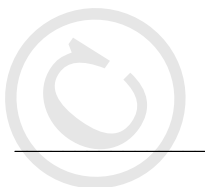


Student-notities
Linux firewall configuratie
Voorbeeld-hoofdstuk

02. Linux netfilter

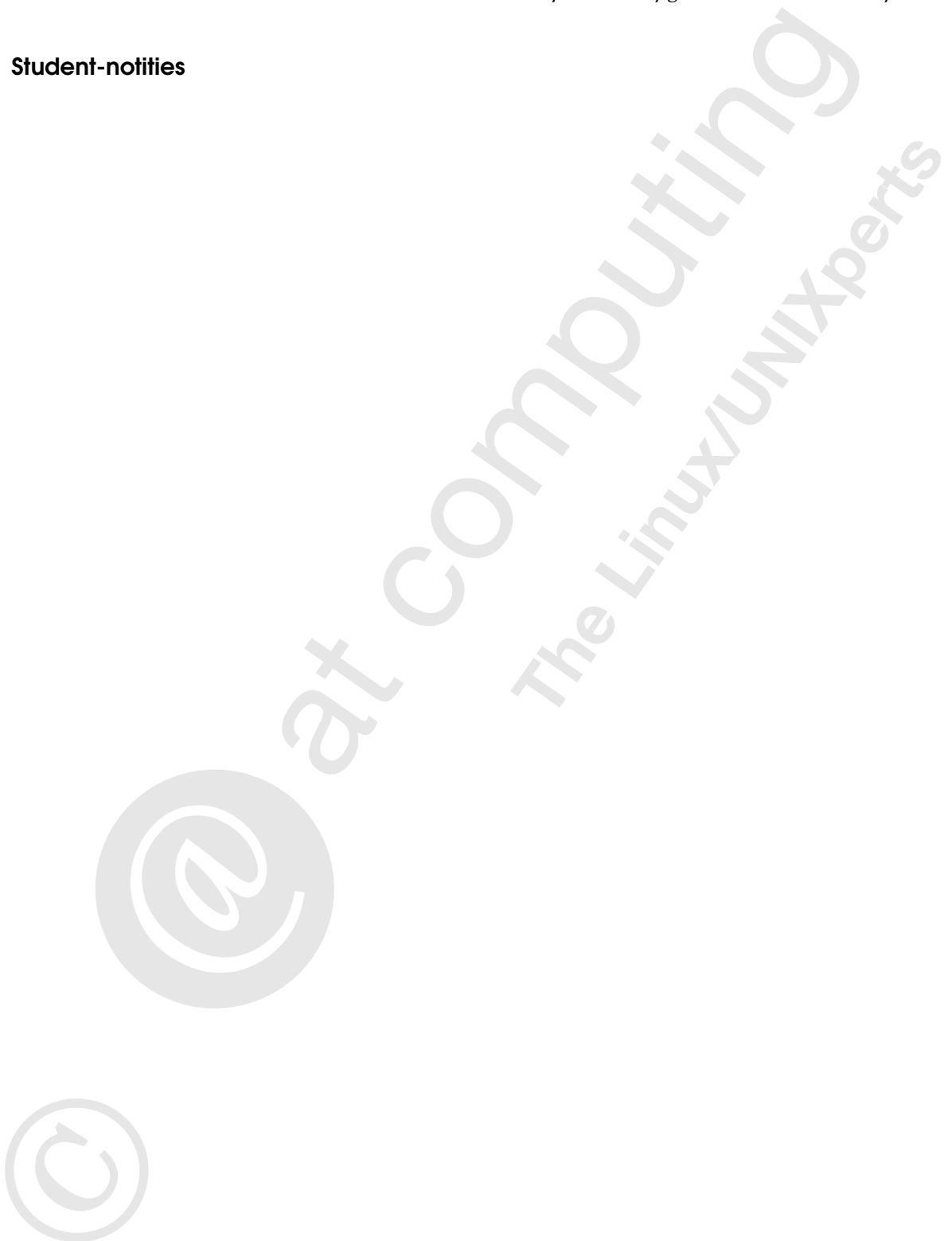
