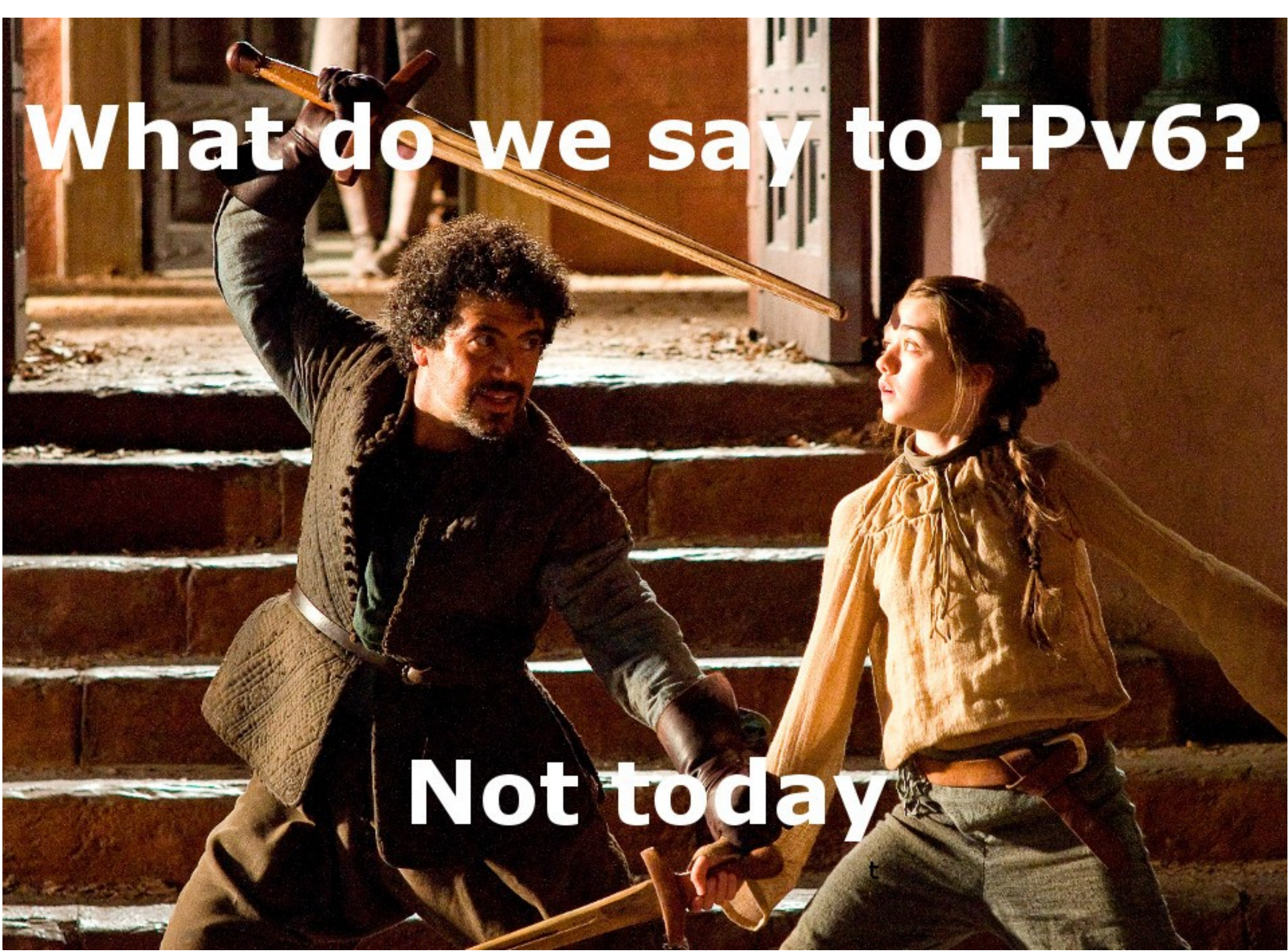
