- RIPng се базира на RIPv2 – в маршрутната таблица IPv6 префикс, next-hop IPv6 адрес – използва мултикаст (FF02::9) за updates Конфигуриране на RIP:Router(config)# router rip Router(config-router)# network 10.0.0.0 Router(config-router)# exit Router(config)# interface ethernet1 Router(config-if)# ip address 10.1.1.1 255.255.255.0 Router(config-if)# no ip split-horizon Router(config-if)# exit Passive-interface eth0 ! не изпраща update-и по interface eth0

**16. Маршрутен протокол OSPF**

Общи положения Open Shortest Path First (**OSPF**) е динамичен протокол за маршрутизация.Той е протокол със следене на състоянието на

връзката (link-state routing protocol) Попада в групата на протоколите за вътрешна маршрутизация (interior gateway protocols - IGP), т.е в

рамките на една автономна система – autonomous system (AS).OSPF Version 2 за IPv4 е дефиниран в RFC 2328 (1998).Обновленията за IPv6 са специфицирани като OSPF Version 3 в RFC 5340 (2008).OSPF е най-широко приложимия IGP в големите корпоративни мрежи; IS-IS, е по-приложим в големи мрежи на доставчици на мрежови услуги.Optimized Link State Routing Protocol (**OLSR**) е оптимизиран за мобилни и др. безжични ad-hoc мрежи.Характеристики на OSPF:OSPF изгражда маршрутни таблици по destination IP адресите в IP пакетите.Поддържа VLSM, респ. CIDR.OSPF разпознава промени в топологията като отпадане на връзки много бързо и се конвергира, за секунди, в нова топология, без зацикляния (loop-free).За целта всеки OSPF рутер събира информация състоянието на връзките (linkstate

information), за да изгради цялостната топология на мрежата, състояща се от области (*areas).*Теоретично топологията се представя като граф,възлите на който са рутерите, а клоните –комуникационните линии между рутерите.На клоните се присвояват стойности, които са

обратно пропорционални на скоростта на линиите. Т.е по-бързата връзка е с по-малка стойност.От тази пълна топология се изчислява “дървото на най-късия път” (shortest path tree), по най-ниската стойност, за всеки маршрут по алгоритъма на (Dijkstra's algorithm). Информацията състоянието на връзките се поддържа във всеки рутер под формата база от данни засъстояние на връзките (link-state database - LSDB), която е мрежовата топология,пълния граф на мрежата. OSPF съседство:OSPF първо установява съседство (adjacency) между директно свързани рутери с цел обмен на маршрутна информация.Hello протоколът отговаря за установяване и поддържане на

съседство.Hello пакети се изпращат периодически през всички активни интерфейси.Съседство между рутерите:Рутерите в Ethernet локална мрежа избират *designated router* (DR) и резервен, *backup designated router*(BDR), който действа като концентратор с цел намаляване на трафика между рутерите.OSPF изпращат "hello" пакети и обновления за състоянието на връзките (link state updates) и по unicast, и по multicast.Състояния на OSPF. Down.**DOWN.** От съседа все още не се получава информация.Състояния на OSPF. Init. **Init** е еднопосочен (one-way) Hello.R1 изпраща Hello пакет.R2 обявява one-way, защото не вижда своя router ID в Hello пакета.Състояния на OSPF. 2-Way:**2-way** състояние при установяване на двупосочни комуникации – начало на OSPF съседство. Избор на DR и BDR.router ID на R2 е по-голям, избран е за DR. (Designated Router или Master) Състояния на OSPF. Exstart.**Exstart** инициализира процеса на синхронизация на топол. Б.Д. (DBD).Избират се Master и slave (или DR и BDR), Seq0 (началния) за DBD обмена.Състояния на OSPF.Exchange:**Exchange**. Рутерът описва цялата си link-state database чрез DBD пакети.Всеки DBD Seq се потвърждава.Състояния на OSPF. Loading.**Loading**. LS request се изпращат да заявят по-нови състояния на връзките (LSA - link state advertisement), които не са били получени по време на exchange.Състояния на OSPF. Full:**Full**. Между OSPF съседите е обменена пълната информация.R1 и R2 са си обменили цялата база от данни и с във Full.Характеристики на OSPF Идентични копия на LSDB периодически се обновяват чрез “наводняване” на всички рутери в дадена OSPF област (area).По принцип area 0 е опорната - "backbone" област на OSPF мрежата, а OSPF областите с други номера обслужват други части от корпоративната мрежа.Всяка допълнителна OSPF area трябва да има директна или виртуална връзка към backbone OSPF area.По строга дефиниция областите в OSPF се номерират като IP адресите. Опорната,например е 0.0.0.0. Но това правило не се спазва стриктно. Cisco IOS, например, допуска едноцифрени номерации.Маршрутизацията между областите се осъществява през опорната.Multicast адреси:Multicast адресите 224.0.0.5 (всички SPF/link state рутери, AllSPFRouters) и 224.0.0.6 (всички Designated рутери, AllDRouters) са резервирани за OSPF (RFC 2328).За разлика от BGP, OSPF не се базира на TCP или UDP (RIP).Директно използва IP (IP protocol 89).OSPF има собствени механизми за откриване и корекция на грешките. Сигурност на операциите:Има възможност за защита на операциите в OSPF чрез размяна на пароли между рутерите.В явен текст или Message Digest 5 (MD5) за автентикация на съседни рутери,преди да са образували съседство и са започнали да приемат link-state advertisements (LSA). Области: OSPF мрежата е разделена на области (*areas)*, които се означават с 32-битови идентификатори.Тези идентификатори се записват подобно на IPv4 адреси. Но те не са IP адреси и спокойно могат да се дублират с истинските.Идентификаторите за IPv6 реализациите на OSPF (OSPFv3) използват същите 32-битови означения.Повечето реализации на OSPF позволяват да се пише и в друг fормат.Например, area 1 в повечето случаи ще бъде възприета правилно като 0.0.0.1, но за някои ще бъде 1.0.0.0. Затова се препоръчва да се ползва десетично-точковата нотация.Областите са логически групи от хостове и мрежи, включително и техните рутери, които имат интерфейси към тези мрежи.Всяка област поддържа собствена link state database, чиято информация може да бъде представена в обобщен вид (summary) пред останалата част от мрежата. Така топологията на областта ще е неизвестна извън нея.И ще се намали служебния трафик за маршрутизация в рамките на OSPF мрежата.virtual link:Всички OSPF области трябва да бъдат свързани към опорната.Но освен по физически, тази връзка може да се осъществи и по виртуален канал.Например, област 0.0.0.1 има физически канал до 0.0.0.0. Но 0.0.0.2 няма директна връзка с опорната, но има с 0.0.0.1.Област 0.0.0.2 по виртуален канал през *транзитната бласт* 0.0.0.1 може да достигне опорната.Транзитната област не трябва да бъде stub по никакъв начин. Backbone area:Опорната област backbone area - *area0.0.0.0*) е ядрото на OSPF мрежата.Всички други области са свързани към нея.Маршрутизацията между областите се осъществява чрез гранични рутери,имащи интерфейси и в опорната, и в някоя от другите области.Тези рутери се наричат Area Border Routers (ABRs).От area 0.0.0.0 се реализира и връзката на OSPF мрежата с външния свят с помощта на рутер, който има външен интерфейс - Autonomous System Border Router (ASBR).По наследство на OSPF мрежата с всичките й области се казва “автономна сiстема”.

Това няма нищо общо с видимите през Internet автономни системи, съгласно RFC1996, които могат да съдържат множество OSPF, RIP и р. мрежи. Stub area:stub area е област, която не получава външни за OSPF мрежата маршрути с изключение на маршрута по одразбиране за навън (default route) Но областта получава маршрути от другите области (*inter-area)*.Дефинирането на такава област е олезно, ако целият излаз към Internet минава през рутери (ASBRs) в Area 0.0.0.0, и трябва да се избере най-добтия път до друга област. Totally stubby area: *totally stubby area* (TSA) е дефиниция, въведена от Cisco (IOS) Подобна е на stub area, но освен външни маршрути не озволява и *summary* маршрути, т.е *inter-area* (IA), междуобластните маршрути не се обобщават в TSA. Единственият начин трафикът да е рутира извън такава област е чрез default route. SA - в периферията на мрежата - рутери от по-нисък клас. TSA може да има повече от един ABR. Трафикът извън областта се праща през ABR с най-ниска метрика (“най-близкия"). Not-so-stubby area:*not-so-stubby area* (NSSA) sъщо е stub area, но която може да импортира външни за OSPF мрежата маршрути и да ги изпраща към опорната област. Но тя не е в състояние да получава външни маршрути от опорната или други области. OSPF метрика:Метриката в OSPF е стойността на пътя -маршрута (*path cost)*. Тя се определя от скоростта (bandwidth), зададена на интерфейса, водещ към съответния маршрут. Cisco работи с метрика 10^8/bandwidth.Тази базова стойност 10^8 по подразбиране може да бъде променена. Скорости от 1000 Mbps (1 Gbps) вече преобладават, а 10 Gbps бурно навлизат в опорните (backbone) линии.При 10^8, 100 Mbps ще е със стойност 1, а 10 Mbps - 10 и т.н. Но за по-бързи от 100Mbit/s,cost <1.Типове OSPF метрика:Описаната в предишния слайд метрика е директно приложима само, ако е от един и същи тип.Имаме четири типа метрика, принадлежащи на 4 типа маршрути. Редът на предпочитане на един от тези 4 типа е както следва: 1. Intra-area – вътрешни за областта; 2. Inter-area – между областите; 3. External Type 1 – включва стойността на външния маршрут плюс сумата от стойностите на вътрешните пътища до ASBR, който е анонсирал маршрута; 4. External Type 2 - включва единственоthe стойността на външния маршрут. Designated router:*designated router* (DR) е интерфейс на рутер избран между sъществуващите в сегмент с множествен достъп и общодостъпно предаване (broadcast multiaccess), т.е LAN Ethernet. DR не е определен тип OSPF рутер. Един от интерфейсите на даден физически рутер може да е DR, а друг - backup designated (BDR), трети да са non-designated.При първоначално включване на OSPF rутерите в даден сегмент, след установяване на съседство всеки-с-всеки се избират DR и BDR по следните критерии: Избор на DR и BDR Рутерът, изпращащ Hello пакети с най-висок приоритет печели “състезанието” за DR, евентуално за BDR. Ако два и повече рутера са с еднакъв най-висок приоритет, рутерът, изпращащ Hello пакети с най-висок RID (Router ID) “побеждава”. RID е най-високият логически (loopback) IP адрес на рутера. Ако такъв не е конфигуриран, избира се най- високия IP адрес, конфигуриран на активен интерфейс. (напр. 192.168.0.1 е по-висок от 10.1.1.2). Рутер със следващ по-големина приоритет или

RID става BDR. Стойностите на приоритета варират от 0 - 254, колкото е по-голяма стойността, толкова по- големи са шансовете за успех в избора на DR или BDR. В рутерите на Cisco приоритетът по подразбиране е 1. Т.е, ако администраторът не се намеси (не е конфигурирал и Loopback интерфейси), RID ще се определя по най- високия IP на активните интерфейси. Желателно е да използваме приоритет и/или Loopback, за да наложим избора на най- мощните рутери за DR и BDR, които да комуникират с всички останали в сегмента. Ако OSPF рутер с по-висок приоритет стане онлайн след избора, той няма да стане DR или BDR, докато не отпадне някой от вече избраните. Ако избран DR отпадне, BDR става новия DR. Провежда се избор на нов BDR. Ако новият DR отпадне, а оригиналният DR същевременно се е възстановил, той отново става DR, но текущият BDR си остава такъв. Designated router:Cелта на избора на DR е да се намали служебния tрафик в мрежата.Осигурява се един единствен източник на обновления на маршрутите. DR поддържа пълна топологична таблица