 **at computing**
The Linux/UNIXperts

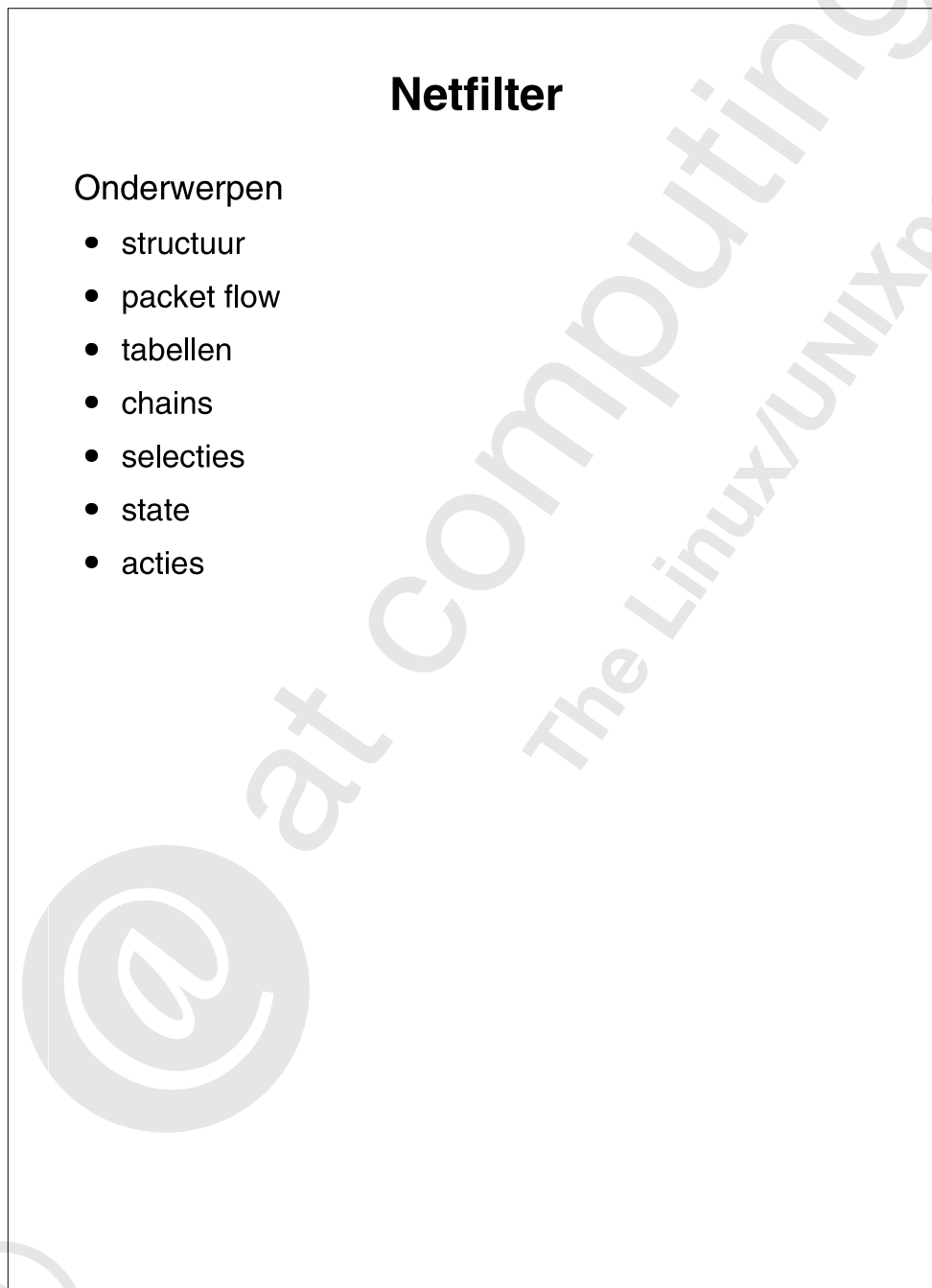
Nijmegen



Copyright © AT Computing 2006
Versie: 1b

Student-notities



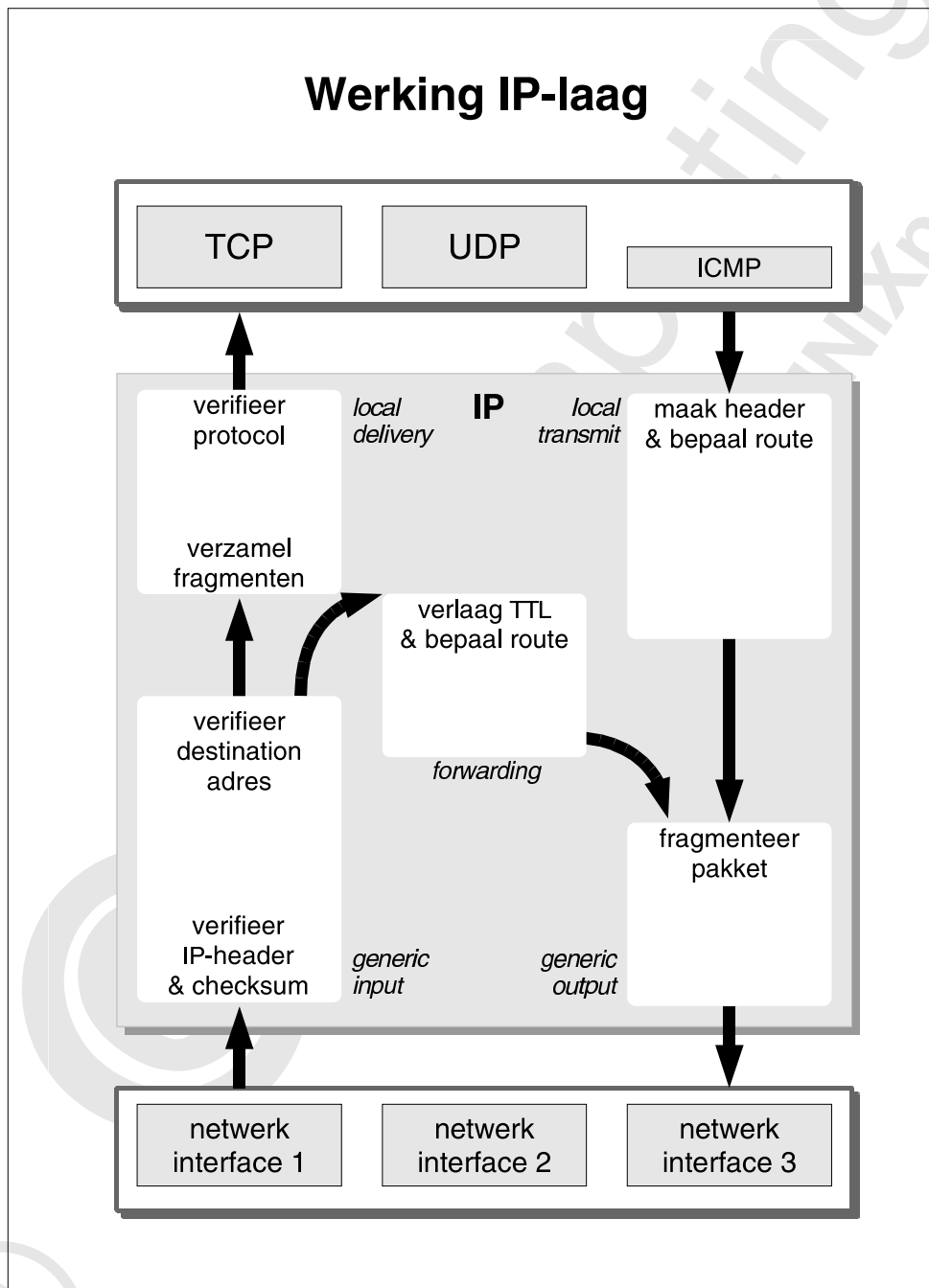


Figuur 1.

Student-notities

In de Linux IP-laag worden bijzonder veel beslissingen genomen. We zullen de blokjes op de sheet hiernaast één voor één de revue laten passeren.