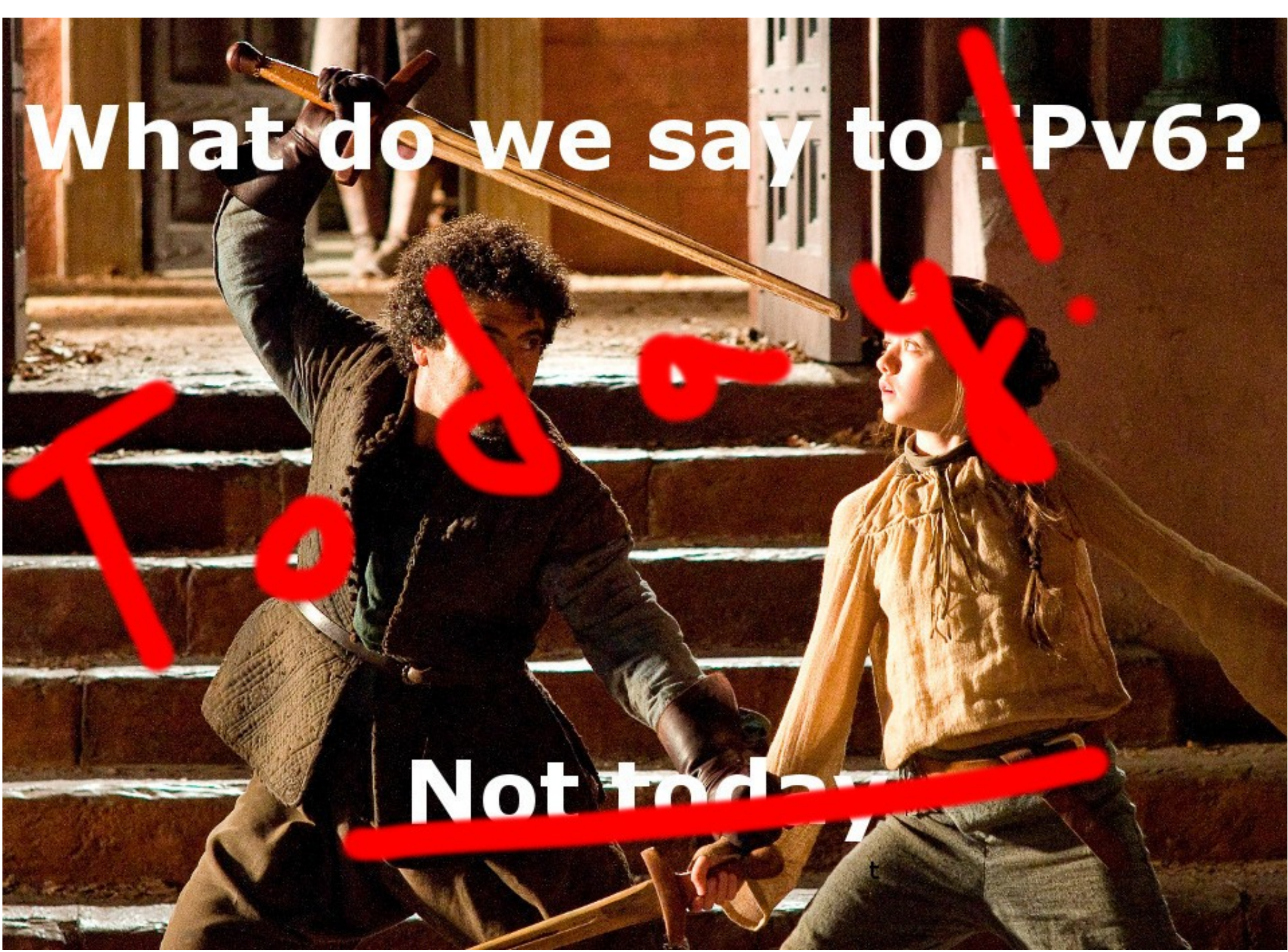