и изпраща обновления (updates) до другите рутери по multicast.Всички маршрутизатори в областта са в slave/master отношения с DR. Те

формират съседства само с DR и BDR.Designated Router (Con't):Slave рутер изпраща update към DR и BDR на multicast адрес 224.0.0.6. DR след това разпраща update до другите рутери в областта на multicast адрес 224.0.0.5. Така няма да се налага рутерите постоянно да си обменят “update-и” всеки-с-всеки, а ще ги получават от един единствен източник. Използването на multicasting допълнително намалява натоварването в мрежата.Външни маршрути. load-sharing. OSPF осигурява по-добро балансиране на натоварването (load-sharing) на външните връзки от другите IGPs.default route до ISP се инжектира в OSPF от множество ASBRs като Type I външен маршрут. Външната стойност на маршрута е една и съща, но другите маршрутизатори ще изпратят външния си трафик до ASBR, до който имат път с най-малка стойност. Чрез нагласяване на стойностите натоварването на ASBRs може да се разпредели равномерно. Ако обаче default route от различни ISP-та се инжектира с различни външни стойности като Type II външен маршрут, маршрутът по подразбиране с по-ниска стойност става основен, а другите ще са резервни. Конфигуриране на OSPF: Имате 5 рутера, конфигурирани с OSPF: •Router A и Router B с а вътрешни за Area (област) 1. •Router C е OSPF area border router (ABR). Area 1 - интерфейс E3, Area 0 - S0. •Router D е вътрешен за Area 0 (backbone area). •Router E е OSPF autonomous system boundary router (ASBR). BGP маршрути се вмъкват (редистрибутират - redistributе) в OSPF и съответно се анонсират от OSPF. (!!!това е само пример на Cisco, но не трябва да се допуска, ще задръсти рутера) interface ethernet 1 ip address 192.168.1.1 255.255.255.0 router ospf 1 network 192.168.1.0 0.0.0.255 area 1 Router A—Internal: Wildcard mask Забележете! 0.0.0.255 вместо 255.255.255.0 В Cisco IOS реализацията на OSPF се прилага wildcard mask (огледална на subnet mask). Т.е вместо .AND. >> .OR. В zebra си е с префикс: /m, router ospf network 10.0.0.0/8 area 0 Router B—Internal Router interface ethernet 2 ip address 192.168.1.2 255.255.255.0 router ospf 202 network 192.168.1.0 0.0.0.255 area 1 Router C—ABR interface ethernet 3 ip address 192.168.1.3 255.255.255.0 interface serial 0 ip address 192.168.2.3 255.255.255.0 router ospf 999 network 192.168.1.0 0.0.0.255 area 1 network 192.168.2.0 0.0.0.255 area 0 Router D—Internal Router: interface ethernet 4 ip address 10.0.0.4 255.0.0.0 interface serial 1 ip address 192.168.2.4 255.255.255.0 router ospf 50 network 192.168.2.0 0.0.0.255 area 0 network 10.0.0.0 0.255.255.255 area 0 Router E—ASBR interface ethernet 5 ip address 10.0.0.5 255.0.0.0 interface serial 2 ip address 172.16.1.5 255.255.255.252 router ospf 109 network 10.0.0.0 0.255.255.255 area 0 redistribute bgp 65001 metric 1 metric-type 1 router bgp 65001 network 192.168.0.0 network 10.0.0.0 neighbor 172.16.1.6 remote-as 65000 Router E – default route ip route 0.0.0.0 0.0.0.0 172.16.1.6 router ospf 109 network 10.0.0.0 0.255.255.255 area 0 default-information originate Optimized Link State Routing:Protocol: Optimized Link State Routing (**OLSR** - rfc3626) протоколът е предназначен за мобилни **ad hoc** мрежи. **Multipoint relays (MPRs).** MPRs са избрани възли, които единствено и само те препращат broadcast съобщения по време на f flooding процеса. Така значително се редуцира служебния трафик. MPR анонсира периодически информация, че има достижимост до възлите, които са го избрали като такъв. Т.е разпространява се частична LS информация. OLSR създава оптимални маршрути (метрика брой хопове).OLSR. Избор на MPR. Даден възел (N2) избира своя MPR: (1) повече 2-hop възли да бъдат обхванати; (2) да се минимизира броя на MPRs.

**17. Маршрутен протокол BGP** Понятие за автономна система (AS):Автномната система (**autonomous system - AS)** е сбор от свързани помежду си IP мрежи (префикси), които са под административното и техническо управление на мрежов оператор или др. организация (напр. СУ). В рамките на AS е възможно да работят различни вътрешни протоколи за маршрутизация (IGP). AS поддържат строго дефинирана политика за маршрутизация в Internet (вж. RFC 1930). AS трябва да има глобален уникален номер (ASN - Autonomous System Number) Този номер се използва при обмен на маршрутизираща информация със съседни AS-и и като идентификатор на самата AS. Кога ни трябва AS (ASN) AS са задължителни при обмен на външни маршрути с други ASs с помощта на протоколи за външна маршрутизация. В момента такъв е BGP (Border Gateway Protocol). Но това не е достатъчно условие, за да искаме да имаме AS. Кога да, кога не – AS AS ни е необходимa единствено и само тогава, когато имаме политика за маршрутизация (routing policy), различна от тази на други партньори - съседи (peers). routing policy – как останалата част от Internet взима решения за маршрутизация на базата на информация от нашата AS. Кога да, кога не – AS Single-homed site, единствен или множество префикси, свързан към един единствен доставчик (т.e. една AS). Не ни е необходима AS. Префиксът/те се поставя в AS на провайдера. Multi-homed site. Необходима е AS. multi-homed означава префикс или група от префикси, която се свързва към повече от един доставчик (т.e. повече от една AS, всяка със своя политика). ASNs До 2007 г. ASNs бяха единствено 16-битови, максимален брой 65536. IANA е резервирала следния блок от AS номера за частно ползване (да не се анонсират в глобалния Internet): 64512 – 65535 ASN 0 означава немаршрутизиранa мрежа. Раздаване на ASNs Всички останали номера (1–64495) са раздадени от IANA. IANA ги разпределя на съответните RIR, които от своя страна присвояват AS номера на организации в тяхната област, които отговарят на дадените по-горе критерии. (RIR за територията на България е RIPE, <http://ripe.net>). Разпределението на ASN ресурса от страна на IANA до момента, можете да видите на : <http://iana.org/assignments/as-numbers/asnumbers>. Xml 32-битови AS номера Поради големия брой PI оператори адресното пространство на 16-битовите автономни системи застрашително се запълва.Към 02.09.2009 са раздадени номера до 56319 включително. 56320-64495 са резервирани от IANA. Затова се въведоха 32-битови AS номера, RFC 4893, които вече се раздават от IANA. 16-битови AS се явяват подмножество на 32- битови AS (с 16 нули отляво). Гарантира се плавен преход за разлика от IРv4 към IPv6. Все пак зависи от софтуера. **Border Gateway Protocol** (**BGP**) **Border Gateway Protocol** (**BGP**) е основният протокол за маршрутизация в Internet. Поддържа таблица от IP мрежи (префикси), които определят достижимостта на мрежите между автономните системи. BGP е протокол с вектор на пътищата, path vector protocol. Вектор на пътищата BGP не използва метриката на вътрешните протоколи, а взема решения за определяне на маршрути на база на пътя между ASs, мрежови политики и/или правила. От 1994 г. насам се използва версия 4 на протокола, която поддържа CIDR и обединяване (агрегация) на маршрути, с което се намалява размера на маршрутните таблици. От януари 2006 г. версия 4 е стандартизирана в RFC 4271. BGP4/4+ в момента. Принцип на действие BGP съседите (neighbors или peers) - маршрутизатори, се формират, след като ръчно са зададени. Между тях се установава TCP сесия по порт 179. Всеки BGP възел периодично изпраща 19- байтови “keep-alive” съобщения за поддържане на връзката. BGP единствен от маршрутизиращите протоколи използва TCP за танспорт, което го прави приложен протокол до известна степен. **iBGP и eBGP** Когато BGP работи в рамките на AS, третира се като вътрешен (iBGP *Interior Border Gateway Protocol*). Когато работи между ASs, нарича се външен (eBGP *Exterior Border Gateway Protocol*). Маршрутизаторите на границата на дадена AS, които обменят информация с друга AS, се наричат гранични (border или edge). **iBGP и eBGP. Конфигурации.** Router B: router bgp 200 neighbor 129.213.1.2 remote-as 100 neighbor 175.220.1.2 remote-as 200 Router C: router bgp 200 neighbor 175.220.212.1 remote-as 200 neighbor 192.208.10.1 remote-as 300 Състояния при установяване на BGP сесия За да установи сесия с партньор (peer), BGP преминава през 6 състояния, описани с краен автомат (finite state machine - FSM). Това са: Idle, Connect, Active, OpenSent, OpenConfirm и Established. В BGP е дефинирана променлива на състоянието, която определя в кое от шестте състояния се намира сесията. При преход от едно състояние в друго се генерират стандартни съобщения. Idle, Connect и ... Първоначално BGP маршрутизаторът е в състояние “Idle”. Инициализира всички ресурси, отказва всички входящи опити за установяване на BGP свързаност и инициира TCP сесия със съседа си. Второто състояние е “Connect”. Маршрутизаторът изчаква да се установи TCP сесия. Ако е успешно, преминава в "OpenSent". Ако не, преминава в състояние "Active", докато се нулира таймера ConnectRetry. Active, OpenSent, Established В състояние "Active" маршрутизаторътът нулира таймера ConnectRetry, след което се връща в състояние "Connect". След "OpenSent" маршрутизаторът изпраща съобщение Open и чака за подобно в отговор. Разменят се съобщения Keepalive и след успех рутерът влиза в състояние “Established”. Готов е да изпраща и получава от съседа си съобщения Keepalive, Update и Notification.Обмен на маршрутна информация BGP съседите си обменят пълната маршрутна информация след установяване на TCP сесия между тях. Или част от маршрутната таблица, зависи от споразумението между страните,политики, филтри и т.н. При промени в маршрутната таблица, BGP маршрутизаторите изпращат на съседите си само променените маршрути.NLRI Не изпращат периодични обновления (routing updates). Рекламират (advertise) само оптималния път до дадена дестинация. В BGP описанието на маршрут до дадена дестинация се нарича Network Layer Reachability Information (NLRI). NLRI включва префикса на дестинацията и дължината му, пътят през автономните системи и следващия възел, както и допълнителна информация - атрибути.Избор на маршрут BGP не носи със себе си “политики”, а по-скоро информация, с чиято помощ BGP рутерите вземат “политически” решения, съгласно наложени правила, определени чрез атрибути. BGP attributes •--Weight •--Local preference •--Multi-exit discriminator •--Origin •--AS\_path •--Next hop •--Community Weight *Weight* е специфичен за Cisco и е локален за рутера. Не се рекламира на съседите.Предпочита се маршрут с най-голяма стойност на weight. Local Preference Локален за AS – нетранзитивен local preference = 100, когато е научена от съседна AS Влияе на избора на път за изходящия трафик Път с най-висок local preference печели MED Между AS – нетранзитивен, опция Да определи най-добрия път за входящ трафик Път с най-нисък MED печели Ако не е зададен MED, приема се =0 (RFC4271). Origin Как BGP научава за конкретен маршрут. Три възможни стойности: **•IGP**―Маршрутът е вътрешен за AS- източник. Когато е в резултат на router BGP командата network. •**EGP**―Маршрутът е научен чрез eBGP. •**Incomplete**―Произходът origin) на маршрута е неизвестен или научен по друг начин. Напр. Разпространен (redistributed) в BGP. AS\_path Когато реклама на м aршрут прминава през авт. система, нейният AS No. се добавя във верижен списък от номера на AS. AS PATH prepend Караме префиксите в Ректорат да излизат през border-rectorate Next-Hop IP адресът, чрез който се достига рекламиращият рутер. За еBGP съседи - IP адреса на връзката между тях. За iBGP, eBGP next-hop се пренася през локалната AS. Community Групиране на дестинации (communities), към които се прилагат решения за маршрутизация. Видове community атрибути: •**no-export**―да не се рекламира този маршрут на еBGP peers (съседи). •**no-advertise**―да не се рекламира този маршрут на всякакви съседи. •**internet**―да се рекламира този маршрут към Internet общността. Избор на най-добър път BGP може да получи множество “реклами” за даден маршрут.Най-добрият се записва в маршрутната таблица по следните критерии: **•next hop** е недостижим, изхвърля. **•weight** = max. Ако weights са равни: **•local preference** = max. Ако LocPref са равни: •Път с произход от **BGP на този рутер**. Ако няма: **•AS\_path** = min. Ако AS\_path са равни: •**origin** = min (IGP < EGP < incomplete). Ако origin са еднакви: **•MED** = min. Ако MED са равни: •Предпочита се **eBGP път пред iBGP път**. Ако пътищата са еднакви (eBGP или iBGP): •Предпочита се пътя през **най-близкия IGP съсед**. При равни условия: •Предпочита се пътя с **най-малък** IP адрес - **BGP router ID**. Свързаност и научаване на маршрути в **BGP** По принцип всички BGP mаршрутизатори в дадена AS трябва да бъдат конфигурирани да “говорят” всеки с всеки (full mesh). При това положение броят на връзките нараства квадратично с увеличаване на броя на рутерите. BGP има две решения на това неудобство: рефлекторна схема (route reflectors - RFC 4456 ) и конфедерации (confederations - RFC 5065). BGP Peer Groups BGP peer group представлява група от BGP съседи, които споделят обща политика, определена от маршрутни карти и филтри - route maps, distribution lists. Вместо политиката да се прилага на всеки съсед поотделно, тя се прилага върху цялата група. Членовете на групата наследяват всички конфигурации на групата. AS 5421 има peering споразумения с основните ISP да й подават само собствените си префикси. Рефлектори В автономна система с iBGP трябва да има свързаност към всички iBGP peers (съседи), т.е “всеки с всеки” - full mesh. С помощта на рефлекторна схема се редуцира броя на iBGP съседите и от там натоварването на процесори и комуникационни канали. Един рутер (или два за резервираност) става рефлекторен сървър, а другите – рефлекторни клиенти. Големина на маршрутната таблица Един от основните проблеми пред BGP, респ. Internet, е растежа на глобалната таблица с маршрутите. Не всички рутери са в състояние да я поемат (RAM, CPU) и ефективно да обработват трафика. И, още по-важно, колкото е по-голяма таблицата, толкова по-бавно се стабилизира (конвергира). В момента броят на префиксите в Глобалната мрежа е над 300 000. Агрегиране и/или сумаризиране на маршрути Да приемем, че на AS1 е присвоено адресно пространство 172.16.0.0/16 (summary). AS1 иска да анонсира по-специфични маршрути: 172.16.0.0/18, 172.16.64.0/18 и 172.16.128.0/18. Префиксът 172.16.192.0/18 не съдържа никакви хостове и AS1 не го анонсира. При това положение AS1 ще анонсира 4 mаршрута:172.16.0.0/16, 172.16.0.0/18, 172.16.64.0/18 и 172.16.128.0/18. Тези 4 маршрута ще бъдат видяни от AS2. Въпрос на политика е дали да ги копира 4-те или или да запише само сумаризирания (summary), 172.16.0.0/16. Ако AS2 иска да изпрати данни към 172.16.192.0/18, те ще се отправят по маршрут 172.16.0.0/16. Граничният маршрутизатор на AS1 или ще изхвърли пакета, или ще го върне като “unreachable” в зависимост от конфигурацията. Ако AS1 реши да не анонсира маршрут 172.16.0.0/16 (т.е да не сумаризира) и остави 172.16.0.0/18, 172.16.64.0/18 и 172.16.128.0/18, в таблицата й ще има три маршрута. AS2 ще вижда тези три маршрута в зависимост от политиката си или ще запише в паметта и трите, или ще агрегира префиксите 172.16.0.0/18 и 172.16.64.0/18 на 172.16.0.0/17. Тогава в паметта на граничния маршрутизатор на AS2 ще се съхраняват само два маршрута: 172.16.0.0/17 и 172.16.128.0/18. Ако AS2 иска да изпрати данни към 172.16.192.0/18, те ще бъдат изхвърлени на нейната граница или към маршрутизаторите в AS2 ще бъде изпратено съобщение “unreachable” (а не към AS1), защото 172.16.192.0/18 няма да е в маршрутната таблица. Извод: За намаляване на редовете в маршрутната таблица, да прилагаме: Агрегация без сумаризация Сигурността на BGP сесиите На глобално ниво. Ние не можем да предвидим маршрутите, които ни се подават, дали са точно те. Идват отдалеч. Известни са случаи на подвеждане (Напр., Китай, Пакистан). Решение: Resource Public Key Infrastructure (RPKI). RPKI От 1.01.11 RIR би трябвало да добавят слой на криптиране, така че ISPs и др. да могат да доказват, че са оторизирани да маршрутизират трафик за дадена AS. Но засега само APNIC ще го правят. Има и скептицизъм, преработка на софтуер и др. игурност. BGP и TCP. На локално ниво: **IPSec AH** (*в рамките на AS5421*) В BGP - удостоверяване чрез парола. Но след установяване на TCP сесия с всичките й предимства и недостатъци. С помощта на IPSec – Authentication Header - защитата на най-ниското възможно ниво - OSI L3. IPSec AH защита Удостоверяваме страните в една TCP сесия с максимално ниво на сигурност - проверка на електронен подпис. Така се защитаваме, например, от IP spoofing.