- *generic input*
Pakketjes die de machine binnenkomen worden als eerste op hun gezondheid beoordeeld. Als het pakket in orde is bevonden (met de checksum), wordt gekeken of het pakket geforward moet worden, of dat het voor de lokale machine bedoeld is (m.a.w. wordt het pakket doorgegeven aan "local delivery" of aan "forwarding").
- *local delivery*
Van een pakket dat aangeboden wordt voor lokale aflevering worden als eerste alle fragmenten bij elkaar verzameld. Pas als alle fragmenten van een gefragmenteerd pakket zijn aangekomen, wordt het pakket in zijn geheel bij de bovenliggende laag aangeboden. Hiervoor moet in het type veld gekeken worden van de IP-header, daar staat immers of het een TCP, IP danwel ICMP pakket betreft
- *forwarding*
Als de generic input module heeft besloten dat het pakket gerouteerd moet worden, wordt hier als eerste de TTL verlaagd. Als de TTL hierdoor op nul terecht komt, zal het pakket verwijderd worden, en zal een ICMP time exceeded naar de afzender worden verstuurd. In de forwarding module wordt ook besloten welke route er genomen moet gaan worden. Dat heeft meerdere gevolgen:
 - Wat moet het adres zijn van de eerstvolgende hop?
 - Welk uitgaand interface moet we gebruiken?
 - Welk afzenderadres gebruiken we?
- *local transmit*
Voor pakketten die de machine zelf gegenereerd heeft, moet een nieuwe header worden aangemaakt. in deze header moet uiteraard ook een afzenderadres staan. Daarom wordt op deze plaats ook de routeringsbeslissing genomen:
 - Wat moet het adres zijn van de eerstvolgende hop?
 - Welk uitgaand interface moet we gebruiken?
 - Welk afzenderadres gebruiken we?
- *generic output*
In dit blok is bekend via welk interface een pakket de machine zal verlaten. Op ieder interface kan een andere MTU (maximale pakketgrootte) zijn ingesteld. Daarom wordt er vlak voor het naar buiten sturen van het pakket besloten of er gefragmenteerd moet worden.



Figuur 2.