What do we say to IPv6?
Not today

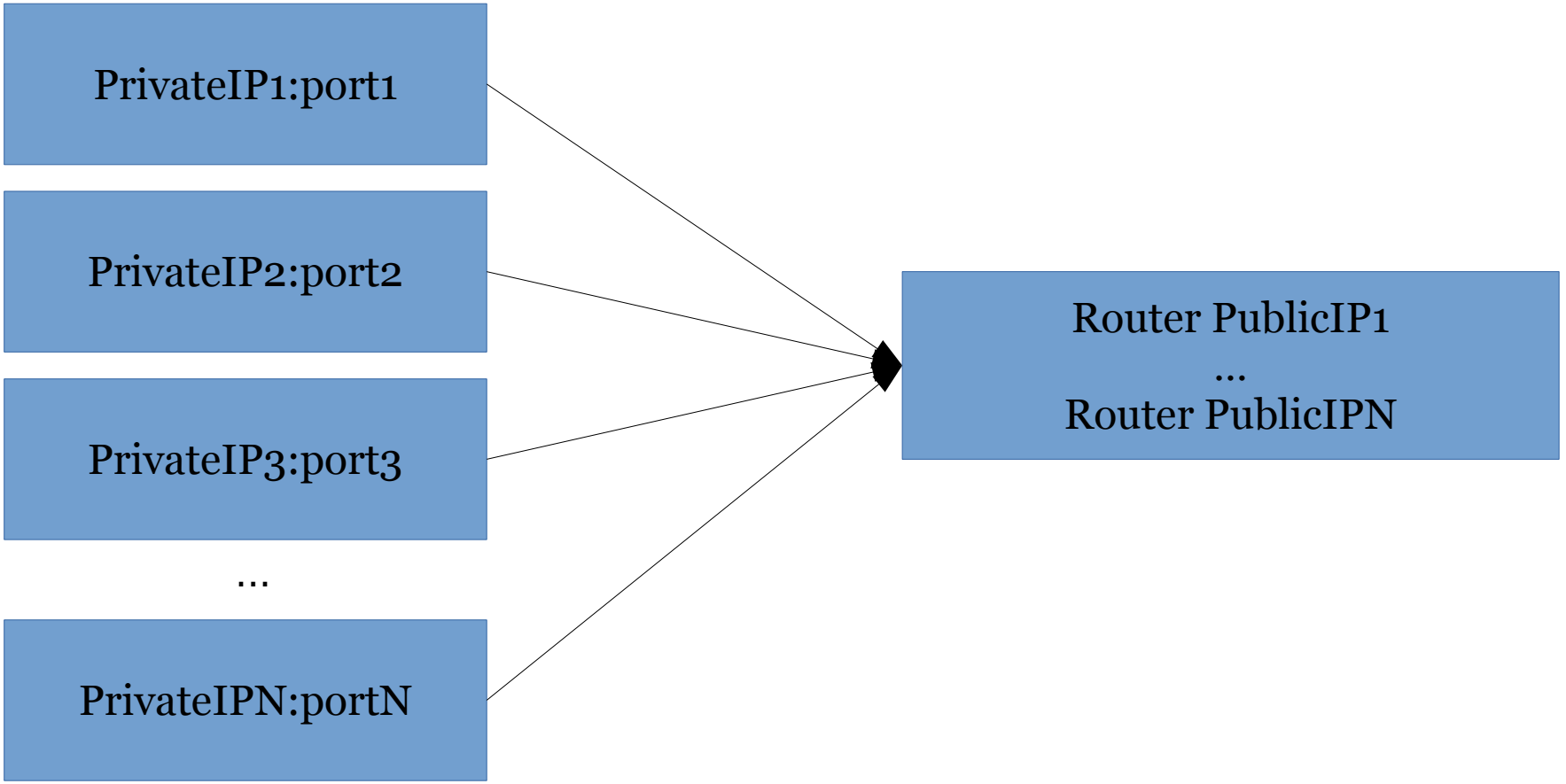

What do we say to IPv6?
Today!
Not today

IPv6:
там, за горизонтом

Проблемы IPv4:

- недостаточность объёма 32-битного адресного пространства;
- разрастание таблиц маршрутизации;
- сложность массового изменения IP-адресов;
- относительная сложность обработки заголовков пакетов IPv4;
- сложность создания полноценных облачных и p2p сервисов поверх NAT

NAT (RFC 1631, RFC 3022)