**18. Задръстване и управление на потоците в мрежа** Задръстване Когато броят на изпратените пакети в мрежата надвиши капацитета на мрежата се получава задръстване (congestion). Задръстването може да се причини от няколко фактора. Ако в даден маршрутизатор започват да постъпват пакети от три или четири линии и всички те трябва да се изпратят по една и съща линия ще се образува опашка. При недостатъчно памет ще се изгубят пакети. Бавни процесори също могат да причинят задръстване. Ако маршрутизаторите не могат да обработват достатъчно бързо пристигащите пакети отново ще се образуват опашки, въпреки че линиите не използват докрай своя капацитет. Управление на натоварванетo vs. управление на потоците Разликата между управление на натоварванетo и управление на потоците е: - управлението на натоварването е свързано с това цялата мрежа да може да се справи с определен трафик - то е на глобално ниво, засяга всички възли в мрежата, съхраняването и препращането на пакетите в маршрутизаторите и всички останали фактори, които намаляват капацитета на мрежата. Управление на натоварването vs. управление на потока - управлението на потоците е свързано с двупосочния трафик между даден източник и приемник. То е свързано с това бърз източник да не изпраща продължително данни, по- бързо отколкото приемника може да ги обработи. натоварване vs. потоци Управлението на потока обикновено се осъществява чрез обратна връзка от приемника към източника, по която се съобщава на източника за състоянието на приемника. То се осъществява най-вече на транспортно ниво. Недостиг на буфери Броят на буферите в един възел на мрежата е ограничен. Пристигнал пакет постъпва във входен буфер, след това преминава в изходен буфер, след това се изпраща потвърждение към източника. Нека възелът има *K* изходни буфера, *m* е максималният брой буфери на една изходна линия, *s* е броят на изходните линии. Ако *K* = *m*, тогава е възможно всички изходни буфери да се ангажират с една натоварена линия, което ще доведе до задръстване. Оптимално натоварване на възела се постига при m = *K* /(koren ot s) С една изходна линия може също да се свърже минимален брой буфери, които не могат да се ползват от други линии. Отстраняване на пакети (drop) Друга стратегия за избягване на задръстването е маршрутизаторите да отстраняват (drop) пакети по желание. Най-просто е да се отстраняват произволни пакети. Друга възможност е пакети да се отстраняват в зависимост от вида на предаваната информация. При трансфер на файлове има смисъл да се отстраняват по-новите пакети, тъй като отстраняването на стар пакет може да доведе до повторно предаване на следващите го пакети, ако получателят приема пакети само с поредни номера. При трансфер на мултимедия, по-новите пакети са по- важни от по-старите, тъй като се цели минимално закъснение. Ограничаване на пакетите в мрежата Трета стратегия е ограничаването на общия брой пакети в мрежата. За целта в мрежата се пускат служебни пакети, които представляват разрешения за изпращане на пакет.За да изпрати пакет един хост, прилежащият му маршрутизатор трябва да улови разрешение. Обратно, когато един хост получи пакет, разрешението се възстановява. По този начин броят на пакетите в мрежата се контролира от броя на разрешенията. Този алгоритъм е сложен за поддръжка. Регулиране на входа Последната стратегия е регулиране на входа при задръстване. Във всеки маршрутизатор се следи натовареността на изходните линии. Когато се надвиши някакъв праг на натовареност линията се слага в състояние на предупреждение. Когато линията е в такова състояние, в обратна посока се изпращат специални задръстващи пакети. Чрез тях изпращача разбира, че трябва да намали скоростта на предаване. В крайна сметка задръстващите пакети достигат до хоста, който генерира пакетите, довели до задръстването и го уведомяват да намали темпото на предаване. Проблем е, че регулирането на входа при задръстване не се прави на транспортно ниво, където е известен пътя между хостовете, и затова е трудно задръстващите пакети да намерят пътя си до хоста-източник. Deadlock Друг голям проблем при управление на натоварването на мрежите е проблемът **deadlock**. Да разгледаме пример с три възела *A*, *B* и *C*. Буферите и на трите възела са пълни. *A* има пакети за предаване към *B*, *B* има за *C*, a *C* има за *A*. По този начин трите възела са блокирани. Кръгът от блокирани възли може да е много по- голям. Проблемът е, че трудно се установява наличието на deadlock в мрежата. Поради това deadlock трябва да се предотвратява. Управление на потока на канално ниво (Flow Control) Друг проблем, който е свързан с управлението на обмена на канално ниво е източникът да изпраща кадри по-бързо, отколкото те могат да бъдат приети от получателя. Този проблем се решава чрез буфериране на кадрите (нареждане на опашка), но при запълване на опашката пристигналите кадри се губят. Управление на потока на канално ниво Въвеждат се механизми за управление на потока от кадри, който осигурява обратна информация на източника за темпа на предаване.Обикновено механизмите по управление на обмена се изпълняват в транспортния слой над по-големи информационни единици, обхващащи последователност от сегменти. Пример за Flow Control на канално ниво е Frame Relay (NBMA). Frame Relay. Flow Control Frame Relay комутаторите създават виртуални канали за свързване на отдалечени LANs или отделни машини. Нарича се Non Broadcast Multiple Access, защото за потребителите е „облак“ - MA, но под него са комутаторите свързани „точка-точка“ в някаква топология - NB.Евтина алтернатива на наетите линии (leased lines). Беше популярна преди появата на MAN. В заглавието на кадъра (frame) има и следните битове, решаващи проблема със задръстванията (congestion control): **FECN**=Forward Explicit Congestion Notification bit **BECN**=Backward Explicit Congestion Notification bit **DE**=Discard Eligibility bit Frame Relay мрежата е с опростен протокол, липсва flow-control връзка по връзка. Затова производителността на мрежата като цяло е податлива на „скокове“ в натоварването от някои услуги - bursts. Затова се нуждаят от механизми за congestion control – управление на задръстванията. Congestion control включва следните елементи: Admission Control. Мрежата решава дали да приеме нова заявка за връзка в зависимост от описания в заявката трафик (traffic descriptor), сравнявайки го с остатъчния капацитет. Traffic descriptor се състои от три компонента: - Committed Information Rate (CIR). Средната скорост (в bit/s), при която мрежата гарантира трансфера на информационни единици по време на интервал T: T = Bc/CIR. - Committed Burst Size (Bc). Максималният брой предадени информационни единици през интервала T. - Excess Burst Size (BE). Максималният брой непотвърдени информационни единици (в bits), които мрежата ще се опита да пренесе по време на интервала. След като мрежата е установила връзката, граничният възел трябва да следи трафика, така че да не се надвишават зададените стойности. Frame relay позволява мрежата да налага информационната скорост за крайните потребители и да изхвърля информация, когато се надвиши заявената скорост. Политиката на избягване на задръстванията включва непосредствено уведомяване. Така че да се постигне определено качество на услугата Quality of Service (QOS). За това служат специалните битове за контрол на задръстванията: FECN и BECN.FECN - Forward Explicit Congestion Notification. Установява се в 1, когато се наблюдава задръстване в посоката на предаване на кадъра (frame), т.е информира **дестинацията,** че има задръстване. BECN - Backwards Explicit Congestion Notification. Установява се в 1, когато се наблюдава задръстване в посоката, обратна на предаване на кадъра (frame), т.е информира **източника,** че има задръстване.