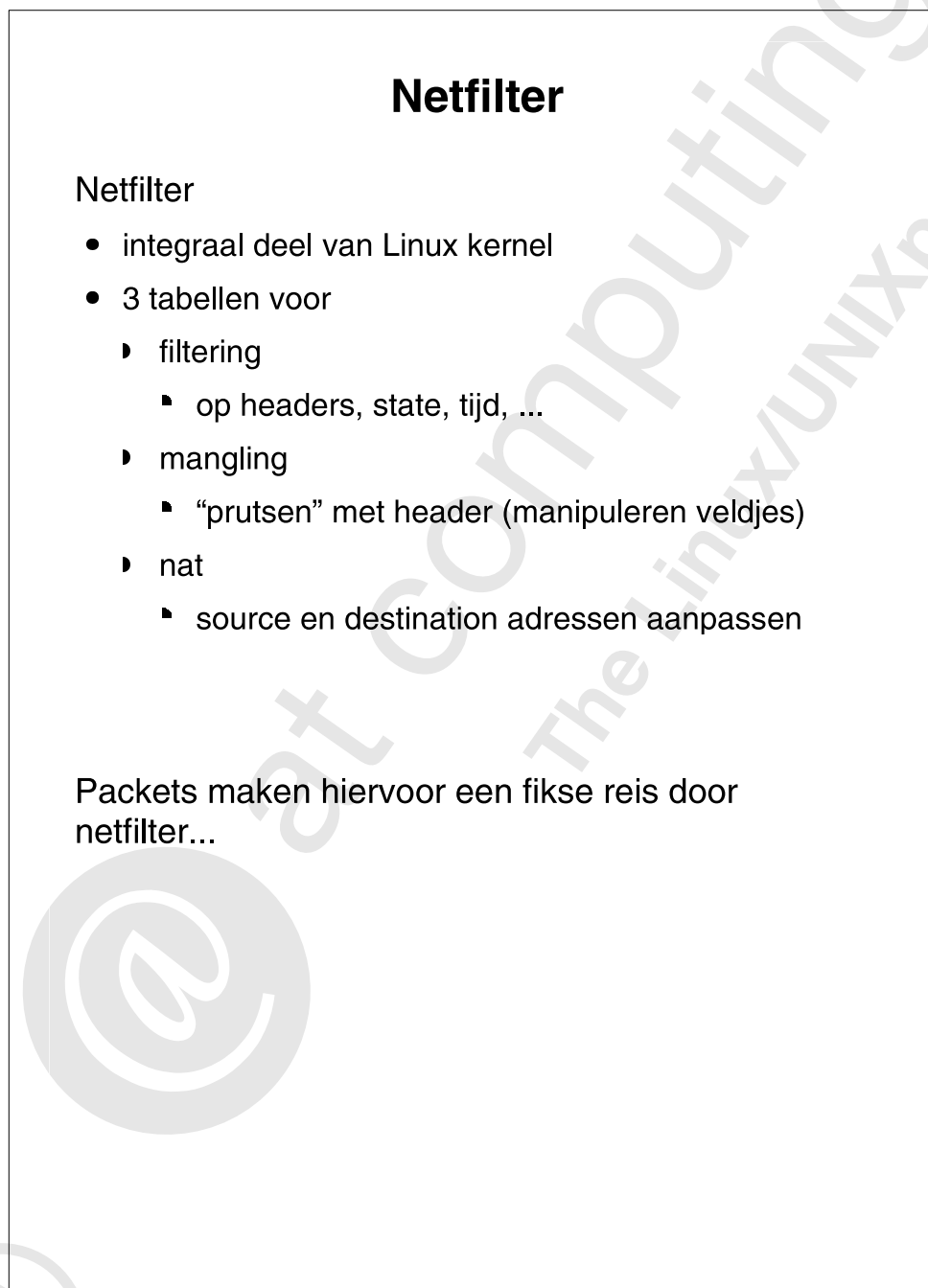
Student-notities

De vijf componenten die we in de IP-laag van Linux hebben gezien, kunnen allemaal filteren. Filteren houdt hierbij in:

- pakketten al dan niet doorlaten
- pakketten al dan niet wijzigen

Alle filtering maakt deel uit van *netfilter*. Netfilter is een volwaardig onderdeel van de Linux kernel en bestaat uit een drietal tabellen:

- *filter*
In deze tabel kan gefilterd worden op allerlei eigenschappen van een pakket: de headers, de tijd, eigenaarschap, state, ... Dit is dus de plaats waar de beslissing wel of niet doorlaten wordt genomen.
- *mangle*
Sommige velden in de header kunnen in deze tabel aangepast worden. Voorbeelden zijn de TTL (max hopcount) en TOS (type of service) velden van de IP header.
- *nat*
Deze tabel wordt gebruikt om source en destination adressen van de IP header aan te passen, ten behoeve van natting. Netfilter zal tegelijkertijd ook de TCP of UDP header aanpassen, zodat van retourverkeer de wijziging "in omgekeerde volgorde" kan worden gedaan.