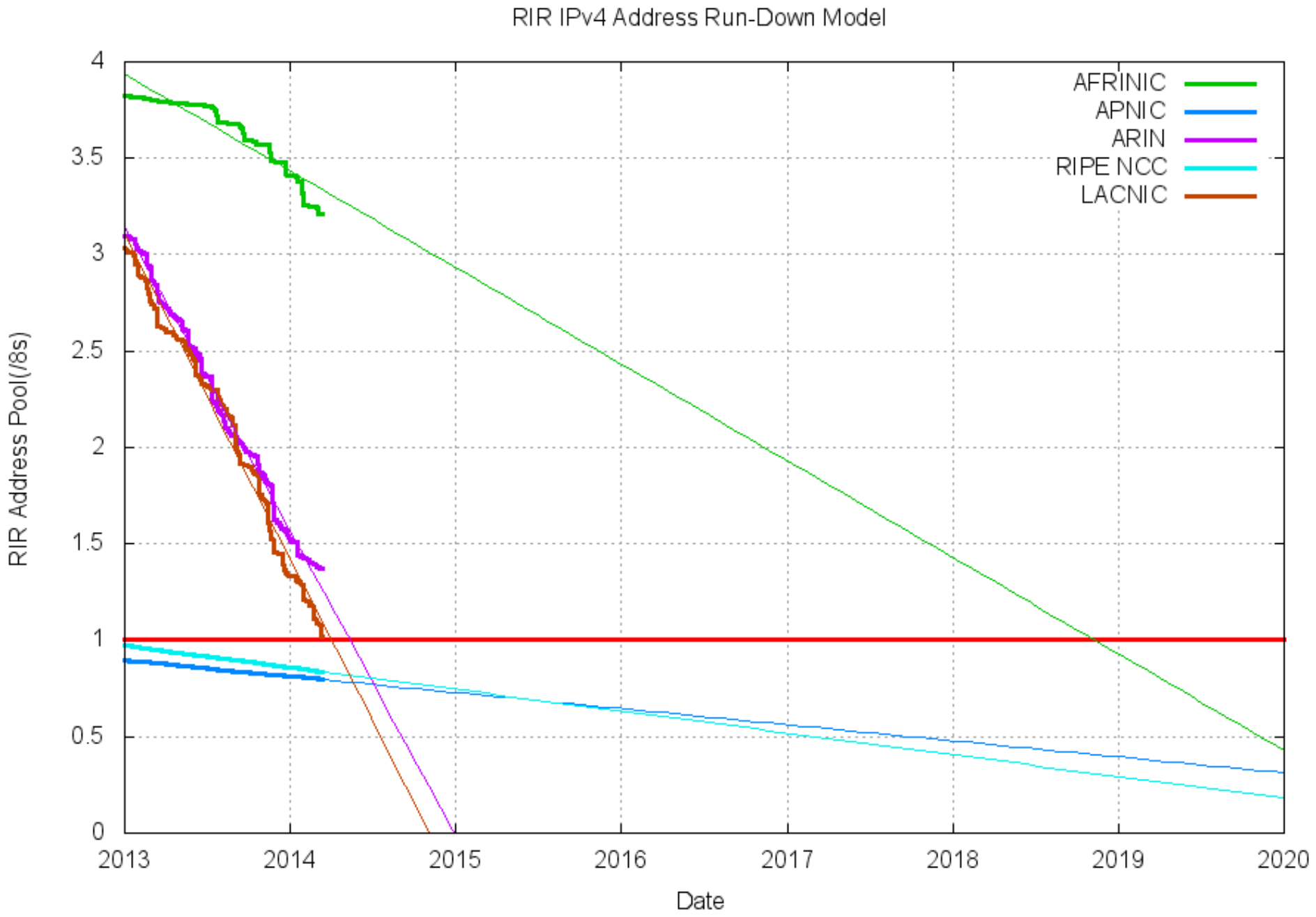
NAT types				
Private pair	Public pair	Permanent mapping	Incoming TCP connection	NAT type
PrivateIP1:port1 PrivateIP1:portN	PublicIP1:port1' PublicIP1:portN'	+		Static NAT
PrivateIP1:port1 PrivateIPN:portN	PublicIP1:port1 PublicIP1:portN	+/–		Port NAT (PAT, Masquarading, Overload NAT, NAPT)
PrivateIP1:port1 PrivateIPN:portM	PublicIP1:port1' PublicIPN:portM'	–		Dynamic NAT
PrivateIP1:port1	PublicIP1:port1'	–	–	Symmetric NAT
PrivateIP1:port1	PublicIP1:port1'	+	+	Cone NAT, Full Cone NAT
PrivateIP1:port1	PublicIP1:port1'	+	–	Address-Restricted cone NAT, Restricted cone NAT
PrivateIP1:port1	PublicIP1:port1'	–	+	Port-Restricted cone NAT