19. Транспортно ниво **Процедури за съединенията** Общи положения Най-общата цел на транспортното ниво е да осигури ефективни и надеждни услуги на своите потребители - процеси в приложното ниво. Транспортното ниво трябва да прехвърля произволно дълги съобщения между два крайни абоната. За тази цел използва услугите на по-долното мрежово ниво. Транспортното ниво предлага два вида обслужване – със съединение (надеждни) и без съединение (ненадеждни). Протоколи със съединение При протоколите със съединение имаме три фази – установяване на съединение, трансфер на данни и закриване на съединението. Транспортното ниво трябва да предоставя интерфейс на приложното ниво. Примерен интерфейс за надеждно обслужване се състои от следните примитиви LISTEN, CONNECT, SEND, RECEIVE, DISCONNECT. Да предположим, че имаме приложение с един сървър и много отдалечени клиенти. Установяване на връзката Първоначално сървърът изпълнява примитива LISTEN, който го блокира докато не се появи клиент. Когато клиентът иска да се свърже със сървъра, той изпълнява примитива CONNECT. Това изпълнение блокира клиента и изпраща пакет за установяване на връзка към сървъра. Ако сървъра е в състояние на слушане (блокиран след изпълнение на LISTEN), той изпраща обратно пакет за потвърждение за създаване на връзка, което отблокира клиента и връзката е създадена. Обмен на данни След установяване на връзката сървъра и клиента могат да обменят данни, като например всеки от тях изпълнява RECEIVE за изчакване на другия да изпълни SEND. Докато двете страни могат да спазват реда, в който трябва да предават данни тази схема работи добре. Всеки изпратен пакет данни трябва да се потвърждава обратно към изпращача. Когато връзката вече не е нужна, тя трябва да се прекрати за да се освободят заетите от нея ресурси. Освобождаване на връзката Освобождаването на връзката има два варианта – симетричен и асиметричен. При асиметричният вариант, само клиентът може да прекрати връзката чрез изпълнение на примитива DISCONNECT, който води до изпращане на пакет за прекратяване на връзката до сървъра. При симетричния вариант и двете страни трябва да затворят връзката, т.е. да изпълнят DISCONNECT независимо един от друг. Когато едната страна изпълни DISCONNECT, това означава че тя повече няма да предава данни, но не означава че няма да приема данни. Транспортни протоколи Услугите, предоставяни от транспортното ниво, се реализират от протокола на транспортно ниво, който действа между транспортното ниво на двата крайни абоната. Всъщност транспортните протоколи си приличат с протоколите на канално ниво. И двата типа протоколи се грижат за отстраняването на грешки, за последователно предаване на данните, за управление на потока и др. Съществуват и много разлики. При протоколите от каналното ниво двете точки взаимодействат помежду си през физически канал. При транспортното ниво този физически канал се заменя от комуникационна подмрежа до целия Интернет. Друга съществена разлика е, че при изпращане на кадър от каналното ниво той или пристига или се изгубва, докато пакетите при транспортното ниво могат да престояват в маршрутизаторите, да обикалят цялата подмрежа, докато достигнат местоназначението. Портове и сокети Когато един процес от приложното ниво иска да се свърже с друг процес, той трябва да може да го идентифицира. Методът е да се присвоят адреси на процесите, които слушат за създаване на връзка. В Internet тези адреси се наричат **портове** и са 16-битови номера. Комбинацията от IP адрес и порт се нарича socket и той уникално идентифицира процес, който се изпълнява на някакъв хост. Установяване на съединение Установяването на съединение изглежда просто, но не е така. На пръв поглед е достатъчно едното транспортно ниво да изпрати пакет за създаване на връзка, а другото да отговори с пакет за приемане на връзката. Проблемите идват от факта, че пакетите могат да се изгубят, да се забави предаването им или да се дублират. Трикратно договаряне За преодоляване на тези проблеми при установяване на връзка се използва процедура за **трикратно договаряне** (three-way handshake). Да предположим, че крайните абонати A и B искат да установят връзка. Изпълняват се следните три стъпки: 1. Изпращане на заявка **A** изпраща заявка, че иска да установи съединение – указва размера на буферите, брой на буферите, скоростта на предаване и др. Тази заявка е специално служебно съобщение на транспортно ниво. За мрежовото ниво заявката е обикновен пакет. Съобщението може да стигне до B, но може и да не стигне. За целта A си включва таймер и ако той изтече преди да се получи отговор от B, A изпраща заявката наново. След три неуспешни опита A решава, че B е недостъпен и се отказва; 2. Отговор B отговаря на A, като включва в oтговора собствени параметри за връзката 3. Потвърждение A потвърждава съединението на B. Това потвърждение е необходимо, тъй като B трябва да знае, че A е получил отговора. След успешно получаване на потвърждаване A и B заделят заявените ресурси. Прекратяване на връзката Прекратяването на връзката също се извършва с трикратно договаряне.