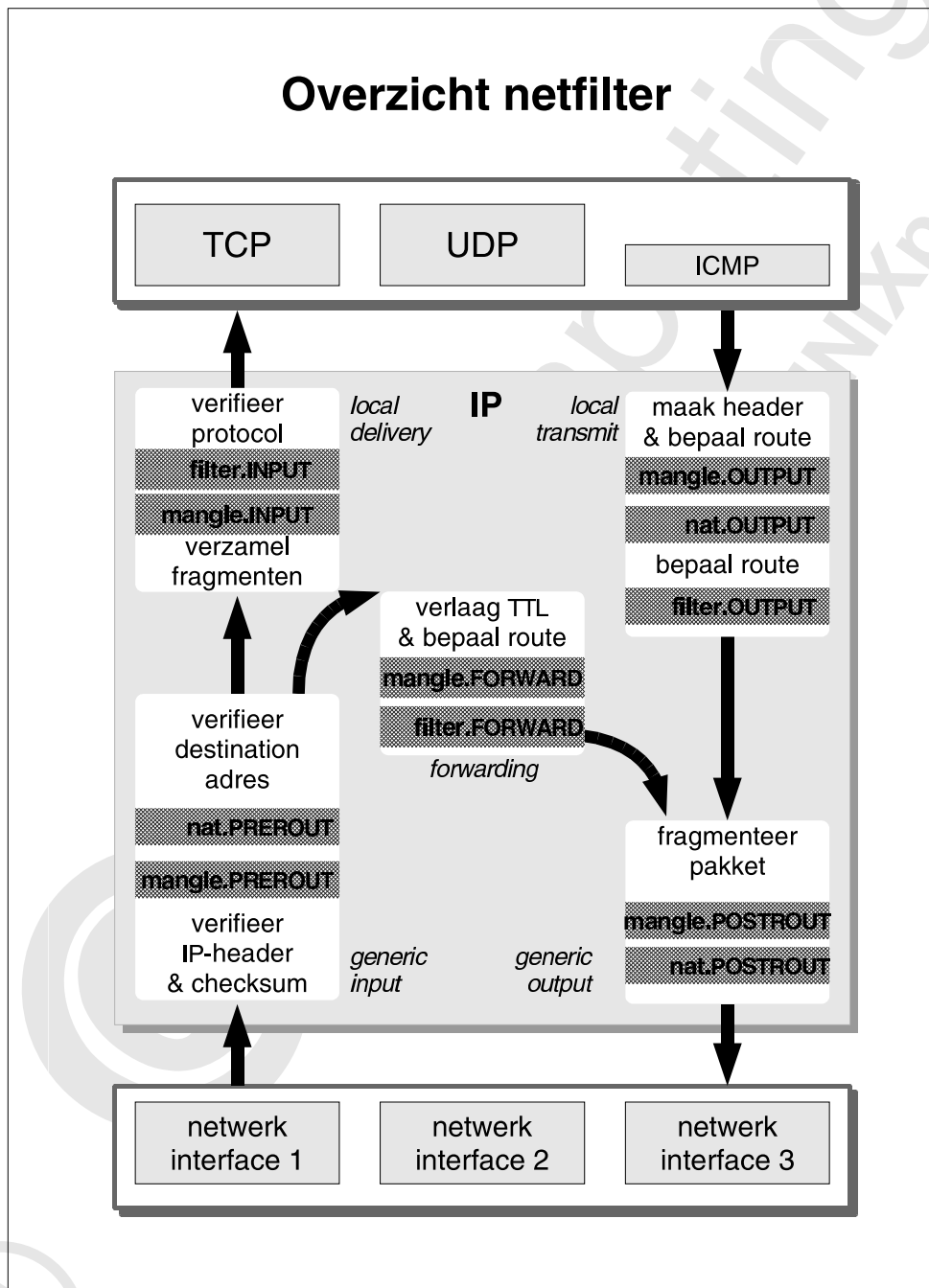


Figuur 3.

Student-notities

We kunnen nu in wat groter detail gaan kijken naar de stappen die doorlopen worden tijdens het filteren. Iedere tabel bestaat uit meerdere onderdelen, *chains* genaamd. We gaan later nog uitgebreid in op chains, maar van belang is te weten dat in de vijf blokken die we hiernaast zien, vast chains worden doorlopen.

- *generic input*
Tussen de test op de gezondheid van het pakket en de "houden we het hier of sturen we het door"-beslissing, worden de PREROUTING chains doorlopen van eerst de mangle tabel, en daarna de nat tabel. Deze chains zijn niet bedoeld voor filtering (wel/niet toelaten), maar voor het aanpassen van headers.
- *local delivery*
Na het verzamelen van de fragmenten worden de INPUT chains doorlopen van de mangle tabel en de filter tabel. De mangle tabel kan gebruikt worden om de header aan te passen ná de routing maar vóór aflevering bij het proces. Echte filtering vindt plaats in de INPUT chain van de filter tabel. Dit is de plaats waar we de meeste beveiliging-georiënteerde beslissingen zullen nemen voor verkeer naar de lokale machine. Dit is ook de plaats waar een eventuele "personal firewall" ingrijpt.
- *forwarding*
Ook in het forwarding blok wordt de mangle en filter tabel doorlopen, maar dan de FORWARD chain. In een router tussen twee netwerken in, is de FORWARD chain van de filter tabel de plaats waar beslissingen genomen worden om verkeer door te laten danwel tegen te houden. De kern dus van waar "men" aan denkt bij een firewall.
- *local transmit*
Voor pakketten die op de machine zelf ontstaan, wordt in het local transmit blok steeds de OUTPUT chain doorlopen van achtereenvolgens de mangle, de nat en de filter tabel. Als je uitgaand verkeer van de machine zelf wilt beperken, is de OUTPUT chain van de filter tabel de aangewezen plaats. De OUTPUT chain van de nat tabel kan gebruikt worden om te natten voor pakketten die vanaf de machine zelf komen.
- *generic output*
Hier worden twee POSTROUTING chains doorlopen: eerst die van de mangle tabel, vervolgens die van de nat tabel. Er kan voor uitgaand verkeer dus op twee plaatsen aan source-natting gedaan worden: In de OUTPUT chain van de nat tabel voor pakketten die van de machine zelf afkomstig zijn, en in de POSTROUTING tabel van de nat tabel voor *alle* pakketten die de machine verlaten. Dus voor zowel door de machine heen gerouteerde pakketten, als voor door de machine zelf gegenereerde pakketten.