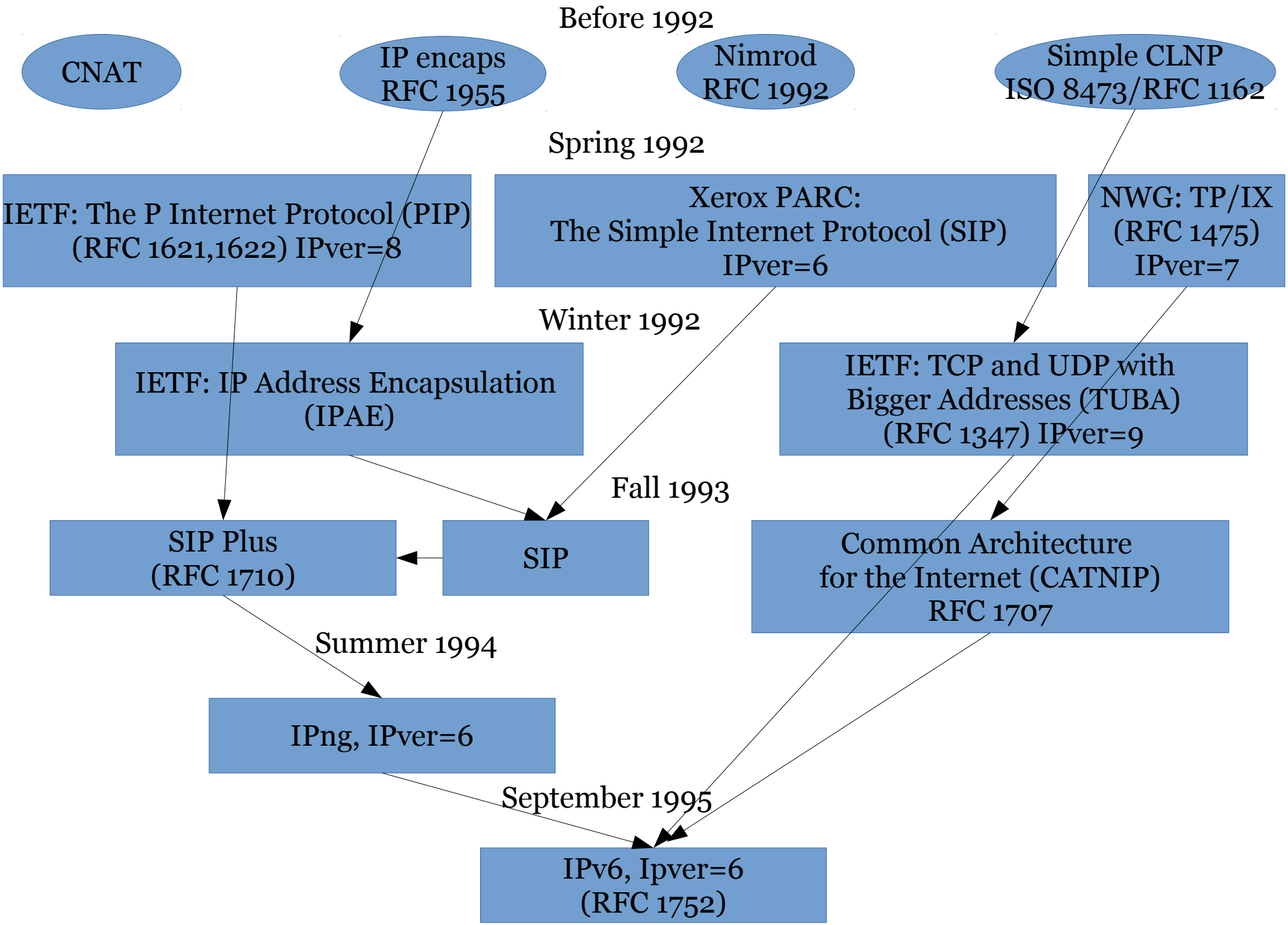
PRO:

- Экономия IP адресов
- Позволяет предотвратить или ограничить обращение снаружи ко внутренним хостам
- Позволяет скрыть определённые сервисы внутренних хостов, повышение безопасности и скрытие «непубличных» ресурсов

CONTRA:

- **Старые протоколы.** Протоколы, разработанные до массового внедрения NAT, не в состоянии работать, если на пути между взаимодействующими хостами есть трансляция адресов (например, FTP). М.б. решено при использовании Application-level firewall.
- **Невозможна идентификация пользователей по IP-адресу.** Из-за сложности с идентификацией пользователей необходимо хранить полные логи трансляций.
- **Иллюзия DoS-атаки.** Если NAT используется для подключения многих пользователей к одному и тому же сервису, это может вызвать иллюзию DoS-атаки на сервис. Частичным решением проблемы является использование пула адресов для трансляции.
- **Пиринговые сети и облачные сервисы.** В NAT-устройствах, не поддерживающих технологию Universal Plug & Play, в некоторых случаях, необходима дополнительная настройка при работе с пиринговыми сетями и некоторыми другими программами, в которых необходимо не только инициировать исходящие соединения, но также принимать входящие.
- **Нарушение принципа инкапсуляции стека протоколов.** Дополнительные задержки и накладные расходы.