**20. Транспортни протоколи** TCP и UDP TCP и UDP Двата основни протокола на транспортния слой в TCP/IP модела са Transmission Control Protocol (TCP) и User Datagram Protocol (UDP). И двата управляват комуникациите между приложения, работещи на компютри в Мрежата и са от типа “край до край” (end to end).Разликите между двта са във функциите, които реализират. UDP е по-опростен, осигурява ненадеждно обслужване с неустановена връзка (connectionless), дефиниран в RFC 768. Предимство е ниското закъснение. Протоколните единици в UDP се наричат дейтаграми, за които подобно на IP пакетите се полагат максимални усилия за доставяне - "best effort". Типични приложения на UDP са: - Domain Name System (DNS) - Video Streaming - Voice (Video) over IP (VoIP); Video over IP - мониторинг и управление на мрежите (SNMP) - опростен пренос на двоични файлове (Trivial FTP). TCP е протокол, който предоставя надеждно обслужване с установена връзка (connection-oriented), дефиниран в RFC 793. TCP внася допълнително закъснение заради функциите по надеждност, спазване на реда на подаване на единиците с данни (сегменти) и управление на потока. Всеки TCP сегмент има 20 байта служебна информация, с които опакова приложните данни, докато при UDP дейтаграмите те са само 8 байта. Типични приложения на TCP са: - Web браузъри и сървъри - E-mail- сигурен обмен на файлове (FTP) Адресиране с портове. Идентифициране на “разговорите” TCP и UDP базираните услуги следят комуникациите между различни приложения в Мрежата. За да разпознаят сегментите и дейтаграмите на всяко приложение, и TCP сегментите, и UDP дейтаграмите имат полета в заглавната част, които уникално идентифицират тези приложения - номерата на портовете. По-конкретно, порта-източник и порта- местоназначение: source port и destination port. source port е номера на комуникацията, свързана с приложението – инициатор на “разговора” (сесия). destination port е номера на комуникацията, свързана с приложението – дестинация, работещо върху отдалечения хост. destination port Номерата на портове се присвояват по различни начини, в зависимост от това дали съобщението е заявка или отговор. Процесите на сървър са със статични номера, а клиентите динамично избират номер на порт при всяка сесия. Клиент изпраща заявка до сървър: destination port е номер на порт, присвоен на процеса (демона) на услугата, стратирана на отдалечения хост. Клиентският софтуер трябва да знае номера на този порт. Той се конфигурира по подразбиране или ръчно. Например, web браузър прави заявка към web сървър. Използва TCP и порт 80, ако не е дефинирано нещо друго. TCP port 80 е по подразбиране присвоен на web сървърите. Повечето известни приложения имат номера на портове по подразбиране. source port source port в заглавието на сегмента/дейтаграмата, съдържащ заявката на клиента, е произволно генериран от номера на портове, по-големи от 1023. source port играе ролята на обратен адрес за приложението, заявяващо услугата. Комбинацията от номера на порт на транспортния слой и IP адреса на мрежовия (IP:port No) идентифицира конкретен процес, работещ а даден хост. Тази комбинация се нарича сокет (socket). Двойката (IP:port No) на източник и местоназначение идентифицира конкретна сесия между два хоста. Например, HTTP заявка за web страница, изпратена към web сървър (port 80) с IPv4 адрес 192.168.1.20 е насочена към сокет 192.168.1.20:80. Ако web браузъра е на хост с IP: 192.168.100.48 и динамично му е присвоен порт 49152, сокет, където трябва да бъде “свалена” web страницата ще е 192.168.100.48:49152. Видове портове IANA (iana.org) се занимава и с присвояване на портове. Различните видове номера: Добре известни - Well Known Ports (0 - 1023). Резервирани за популярни услуги и приложения – HTTP, POP3/SMTP, Telnet и др. Регистрирани - Registered Ports (1024 - 49151). Тези номера се присвояват на потребиъелски процеси и приложения.Ако не се използват в момента, могат да се използват динамично от клиентските процеси. Динамични или частни портове (49152 - 65535). Известни и като Ephemeral (краткотрайни) портове. Обикновено се присвояват динамично на клиентски приложения, иницииращи сесия. Услугите не използват такива портове (с изключение на peer-to-peer file sharing програми). Номера, използвани и в TCP, и в UDP. За постигане на по-висока производителност някои приложения в даден момент се опират на TCP, в друг - на UDP. Например, ниското закъснение, внасяно от UDP, позволява на DNS да обслужва бързо много клиентски заявки. Но понякога зпращането на заявената информация може да изисква надеждността на TCP. Т.е DNS използва известния порт 53 и с двата протокола. Списък с номерата на портовете можете да намерите на: <http://www.iana.org/assignments/port-numbers> Сегментиране и възстановяване TCP и UDP се справят със сегментирането по различен начин. В TCP в заглавната част на всеки сегмент се съдържа пореден номер (sequence number). Благодарение на него в Транспортния слой на дестинацията се възстановява реда, по който сегментите са били предадени. Услугите в UDP също следят сесиите между приложенията, но не и реда, по който се предава информацията, и поддържането на връзката.В UDP заглавието липсва последователен номер (sequence number). UDP е с по-опростена структура и внася по-малко излишна а потребителя служебна информация (overhead) от TCP. Затова осигурява по-бърз пренос на данните. Но приложенията, които се базират на UDP, трябва да се съобразяват с факта, че е възможно данните в различен ред от този, по който са били предадени. Transmission Control Protocol Transmission Control Protocol (**TCP**) осигурява надеждно, подредено доставяне на потока от байтове от програма на един компютър до програма на друг компютър. TCP следи за размера на съобщенията, скоростта на обмен и натоварването на мрежовия трафик. Приложимост на TCP TCP се ползва от най-популярните Internet приложения - WWW, E-mail, FTP, SSH, Telnet и някои групови медийни приложения (streaming media). TCP е оптимизиран за доставяне на съобщения без грешка в съдържанието, но не и за гарантирано време на доставяне. TCP понякога внася големи закъснения (в порядъка на секунди) заради подреждането на пристигналите пакети и повторни предавания. Затова не е подходящ за приложения в реално време като Voice over IP (VoIP) и Video over IP. За тях се препоръчва Real-time Transport Protocol (RTP), който работи върху User Datagram Protocol (UDP). TCP и IP Между “подател” и “получател” в Интернет се разменят *съобщения*. Докато IP се грижи за действителното доставяне на данните, TCP следи отделните диници от данни - *сегменти*, на които е разбито съобщението с цел ефективна маршрутизация. Например, HTML файл се праща от Web сървър. TCP софтуерът на този сървър разделя последователността от байтове във файла на сегменти, които препраща поотделно към IP софтуера – мрежовия слой. Мрежовият слой опакова всеки TCP сегмент в IP пакет. TCP и IP Всички IP пакети, които произхождат от горния HTML файл са с един и същ адрес на получател, но те могат да минат по различни пътища през Мрежата. Клиентската програма на компютъра-получател, по-точно TCP софтуера (транспортния слой), възстановява оригиналното подреждане на сегментите, като гарантира, че са получени без грешка, и ги препраща към приложната програма. TCP процеси на сървъра На една и същ машина, сървър, nе е възможно да има две услуги, присвоени на един и същ номер на порт. Структура на TCP сегмента TCP сегментът се състои от две части: заглавие (header) и данни (data). Заглавната част на TCP сегмента се състои от 11 полета, от които 10 са задължителни. 11-то е "options". Source port (16 бита) – номер на изпращащ порт Destination port (16 бита) – номер на получаващ порт Sequence number (32 бита) – двуяка роля: - ако флаг SYN е вдигнат, това е първоначалният номер. Последователният номер на първия байт ще бъде именно този sequence number + 1. - ако флаг SYN не е вдигнат, това е sequence number на първия байт с данни. Полето Acknowledgement number е номерът на първия байт данни, който се очаква да се получи със следващия сегмент, изпратен от другия край на TCP връзката. Например, при успешно получаване на сегмент с размер на полето данни 500 байта и пореден номер на началния му байт n, към източника на този сегмент се изпраща TCP сегмент, в който потвърждението е с номер n+501. Полето TCP header length е 4-битово и определя дължината на заглавната част на TCP сегмента в 32- битови думи. То е задължително, тъй като полето за опции е с променлива дължина. Фактически с това поле се определя началото на полето Data в рамките на TCP сегмента. Заглавната част на TCP сегмента съдържа и 6 еднобитови флага: URG – валиден е указателят за спешни данни (Urgent pointer). Установяването на този флаг означава, че трябва да се преустанови обработката на получените данни, докато не се обработят байтовете, към които сочи указателят за спешни данни; ACK – валиден е номерът на потвърждение, записан в полето Acknowledgement number на заглавната част; PSH – при активирането на този флаг, програмните модули управляващи транспортния слой на източника и на приемника трябва да изпратят незабавно наличните данни колкото е възможно по-бързо към техния получател, т.е. източникът не изчаква да се съберат данните за образуване на пълен сегмент с избрания размер и съответно получателят не чака запълването на приемния буфер; RST – сегмент, в който е установен този флаг, служи за прекратяване на TCP връзката. Използва се в случаите, когато връзката е нарушена (например, поради повреда в хоста) или когато се отхвърля невалиден егмент или се отказва опит за установяване на връзка; SYN – сегмент с установен флаг SYN се използва при установяване на TCP връзка и за изпращане на началния номер, от който ще бъдат номерирани байтовете на изходящия информационен поток; FIN – сегмент, в който е установен този флаг, означава, че изпращачът прекратява предаването на данни. Поради двупосочния характер на информационния обмен това не означава, че TCP връзката е прекратена. Полето Window size определя темпа на информационния обмен от гледна точка на получателя на информационния поток. Стойността на прозореца указва на отсрещната страна колко байта могат да бъдат изпратени и съответно приети без препълване на входия буфер след последния потвърден номер на байт. При получаване на данни, размерът на прозореца намалява. Ако той стане равен на 0, изпращачът трябва да престане да предава данни. След като данните се обработят, получателят увеличава размера на своя прозорец, което означава, че е готов да получава нови данни. Полето Urgent pointer се използва да укаже позицията на първия байт на спешните данни спрямо началото на полето данни. Полето Checksum се изчислява върху целия TCP сегмент. При неговото изчисляване участват и някои полета от заглавната част на IP дейтаграмата, в която е опакован сегмента. Полето Options на заглавната част на TCP сегмента е предназначено да предостави допълнителни възможности за управление на обмена, които не се осигуряват от останалите полета на заглавието. Най-важната възможност е указване на максимална дължина на сегмента. Всеки хост указва своята максимална дължина на сегмента и за осъществяване на обмена се приема по-малката от двете. Ако mаксималната дължина на сегмента не се договори се приема по подразбиране, че нейната стойност е 556 байта, което е допустимо за всички интернет хостове. Установяване и терминиране на TCP съединение Когато два хоста комуникират по TCP, първо трябва да се оформи съединение, преди да започне обмена на данни. След приключване на комуникацията сесиите се затварят - терминират. Всичките тези процедури реализират функциите по надеждност в TCP. За установяване на връзка хостовете изпълняват трипосочна процедура – “ръкостискане” (three-way handshake). Контролните битове в TCP заглавието регистрират прогреса и състоянието на връзката. Three-way handshake прави: • Установява, че отсрещното устройство съществува в мрежата; • Уверява се, че отсрещното устройство има активна услуга и приема заявки на отдалечения порт, който инициатора смята да използва за сесията • Информира отсрещното устройство, че клиент източник има намерение да установи сесия с този номер на порт 3 стъпки на TCP съединение При TCP съединенията хостът-клиент инициира сесията към сървъра. За да си изясним как работи “three-way handshake”, нека видим какви стойности на параметри си разменят двете старани. Трите стъпки в установяване на TCP съединение са: 1. Клиентът-инициатор изпраща сегмент SYN, съдържащ начална стойност на последователността SEQ, и представляващ заявка за начало на сесия. 