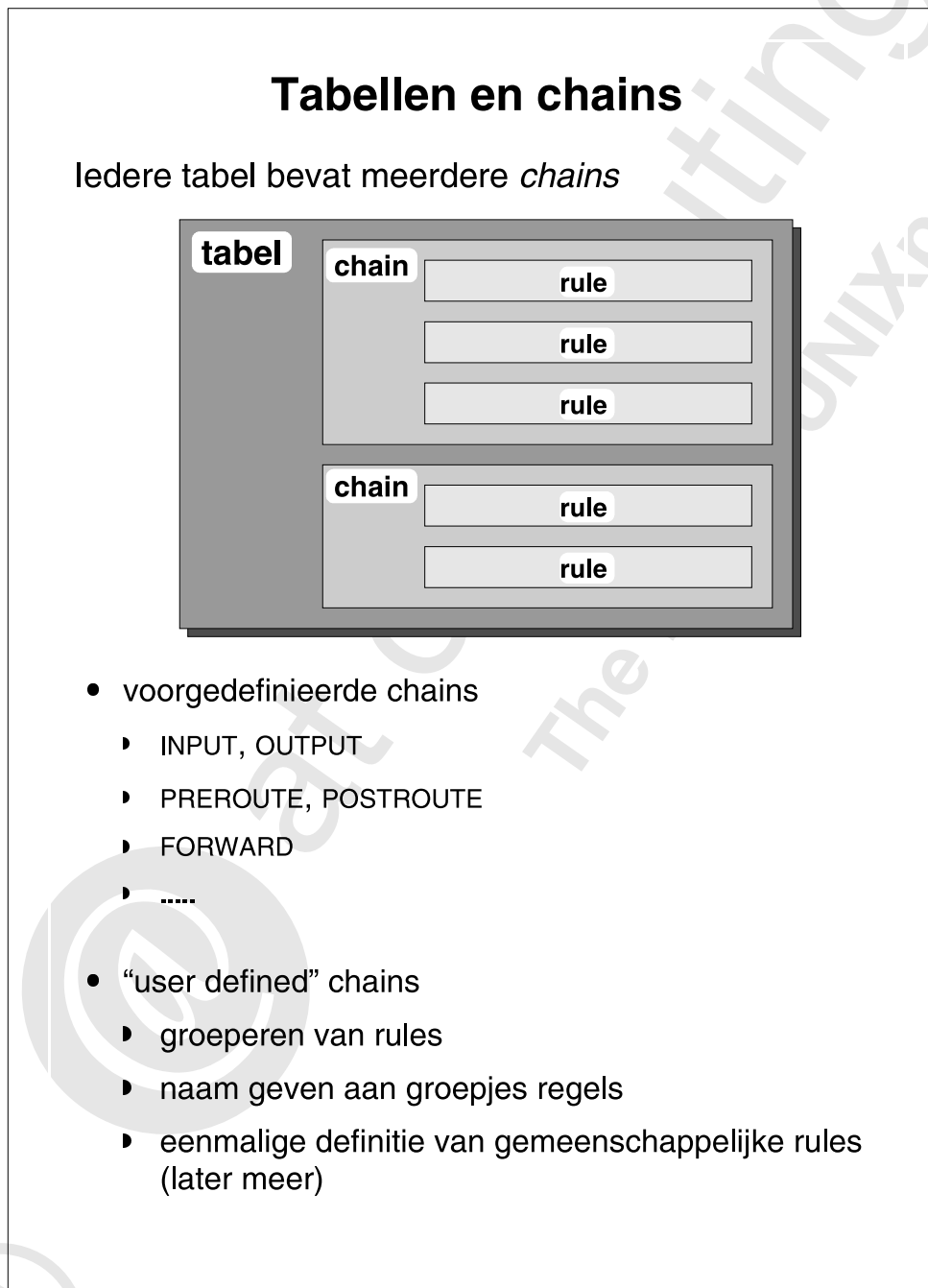


Figuur 4.