2. В отговор сървърът изпраща сегмент, съдържащ стойност за потвърждение ACK (SEQ + 1). Освен това - собствената си синххронизираща стойност на последователност SYN SEQ, която е по- голяма от получения и потвърден номер ACK – следващия очакван bайт. 3. Клиентът-инициатор отговаря със стойност ACK, равна на (получена SEQ + 1). С това съединението е установено. Терминиране на TCP сесия По 4-стъпкова процедура се разменят флагове за терминиране на TCP съединение. TCP. Потвърждение и прозорци. Стойностите SEQ и ACK в заглавието на сегмента съвместно служат за потвърждение на получените байтове с данни. SEQ е относителния брой байтове, които са предадени в дадената сесия + 1 (номера на първия байт с данни в дадения сегмент). TCP използва числото ACK в сегментите, които се изпращат обратно на източника, за да покаже кой е следващия байт, който приемника очаква да получи в настоящата сесия. Това се нарича очакваното потвърждение. Източникът на сесията е информиран, че дестинацията е получила всички байтове в този поток от данни с изключение на байта под номер, равен на номера, съдържащ се в потвърждението. Очаква се хоста-инициатор да изпрати сегмент със SEQ = (ACK). Запомнете. Всъщност всяко съединение представлява две еднопосочни сесии. Стойностите SEQ и ACK се разменят и в двете посоки. TCP. Потвърждение и прозорци. Пример. В примера на фигурата на по-следващия слайд левият хост изпраща данни към десния. По-точно, изпраща сегмент с 10 байта данни и SEQ = 1. Хостът отдясно получава сегмента и вижда, че SEQ = 1 и има 10 байта с данни. Хостът отдясно връща обратно сегмент към левия хост, за да потвърди получаването на данните. В този сегмент ACK = 11, с което показва, че очаква да получи следващ байт с данни под номер 11. Забележете! Стойността ACK левия хост остава 1, за да покаже, че сегментът е част от вървящата сесия и стойността в полето Acknowledgment Number е валидна. След като левият хост е получил потвърждение, вече може да продължи текущата сесия, като изпрати следващия сегмент с данни, започващи от байт 11. Ако левият хост в този пример трябва да чака потвърждение на всеки 10 байта, закъснението ще е много голямо. За да се намали служебния трафик от тези потвърждения, възможно е да се изпратят множество сегменти, които да се потвърдят с едно единствено TCP в обратната посока. В това потвърждение се съдържа число, показващо общия брой на байтовете, получени в тази сесия. Например, стартираме със SEQ = 2000. Получени са 10 сегмента с по 1000 байта всеки. На източника (левия хост) ще бъде върнат ACK = 12000. Количеството данни, които източникът може да изпрати, преди да получи потвърждение, се нарича размер на прозореца. Window Size е поле в заглавието на сегмента, което улеснява улавянето на загубени данни и управлението на потока. TCP. Повторно предаване. Колкото и добре да е проектирана дадена мрежа, не може без загуби на данни. TCP осигурява методи за управление на тези загуби на сегменти. Един от тях е механизмът за повторно предаване на сегменти с непотвърдени данни. TCP услугата в хоста-приемник потвърждава само данни, състоящи се от непрекъсната последователност от данни. Ако липсват един или повече сегменти, потвърждават се само данните в сегментите, които попълват плътно потока. Например, получени са сегменти със SEQ = 1500 - 3000 и 3400 – 3500. Тогава ACK = 3001. Това е така, защото има сегменти със SEQ = 3001 – 3399, които не са получени. TCP в хоста-източник не е получил потвърждение след предварително зададено време. Тогава той ще се върне към последния ACK, който е получен, и ще предаде повторно данните от тази точка нататък. Процесът на повторно предаване не е дефиниран в RFC, зависи от конкретната реализация на TCP. В типичната TCP реализация хост предава сегмент, поставя копие от него в опашка за препредаване и стартира таймер. След получаване на потвърждение сегментът се изтрива от опашката. Ако след нулиране на таймера не се получи потвърждение, сегментът се предава отново. Днес хостовете могат да поддържат и опцията “Селективно потвърждение”. В такъв случай дестинацията би потвърдила байтове в непоследователни сегменти, без да е необходимо препредаване на липсващи данни. TCP. Управление на потока и задръстванията. TCP има и механизми за управление на потока (flow control). Flow control синхронизира скоростите на потока от данни между двете услуги в сесията. Когато източникът е информиран, че оределено количество данни в сегментите е получено, тогава може да продължи с изпращане на повече данни за дадената сесия. Полето Window Size определя количеството данни, което може да бъде предадено, преди да бъде получено потвърждение. Първоначалният размер на прозореца се определя в началото на сесията чрез three-way handshake. Механизмът за обратна връзка в TCP нагласява ефективната скорост на предаване на данните към максималния поток, който мрежата и устройството-получател могат да поддържат без загуби и без повторни предавания. В примера на следващия слайд първоначалният размер на прозореца е 3000 байта. След предаването на тези 3000 байта изпращачът очаква потвърждение, преди да продължи със следващи сегменти. След получаването му може да предаде още 3000 байта. В периоди, когато мрежата е задръстена или на получателя му липсват ресурси, закъснението може да се увеличи. И ефективната скорост на предаване на данните за тази сесия да намалее. Така се справяме с проблема с недостиг на ресурси. TCP. Редуциране на загубите. Друг начин за контролиране на потока от данни е да определяме размера на прозореца динамично. При недостиг на мрежови ресурси TCP редуцира размера на прозореца, с което намалява и скоростта на предаване. Хостът-получател в TCP сесията изпраща към подателя стойност на размера на прозореца, която показва броя на байтовете, които е в състояние да получи. Както се вижда на фигурата в по-следващия слайд, имаме загуба на един сегмент. Получателят променя размера на прозореца от 3000 на 1500. След периоди без загуби на данни или липса на ресурси получателят ще започне да увеличава прозореца. Това ще вдигне ефективната скорост и ще продължи, докато се появят загуби и трябва да се намали прозореца. Динамичното увеличаване и намаляване на прозореца е непрекъснат процес в TCP, който определя оптималния размер във всеки един момент за дадена сесия. Подробности за справяне със задръстванията в TCP има в RFC 2581. UDP UDP е по-опростен транспортен протокол с неустановена връзка. Това не означава, че приложенията, които се базират на UDP, са непременно ненадеждни. Функциите по надеждност се осъществяват другаде. Основни приложни протоколи, които “стъпват” на UDP са: Domain Name System (DNS) - Simple Network Management Protocol (SNMP) - Dynamic Host Configuration Protocol (DHCP) - Routing Information Protocol (RIP) - Trivial File Transfer Protocol (TFTP) - онлайн игри Онлайн игрите или VoIP могат да понесат някои загуби на данни. Но не и закъсненията, които внася TCP. Други приложения като DNS или TFTP ще повторят заявката, ако не получат отговор. И не им трябват гаранциите на TCP. UDP. Възстановяване на дейтаграми. UDP е connectionless и сесии не се установяват като в TCP. UDP е по-скоро транзакционен протокол. Т.е, ако приложението има данни за предаване, то ги предава. Много UDP-базирани приложения изпращат малки количества данни, които се побират в един сегмент - дейтаграма. Но някои изпраща по-големи количества, които се разделят на множество сегменти. При изпращане на множеството дейтаграми от едно приложение те могат да поемат различни пътища в мрежата и да пристигнат в разбъркан ред. UDP не следи последователността на дейтаграмите при приемане като TCP. Ако тя е от значение, за това се грижи приложната програма. UDP. Сървърски процеси и заявки. И на UDP-базирани сървърски приложения се присвояват Well Known или Registered портове. UDP. Клиентски процеси. И тук като в TCP клиентското приложение изпраща заявка към сървъра. Клиентският UDP процес избира на случаен принцип номер на порт от свободните. Случайният избор на порт помага за повишаване на сигурността. Номерът няма да е предварително известен на злосторника. В UDP не се създават сесии. След като данните са готови и портовете са идентифицирани, UDP формира дейтаграма и и я подава към мрежовия слой за изпращане по мрежата. **21. DNS система за iменуване. Процес на резолвинг на имената.** Защо ни трябва Чрез IP адресите се осъществява адресирането на дейтаграмите, които носят в себе си данните. Неудобното е, че те са числа и трудно се запомнят. Затова се въвежда система за именуване – **DNS**. Domain Name System Domain Name System (DNS) е йерархична разпределена база от данни. Тя съхранява информация за съотвтствието между Internet хост имена и IP адреси и обратно,информация за маршрутизиране на ел. поща и др. данни, използвани от Internet приложения. Клиентите търсят информация в DNS, извиквайки ***resolver library***, която изпраща заявки до един от сървърите за имена (***name servers***) и интерпретира отговорите. BIND софтуерът съдържа сървър за имена **named**, и две библиотеки - *resolver libraries*: **liblwres** и **libbind**. **ISC BIND** BIND (Berkeley Internet Name Domain) е реализация на DNS протоколите и осигурява отворена система за редистрибуция на основните компоненти на Domain Name System: - Domain Name System server (named); - Domain Name System resolver library; - средства за верифициране на операциите на DNS server. Домейни и имена на домейни Данните, съхранени в DNS са *domain names,* организирани в дървовидна структура. Всеки възел в дървото се нарича *domain и му се дава етикет*. Името на домейна във възела е поредица от етикетите, показващи пътя от възела до корена (*root)*. В писмена форма се представя като низ от етикети, от дясно наляво, разделени с точки. Домейните представляват области oт имена. Домейните са от първо, второ и трето ниво. *(Ако не се брои root.)* Няма пречки да има домейни от четвърто ниво, но те почти не се използват. Основният домейн е така нареченият **root** домейн. Той няма име и е един единствен. Представя се с точка. Домейни и поддомейни Под него се нареждат домейните от първо ниво, **top-level domain** (**TLD**). Управлението на TLDs е делегирано на различни организации от страна на ICANN, която менижира IANA, и е отговорна за DNS root зоната. Най-често използвани TLDs са: generic top-level domains (gTLD) – отворени за регистрация за всеки по света, например: com, net, org, biz и др. В началото всички те са в САЩ, но после в тях влизат още много имена на обекти извън САЩ, те нарастват твърде много. Затова се въвежда друга голяма група от домейни на първо ниво, свързани с географското разположение по държави – uk, de, bg др. Това са country-code top-level domains (ccTLD), показващи принадлежност към държава. Състоят се само от две букви. В повечето случаи съвпадат с кода на страната по ISO 3166. infrastructure top-level domain: Има само една TLD - Address and Routing Parameter Area (ARPA). Управлява се от IANA и има отношение към обратния резолвинг. Цялостното име, което включва домейните и обекта се нарича **URL** (uniform resource locator). Пример за URL е <http://www.fmi.uni-sofia.bg> URL В това URL bg е името на домейна от първо ниво, uni-sofia е името на поддомейна на bg от второ ниво, fmi е името на домейна от трето ниво www е web-сървъра от домейна fmi, http е името на протокола по който клиента се свързва към съответния обект. Колкото са точките в едно URL, толковата са нивата на домейните без да се брои root. В URL точката на root се пропуска (подразбира се). Resolving DNS е йерархична именна система с три компонента – именно пространство (как се изграждат имената), resolver-и и именни сървъри (name servers). Resolver-те са абонатите в Internet, които знаят URL и искат да получат съответния IP адрес. Процесът на преобразуване се нарича resolving. Той се извършва от DNS протокола. DNS протокол DNS основно използва User Datagram Protocol (UDP) на порт 53 за обслужване на заявки. DNS заявките се състоят от една единствена UDP заявка от клиента, последвана от един единствен UDP отговор от сървъра. DNS протокол Transmission Control Protocol (TCP) се използва, когато в отговора се съдържат повече от 512 bytes или при трансфер на зони. Някои операционни системи като HP-UX използват TCP за всички заявки. Зони За по-лесно администриране пространството с имената е разделено на области, наречени зони (*zones)* Всяка зона започва от възел и се простира надолу до “листата” (leaf nodes) или до възли, където стартират други зони. Данните за всяка зона се съхраняват в сървър за имена (*name server)*, който отговаря на запитвания (queries) в рамките на зоната, използвайки *DNS протокол*. Данните, които са обвързани с всяко име на домейн, се съхраняват под формата на ресурсни записи, *resource records* (RRs). От особена важност е да се разбере разликата между *зона* и *домейн,* за да се вникне в същността на сървъра за имена. Зона е точката на делегиране на DNS дървото.Зоната се състои от тези последователни части от дървото на домейните, за които сървърът за имена има пълна информацияи върху която има власт. Състои се от всички имена на домейни, от дадена точка надолу по дървото с изключение на тези, които са делегирани на други зони. Точката на делегиране се маркира с един или повече записа: *NS records,* в родителската зона, които трябва да съвпадат с еквивалентни NS записи в корена на делегираната зона. Напр., да вземем домейна example.com, който включва имена като host.aaa.example.com и host.bbb.example.com. example.com зоната включва делегирания за зоните aaa.example.com и bbb.example.com.Една зона може да съответства точно на един единствен домейн, но може и да включва само част от домейна. Като останалата част от него да бъде делегирана на други сървъри за имена. Всяко име в DNS дървото е *domain*, даже ако е *terminal*, т.е няма *subdomains (поддомейни)*. Всеки *поддомейн е домейн* и всеки *домейн* с изключение на root (коренния) е също *поддомейн*. Терминологията не е интуитивна, за по-добро разбиране прочетете RFCs 1033, 1034 и 1035. **master** и **slave** зони Макар че BIND се нарича "domain name server", тъой се занимава предимно със зони. Декларациите **master** и **slave** във файла named.conf определят зони а не домейни. Ако питате някой друг сайт дали иска да бъде *slave* сървър на вашия *domain*, вие всъщност молите за **slave услуга** за някакъв набор от зони. Видове зони Master Сървърът чете данните за зоната директно от локалния диск (т.е от zone file) и е овластен да дава отговори за тази зона. Hint В тази зона се дефинират root-servers. Slave Зона slave е реплика на master зона и получава данни за тази зона чрез зонов трансфер. slave ще даде овластен отговор за зоната, само ако има валидни (не timed out) данни за зоната. Редът masters определя IP адрес/и на master сървър/и, с които slave контактува, за да refresh или update копие на зоната. Authoritative Name Servers Всяка зона се обслужва най-малко от един овластен сървър за имена (*authoritative name server)*, който държи всички данни за зоната. За по-висока надеждност се препоръчва зоната да има два или повече такива сървъри. В отговорите на *authoritative servers*, в пакета с отговора, е вдигнат бит "authoritative answer" (AA). Така по-лесно се диагностицират (debugging) DNS конфигурациите с инструменти като **dig**. Primary Master *authoritative server*, където се поддържа главното (master) копие на данните за зоната. Нарича се *primary master* сървър или просто *primary*. Той зарежда съдържанието на зоната от локален файл, редактиран ръчно или генериран от някакъв друг локален файл. Този файл се нарича *зонов* - *zone file* или *master file.*Slave Servers Другите authoritative servers, *slave* сървъри (известни още като *secondary*) зареждат съдържанието на зоната от друг сървър чрез процес на репликация - *zone transfer*. Обикновено данните се прехвърлят директно от primary master, но е възможно и от друг slave. Т.е, *slave server* може да действа като *master* за подчинен *slave server*. Caching Name Servers **resolver** библиотеките, които присъстват в повечето операционни системи, са *stub resolvers*, т.е те не са способни да изпълняват пълния процес на DNS резолюция, “говорейки” директно с authoritative servers. Те разчитат на локален сървър за имена, който да изпълнява резолюцията вместо тях. Такъв сървър се нарича “***recursive***” (рекурсивен) сървър за имена, защото изпълнява *рекурсивни търсения* за сметка на локалните клиенти. Caching (*recursive*) Servers За да се подобри производителността, рекурсивните сървъри кешират резултатите от търсенията, които са изпълнили. Процесите на рекурсия и кеширане са взаимно свързани, на термините *recursive server* и *caching server* често се гледа като на синоними. Перодът от време, за който един запис се държи в кеша, се контролира от Time To Live (TTL) полето в него. Caching Servers. Forwarding. Кеширащият сървър за имена не е необходимо да изпълнява сам пълното рекурсивно търсене. Вместо това той препраща (*forward)* някои или всички заявки, които не може да удовлетвори, от своя кеш към кеша на друг сървър за имена, който се определя като ***forwarder***. Многофункционални сървъри Сървърът за имена BIND може едновременно да бъде и **master** за някои зони, и **slave** за други зони, и кеширащ (рекурсивен) сървър за определен брой локални клиенти. Все пак, функциите на овластени (authoritative) услуги за имена и такива на caching/recursive са логически разделени. Затова е по-изгодно да работят на различни машини. Така ще се повиши надеждността и сигурността. Ресурсни записи SOA определя кой е първичният сървър и как се обработват данните към него. NS съдържа информация кои DNS сървъри са отговорни за този домейн. MX указва име на хост, готов да приема електронна поща в рамките на домейн. Адресните записи съдържат съответствие между име и IP-адрес. Имат следния формат: <hostname> A <IP address> Ресурсни записи В DNS е възможно създаването на прякори, т.е. няколко имена да отговарят на един и същ IP адрес. Това става с помощта на CNAME-записите, които имат следния формат: mail CNAME tiger proxy CNAME tiger tiger A 62.44.118.1 **Root** сървъри за имена Кореновият сървър за имена (**root nameserver**) е DNS сървър, който отговаря на запитвания относно имената в коренния домейн и отправя заявките към конкретни top-level domain (TLD), т.е към техните сървъри за имена. Всички имена в Internet завършват с точка **.** - напр., "**www.wikipedia.org.”** Но съвременният DNS софтуер не се нуждае от нея, когато се опитва да транслира домейн име в IP адрес. Празният низ след крайната точка се нарича коренов домейн (root domain), а всички останали (т.е. .com, .org, .net, и т.н.) се съдържат вътре в коренния (root). Когато компютър в Internet иска да открие съответствие (resolve) за домейн име, започва от дясно на ляво, запитвайки всеки name server поред относно елемента от ляво. **root nameservers** (отговарящи за домейна . ) знаят кои сървъри са отговорни за top-level домейните. Всеки такъв домейн (напр. .bg) има вой набор от сървъри, които от своя страна делегират към nameserver-те, отговарящи за отделните имена на домейни (като uni-sofia.bg), които пък отговарят на запитванията за IP адреси на поддомейни или хостове (напр. Poshta.fmi). Информацията не се променя често, затова се кешира, така че DNS *търсенията* към **root nameservers** са относително редки. Но в Internet има доста некоректно конфигурирани системи, които генерират трафик към root servers. Напр., заявки с източник адрес 0.0.0.0 (т.е където и да е, навсякъде) отиват натам. В момента има **13 root name servers**, като имената им са с формат *буква*.root-servers.net (*буква* е от A до M В момента C, F, I, J и K сървърите се намират на много места, на различни континенти, като за целта се използват anycast анонси, с което се осигурява разпределена услуга. Това предпазва и от DoS и DDoS атаки. Регистриране на име Регистрирането на име не е автоматично, а става чрез специална заявка към регистратор за съответния домейн или фирма, на която са делегирани съответни права за регистрация. За домейна .bg регистратор е register.bg. Резолвинг на имена За да се използва системата на URL-имената в клиента (resolver) трябва да има агент, който да може да работи с URL - началото на resolving процеса. Освен това в клиента трябва да има и малък кеш, в който да се съхранява информация за вече заявени и resolve-нати адреси за този клиент. Също така, клиентът трябва да разполага с адрес на DNS сървър, който отговаря за съответната област. Когато към агента се подаде URL за resolve-ане той първо проверява дали отговора не стои в кеша. Ако не, той изпраща заявка до DNS сървър. DNS сървърът може да формира три типа заявки – рекурсивна, итеративна или инверсна. • Една тежка процедура, която товари много ***root*** сървърите Рекурсивна заявка При рекурсивна заявка DNS сървърът има п прилежащ към него друг сървър за имена. Този сървър също може да има кеш, който евентуално да съдържа отговора. Сървърът може да съдържа отговора в своите зонални файлове. Ако и двата случая не са налице, но има конфигуриран друг сървър за имена, той ще изпрати заявката към него и т.н. В един момент някой сървър по описаната верига може да направи рекурсивната заявка в итеративна. Итеративна заявка При итеративната заявка сървър е в свободното Internet пространство. Той започва да раздробява съответното URL ипостъпково, съгласно структурата на URL започва resolve-то. Първо се изпраща заявка към root-сървъра, като се иска адреса на сървъра, който отговаря за TLD. След това се праща заявка към сървъра от първо ниво за адреса на сървъра, който отговаря за домейна от второ ниво, участващ в URL-то и т.н. Връщане на отговор Например, URL [www.fmi.uni-sofia.bg](http://www.fmi.uni-sofia.bg) След като една рекурсивна заявка е превърната в итеративна и итеративната заявка е изпълнена, полученият отговор се връща обратно по веригата на рекурсивната заявка и се стига обратно до resolve-ра, който слага получения отговор в кеша си. Инверсни заявки Инверсните заявки служат за обратен resolve – по IP адрес да се получи URL. В сървърите за имена има специални записи, предназначени за инверсни заявки: домейна *in-addr.arpa* и (Pointer) PTR записите. Йерархията на имената тук е спазена с помощта на специалния домейн “INADDR. ARPA”, разположен в резервирания .ARPA TLD (Address and Routing Parameter Area) “IN-ADDR” означава “INternet ADDRess”. За IPv6 reverse lookup домейнът е ip6.arpa in-addr.arpa имена in-addr.arpa имената се записват в ред, обратен на записа на IP адресите – от младши към старши или отляво надясно. Например, машина с IP адрес 10.1.2.3 ще има inaddr. arpa име 3.2.1.10.in-addr.arpa. Това име ще има PTR ресурсен запис: $ORIGIN 2.1.10.in-addr.arpa PTR foo.example.com. **$ORIGIN** се поставят само за пояснение, но не са задължителни. Classless reverse DNS zone "64.96.44.62.in-addr.arpa" { type slave; masters { 62.44.96.80 ; }; file "slaves/64.96.44.62.in-addr.arpa"; }; В миналото Internet регистраторите и ISPs алокираха октет- базирани IP адресни блокове от по 256 (Class C) или по-големи - класове B и A. С въвеждането на CIDR се алокират по-малки адресни блокове. RFC 2317 решава този проблем чрез делегиране на права за администриране: IPv6 reverse Обратният DNS резолвинг за IPv6 адреси използва домейна ip6.arpa. IPv6 адресите се представят като последователност от nibbles (полуоктети – 16- ни цифри) в обратен ред (както при Ipv4). Например, домейн за IPv6 address 2001:db8::567:89ab: b.a.9.8.7.6.5.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0. 8.b.d.0.1.0.0.2.ip6.arpa. rndc С помощта на програмата remote name daemon control (**rndc**) администраторът контролира работата на name сървъра. След всяка промяна в zone и/или reverse файл се изпълнява rndc reload (e.g.): [root@ns named]# vim uni-sofia.bg [root@ns named]#rndc reload uni-sofia.bg или [root@ns named]# vim 96.44.62.in-addr.arpa [root@ns named]#rndc reload 96.44.62.in-addr.arpa DNSSEC За борба срещу DNS измами като cache poisoning: електронно подписване на ресурсните записи в зоните с помощта на криптография с публичен ключ (RSA и DSA базирана); урежда съхраняването на извършените върху ресурсните записи електронни подписи под формата на допълнителни ресурсни записи в същата зона; дава възможност за проверка на извършените върху ресурсните записи електронни подписи от страна на рекурсивни/кеширащи сървъри за имена, с цел проверка на автентичността на записите. *DSA - Digital Signature Algorithm RSA - Ronald L.* ***R****ivest, Adi* ***S****hamir und Leonard* ***A****dleman.***23. Електронна поща в Интернет** Общи положения **Електронната поща** (**e-mail** или **email)** е метод за обмен на електронни съобщения. Едно съобщение се състои най-малко от съдържанието си, адрес на автора и адресите на един или повече получатели. Корените на днешната email са в Arpanet – стандарт за кодиране на съобщения - RFC 733. Преходът от Arpanet към Internet в началото на 1980-те добави постепенно новостите към основната услуга: - транспортния протокол Simple Mail Transfer Protocol (SMTP), RFC 821 през 1982 г. - ревизия на RFC 733 - RFC 822 - прикрепяния на мултимедия – от 1996 г. - от RFC 2045 до RFC 2049, известни като Multipurpose Internet Mail Extensions (MIME). Email се базират на модела с пълно буфериране (store-and-forward). Сървърът за електронна поща приема, препраща, доставя или съхранява съобщения за сметка на потребителите. Тяхна задача е само да се свържат към email инфраструктурата с помощта на компютрите си. Email се базират на модела с пълно буфериране (store-and-forward). Сървърът за електронна поща приема, препраща, доставя или съхранява съобщения за сметка на потребителите. Тяхна задача е само да се свържат към email инфраструктурата с помощта на компютрите си. Email се базират на модела с пълно буфериране (store-and-forward). Сървърът за електронна поща приема, препраща, доставя или съхранява съобщения за сметка на потребителите. Тяхна задача е само да се свържат към email инфраструктурата с помощта на компютрите си. Терминология Mail-box – файл или директория/и от файлове, където се съхраняват входящите съобщения. mail user agent (MUA) е приложна програма, стартирана от потребителя. Използва се за оформяне и изпращане на съобщения, както и за показване, сортиране като файлове и принтиране на получени в кутията съобщения. Такива са elm, mailx, mh, zmail, Mozilla Thunderbird, MS Outlook и др. Mail transfer agent (MTA) осъществява маршрутизацията на съобщенията, подадени от MUA, до получателя. Най-популярният MTA е **sendmail**. Delivery agent поставя съобщението в пощенската кутия на потребителя. Най- популярният DA е mail.local. Какво става 1. mail user agent (MUA) на Алис форматира съобщението в e-mail формат и с помощта на SMTP го изпраща към местния mail transfer agent (MTA) - smtp.a.org. 2. MTA гледа за крайния адрес, според SMTP протокола (а не в главата на съобщението) - bob@b.org. В e-mail адреса частта пред @ е локалната част, най-често потребителското име на получателя. Частта след @ е име на домейна. MTA по името на домейна определя пълното домейн име на пощенския сървър в DNS. 3. DNS сървърът за домейн b.org - ns.b.org, отговаря с MX записи, изброяващи пощенските сървъри в този домейн, в случая mx.b.org. 4. smtp.a.org изпраща съобщението до mx.b.org по SMTP, който го доставя до пощенската кутия (mailbox) на bob. 5. Bob натиска бутон "get mail" на своя MUA, с което изтегля съобщението с помощта на Post Office Protocol (POP3). Алтернативи на последователността • Alice може да няма MUA на компютъра си, а да се свърже към webmail услуга. • На компютъра на Alice може да е “качен” MTA, т.