Student-notities

Zoals gezegd bevat een tabel een of meerdere *chains*. Een chain zelf bestaat weer uit een of meerdere *rules*. Eigenlijk is een chain niets anders dan een aantal regels (rules) die bij elkaar horen en samen een naam gekregen hebben.

Er zijn voorgedefinieerde chains (zoals INPUT, OUTPUT), maar je kunt zelf ook nieuwe chains toevoegen. Zelfgemaakte chains kunnen gebruikt worden om regels die in meerdere chains zouden moeten zitten, slechts één keer op te schrijven, en daar in de voorgedefinieerde chains gebruik van te maken. Ook kan het handig zijn een eigen chain te maken om een groepje regels die bij elkaar horen een naam te geven.



Figuur 5.

Student-notities

Als we verder inzoomen op de rule, zien we dat ook die uit meerdere onderdelen (kan) bestaan. Een regel bestaat uit selecties (daar komen we verderop uitgebreid op terug — maar denk bijvoorbeeld aan selecties op poortnummers) en uit acties. Alle regels in een chain worden één voor één doorlopen, tot er een actie wordt gebruikt die het doorlopen stopt.

Netfilter houdt per regel bij hoeveel pakketten (en bijgevolg hoeveel data-bytes) op deze regel pasten. Als alle selectiecriteria van een rule passen worden deze tellers opgehoogd, en wordt de actie die bij de regel hoort, uitgevoerd. Als er geen actie is, is deze regel "klaar", en gaan we naar de volgende regel.

Acties kunnen inhouden:

- Het in zijn geheel langslopen van de regels in een andere chain (te zien als een soort subroutine- of functieaanroep in een programmeertaal).
- Het afbreken van het doorlopen van de regels in deze chain. Er wordt verder gegaan bij de regel die deze chain "aanriep". Actie: RETURN.
- Het afbreken van het doorlopen van de regels in deze chain. Er wordt *niet* doorgedaan bij de aanroepende chain. We zijn dus klaar met dit pakket. Acties: DROP, ACCEPT, REJECT. ACCEPT wil zeggen dat het pakket verder mag, DROP en REJECT zullen het pakket tegenhouden. DROP doet dit stilzwijgend, REJECT zal een (instelbaar) ICMP bericht naar de afzender terugsturen.
- Niets. Er wordt verder gegaan bij de volgende rule in deze chain. Uiteraard wordt wel geteld hoe vaak de selectie van deze rule paste.

Als alle rules in een chain doorlopen zijn, en er is dus geen volgende rule te vinden in de chain gebeurt het volgende:

- Als het een zelf gemaakt chain betreft, wordt verder gegaan bij de "aanroepende" chain. Dit komt overeen met een impliciete RETURN als laatste actie in de chain.
- Als het een ingebouwde chain betreft (zoals FORWARD en OUTPUT), wordt als actie uitgevoerd wat voor deze tabel als *policy* is ingesteld. Dat kan DROP of ACCEPT zijn. Met de chain policies DROP en ACCEPT worden dus respectievelijk default deny en default allow ingesteld.



Chain policy

Rules: selectie(s) + actie(s)

Mogelijke gevolgen van een actie

- een andere chain wordt (even) langsgelopen
- doorlopen van chain eindigt – ga verder bij "vorige" chain (RETURN)
- definitieve beslissing wordt genomen - doorlopen rules eindigt
 - ▶ DROP: weiger packet
 - ▶ REJECT: weiger packet met opgaaf van reden
 - ▶ ACCEPT: accepteer packet
- niets (dus: volgende rule behandelen)

Als laatste rule niet "stopt"

- eigen gemaakte chain: ga verder bij "vorige" chain (impliciete RETURN)
- standaard chain: actie is de chain policy (impliciete DROP of ACCEPT)
 - ▶ deze policy implementeert default deny of default accept

Figuur 6.

Student-notities

In het resterende deel van dit hoofdstuk gaan we in op aanroepen vanuit de ene chain naar de andere chain, en kijken we naar de verschillende selecties. Verder behandelen we uitgebreid de implicaties van *statefull* filtering.



at computing
The Linux/UNIXperts