е да прескочи стъпка 1. • Bob има много начини да изтегли пощата си, например, по протокол Internet Message Access Protocol, като се логне на mx.b.org и си я чете директно от там, или и той с webmail. • В един домейн има няколко пощенски сървъра, така че те могат да продължат да приемат поща, даже когато главният е отпаднал. • За повишаване на сигурността и конфиденциалността е добре пощата да се криптира (OpenPGP, X.500 сертификати). Но това е друга тема. open mail relay MTAs, които приемат съобщения от произволни податели и полагат максимални усилия да ги препратят по посока към получателите. Такива MTAs се наричат open mail relay. В зората на Internet, когато мрежите не бяха надеждни, това усилие беше похвално. Да може съобщението все пак да достигне целта си през един или повече relay. Но от този механизъм се възползваха недобросъвестни “изпращачи” на спам и друга нерегламентирана поща. Затова днешните MTAs не са open mail relays и не приемат поща от open mail relays, която си чист спам. Това важи и за сървърите СУ. Използваме и отворен софтуер SpamAssassin. Формати Форматът на e-mail съобщенията е дефиниран в RFC 5322 и серия от RFC-та, RFC 2045 до RFC 2049, "Multipurpose Internet Mail Extensions" или MIME. e-mail съобщенията се състоят от два основни дяла, отделени с празен ред: Header (глава) Структурирано е от полета, обобщение (summary), подател (sender), получател (receiver) и др. Body (тяло) Самото съобщение като неструктуриран текст. Понякога завършва и с “подпис”, signature block. Полета в заглавието Заглавието включва най-малко следните полета: From: e-mail address и евентуално името на изпращача. При подателя се попълва автоматично. To: Адрес(ите) и евентуално име(ната) на получател(ите). CC: До кой да се изпрати видимо за получателя To: копие. Bcc: Blind Carbon Copy До кой да се изпрати невидимо за получателя To: копие. Subject: Или Относно: Предмета на съобщението. Date: Дата и час на изпращане в локалното време. Поставя се автоматично. Спекулации с "From" С полето "From" може лесно да се заблуждава, затова се препоръчва да се ползва цифрово подписване (OpenPGP или X.500 сертификат). Други важни полета In-Reply-To: Message-ID на съобщението, на което настоящото е отговор. Received: Проследява пътя, по който е минало съобщението, през кои пощенски сървъри. Показва кой е истинския подател по IP адрес. References: Message-ID на това съобщение и на това, на което е отговор. Reply-To: Адресът за отговор на подателя. Пример. Заглавие на phishing съобщение Subject: Уважаеми Uni-sofia.bg потребителски акаунт From: Софийски университет <webmaster@ Uni-sofia.bg> Date: Sun, March 21, 2010 13:22 To: undisclosed-recipients:; Priority: Normal Mailer: SquirrelMail/1.4.13 Return-Path: [web-master@Uni-sofia.bg](mailto:web-master@Uni-sofia.bg) Received: from mailbox.uni-sofia.bg ([unix socket]) by mailbox.uni-sofia.bg (Cyrus v2.3.7-Invoca-RPM-2.3.7- 7.el5\_4.3) with LMTPA; Sun, 21 Mar 2010 13:22:36 +0200 X-Sieve: CMU Sieve 2.3 Received: from olc-11.verat.net (olc-11.verat.net [62.108.127.37]) by mailbox.uni-sofia.bg (8.13.8/8.13.8) with ESMTP id o2LBMYL9015117 for <stefan@ucc.uni-sofia.bg>; Sun, 21 Mar 2010 13:22:35 +0200 Received: from webmail.verat.net (webmail.verat.net [85.222.160.153]) by olc-11.verat.net (Postfix) with ESMTP id D595BFC999; Sun, 21 Mar 2010 12:18:43 +0100 (CET) Received: from 41.138.189.77(SquirrelMail authenticated user djmaxa) by webmail.verat.net with HTTP; Sun, 21 Mar 2010 12:22:33 +0100 (CET) Message-ID: < [3754.41.138.189.77.1269170553.squirrel@webmail.verat.net](mailto:3754.41.138.189.77.1269170553.squirrel@webmail.verat.net) > Date: Sun, 21 Mar 2010 12:22:33 +0100 (CET) Subject: =?windows-1251?Q?=D3=E2=E0=E6=E0=E5=EC=E8\_Uni-sofia.bg\_ From: =?windows-1251?Q?=D1=EE=F4 ... =?windows- 1251?Q?=E5=F2?= [web-master@Uni-sofia.bg](mailto:web-master@Uni-sofia.bg) **Reply-To:** [**w0642406@gmail.com**](mailto:w0642406@gmail.com)User-Agent: SquirrelMail/1.4.13 MIME-Version: 1.0 Content-Type: text/plain;charset=windows-1251 Content-Transfer-Encoding: 8bit X-Priority: 3 (Normal) Importance: Normal To: undisclosed-recipients:; X-Spam-Status: No, score=-2.6 required=5.0 tests=BAYES\_00 autolearn=ham version=3.2.5 X-Spam-Checker-Version: SpamAssassin 3.2.5 (2008-06- 10) on mailbox.uni-sofia.bg Кодиране. UTF-8. Първоначално E-mail е била 7-bit ASCII. Стандартът MIME въведе предаване и на не-ASCII данни. **UTF-8** (8-bit UCS1/Unicode Transformation Format) е кодиране на знаците с променлива дължина за Unicode2. Представя всеки знак в Unicode стандарта, но в същевременно е обратно съвместим с ASCII. Затова става все по-предпочитан за e-mail, web и др. UTF-8 кодира всеки знак (code point) с 1 до 4 байта, като с един байт се кодират 128 US-ASCII знаците. Internet Mail Consortium (IMC) препоръчва всички email програми да са в състояние да изобразяват и създават поща с помощта UTF-8. 1Universal Character Set (UCS) **ISO/IEC 10646** стандарт, разработен съвместно с Unicode Consortium. 2Unicode осигурява уникален номер за всеки знак, независимо от платформата, независимо от програмата, независимо от езика. UTF-8 Първите 128 знака (US-ASCII) им трябва 1 байт. Следващите 1920 – 2 байта. Това са латински букви с диакрити, гръцки, кирилица, арменски, арабски, иврит и др. 3 байта са необходими за за останалите лингвистични знаци. 4 байта – за знаци в други равнини на Unicode, рядко използвани в практиката. Диакрити И обикновен текст и HTML се допускат в e-mail. Малка буква *a* с диакрит: **SMTP** Протоколът за изпращане на поща е **SMTP** (Simple Mail Transfer Protocol). Базира се на транспортен протокол TCP. От клиента, от порт с номер по-голям от 1024, се прави заявка за съединение към IP адреса на пощенския сървър на порт 25, т.е. порт 25 стои отворен в пощенския сървър и чака заявка за съединение. Ако сървърът е в състояние да получи заявката, отговаря с 220, което означава готов. След това клиентът изпраща съобщение HELLO, а при успех сървърът отговаря с 250. След това клиентът изпраща MAIL FROM (от кого е пощата), RCPT TO (кой е получателя) и накрая се прехвърля самото съобщение, след което връзката се разпада. Описаното си е едно TCP/IP съединение. В неговите рамки се обменят ASCII съобщения, които са с определена структура. Пример на SMTP сесия По-долу имате един типичен пример на изпращане на съобщение по SMTP до две пощенски кутии (*alice* и *theboss*) в един и същ домейн (*example.com*). “Репликите” на сървъра са означени със (*S:*), а на клиента - с (*C:*). След като изпращачът на съобщението (STMP client) установи надежден канал получателя (SMTP server), сесията се отваря с поздравление от страна на сървъра. Клиентът започва диалог, отговаряйки с команда HELO, в която се идентифицира. Пример на SMTP сесия S: 220 smtp.example.com ESMTP Postfix C: HELO relay.example.org S: 250 Hello relay.example.org, I am glad to meet you C: MAIL FROM:[bob@example.org](mailto:bob@example.org) S: 250 Ok C: RCPT TO:[alice@example.com](mailto:alice@example.com) S: 250 Ok C: RCPT TO:[theboss@example.com](mailto:theboss@example.com) S: 250 Ok C: DATA S: 354 End data with <CR><LF>.<CR><LF> C: From: "Bob Example" [bob@example.org](mailto:bob@example.org) C: To: Alice Example [alice@example.com](mailto:alice@example.com) C: Cc: [theboss@example.com](mailto:theboss@example.com) C: Date: Tue, 15 Jan 2008 16:02:43 -0500 C: Subject: Test message C: C: Hello Alice. C: This is a test message with 5 header fields and 4 lines in the message body. C: Your friend, C: Bob C: . S: 250 Ok: queued as 12345 C: QUIT S: 221 Bye {Сървърът затваря сесията} POP3 Крайният получател на писмото не е SMTP-сървъра. В него се събират изпратените писма до съответния домейн. Сървърът трупа тези писма на диск при себе си. С помощта на друг протокол крайният получател изтегля получените писма от пощенския сървър. Например, **POP3** (Post Office Protocol). При него сървърът слуша на порт 110. За разлика от SMTP, POP3 поддържа автентикация на клиента (username + password), т.е. притежателят на пощенската кутия е регистриран като POP3 потребител. Когато клиентът се свърже към POP3 сървъра, той първо се идентифицира, след което може да извърши други команди за прочитане на получените от него mail-ове. **IMAP** Internet Message Access Protocol (IMAP или IMAP4) “слуша” на порт 143 и е другата възможност крайният клиент да получи достъп до пощата си, стояща на отдалечен сървър. Сегашната версия, IMAP version 4 revision 1 (IMAP4rev1) е дефинирана в RFC 3501. IMAP поддържа и online, и offline режими. E-mail клиенти с IMAP оставят съобщенията на сървъра, докато потребителят не реши да ги изтрие. IMAP позволява повече от един клиент да има достъп до една и съща пощенска кутия. Т.е можете да имате достъп до пощата си едновременно от няколко места. За разлика от POP3, IMAP4 клиентите са свързани за сървъра, докато потребителският интерфейс е активен. потребителите изтеглят съобщения на диска си само по желание. IMAP4 клиентите могат да създават, преименуват и/или изтриват пощенски кутии (потребителят ги вижда като папки) на сървъра и да местят съобщения между кутиите. POP3 и IMAP услуги предлагат Cyrus IMAP server (http://cyrusimap.web.cmu.edu/) и Dovecot ([www.dovecot.org](http://www.dovecot.org)). Документация на mail решение Системата работи в рамките на УИЦ на СУ, като върху нея се хостват няколко домейна. Sendmail за МТА Cyrus за MDA RedHat Directory Server за LDAP база за потребителски акаунти и конфигурация на sendmail phpLDAPadmin - за външно управление на LDAP директорията Saslauthd за посредник между cyrus и LDAP базата Spamassassin за антиспам защита Clamav за антивирусна защита http://www.mimedefang.org/ за връзка на sendmail със антивирусната защита и за допълнителни филтри SquirrelMail и HastyMail за уеб достъп до пощенските кутии. Пощенски концентратори (mail hubs) са SMTP сървъри. Замисълът на изграждането им е цялата входяща електронна поща за пощенски домейни в мрежата на СУ, да преминава през тях. Избягва се риск от пробив в сървъри за електронна поща, които не са обект на квалифицирана поддръжка и наблюдение (повечето факултетски сървъри за електронна поща). Алгоритъм на MX йерархия в DNS. MX йерархия, която обслужва схема на доставка на електронна поща чрез пощенските концентратори: $ORIGIN domain.uni-sofia.bg. MX 10 mail.*faculty*.uni-sofia.bg MX 20 ns.uni-sofia.bg. MX 20 ady.uni-sofia.bg. MX йерархия Съгласно тази MX йерархия при инциране на сесия за предаване на писмо към получател с пощенска кутия в домейна *faculty*.uni-sofia.bg: - опит да се установи SMTP сесия към mail.*faculty*.uni-sofia.bg. Ако този опит пропадне: - установяване на SMTP сесия към един от двата SMTP сървъра, с MX приоритет 20 (ns.uni-sofia.bg или ady.uni-sofia.bg). Принцип на действие на схемата Сървър от интернет изпраща електронно писмо до получател с кутия в пощенския домейн *faculty*.uni-sofia.bg. Изпращачът използва MX йерархията и неговият пощенски процес намира хоста с най-малък числов идентификатор, mail.*faculty*.uni-sofia.bg. Иницира се SMTP сесия от изпращача към mail.*faculty*.uni-sofia.bg. Принцип на действие на схемата Маршрутизатор пред сървъра *faculty*.uni-sofia.bg отхвърля пакета за инициране на SMTP сесията, като връща ICMP съобщение "ICMP port unreachable" или "ICMP host unreachable". След като изпращача получи това ICMP съобщение, избира в случаен ред един от двата пощенски концентратора (ns.uni-sofia.bg или ady.uni-sofia.bg) и отново ще иницира SMTP сесия към избрания концентратор. Концентраторът приема писмото и после го доставя до *faculty*.uni-sofia.bg. SquirrelMail и Hastymail squirrelmail.org е с вградена PHP поддръжка за IMAP и SMTP, страниците са чист HTML 4.0 (не е необходим JavaScript) за съвместимост с всички браузъри. mailbox.uni-sofia.bg hastymail.org - IMAP/SMTP клиент, писан на PHP. Съвместим с PDAs, мобилни апарати, текстови и други браузъри. mailbox.uni-sofia.bg/mobile