RIR IPv4 Address Run-Down Model
RIR Address Pool(/8s)
Date
AFRINIC
APNIC
ARIN
RIPE NCC
LACNIC
0
0.5
1
1.5
2
2.5
3
3.5
4
2013
2014
2015
2016
2017
2018
2019
2020


Before 1992
CNAT
IP encaps
RFC 1955
Nimrod
RFC 1992
Simple CLNP
ISO 8473/RFC 1162
Spring 1992
IETF: The P Internet Protocol (PIP)
(RFC 1621,1622) IPver=8
Xerox PARC:
The Simple Internet Protocol (SIP)
IPver=6
NWG: TP/IX
(RFC 1475)
IPver=7
Winter 1992
IETF: IP Address Encapsulation
(IPAE)
IETF: TCP and UDP with
Bigger Addresses (TUBA)
(RFC 1347) IPver=9
Fall 1993
SIP Plus
(RFC 1710)
SIP
Common Architecture
for the Internet (CATNIP)
RFC 1707
Summer 1994
IPng, IPver=6
September 1995
IPv6, Ipver=6
(RFC 1752)

CATNIP, SIPP, TUBA comparison (1995, rfc 1752)

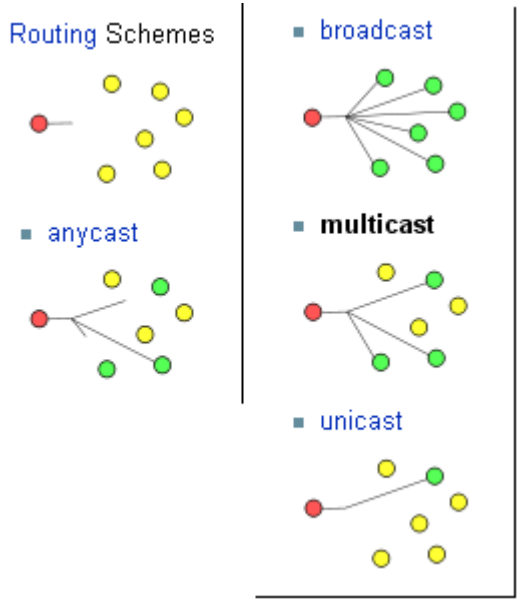
	CATNIP	SIPP	TUBA
complete spec	no	yes	mostly
simplicity	no	no	no
scale	yes	yes	yes
topological flex	yes	yes	yes
performance	mixed	mixed	mixed
robust service	mixed	mixed	yes
transition	mixed	no	mixed
media indepdnt	yes	yes	yes
datagram	yes	yes	yes
config. ease	unknown	mixed	mixed
security	unknown	mixed	mixed
unique names	mixed	mixed	mixed
access to stds	yes	yes	mixed
multicast	unknown	yes	mixed
extensibility	unknown	mixed	mixed
service classes	unknown	yes	mixed
mobility	unknown	mixed	mixed
control proto	unknown	yes	mixed
tunneling	unknown	yes	mixed

IPv6 Features :

- Первое и самое основное – увеличение адресного пространства – поддерживает 2^{128} адресов (по 7 адресов каждому атому каждого человека на Земле. =)
- IPv6 хосты конфигурируются автоматически при подключении к маршрутизируемой IPv6 сети – при первом подключении к сети хост посылает link-local-multicast (broadcast) запрос на конфигурационные параметры, при нормальной конфигурации рутер отвечает пакетом (*router advertisement* packet) с конфигурационными параметрами сети; также поддерживается DHCPv6 и ручное конфигурирование.
- Multicast, который был опционален в IPv4 стал главной особенностью IPv6

IPv6 может быть трех типов:

- **Unicast** – идентифицирует один интерфейс узла сети;
- **Anycast (one-to-nearest)** – идентифицирует множество интерфейсов (обычно принадлежащих различным узлам); пакет, посылаемый по anycast-адресу должен быть доставлен только на один из интерфейсов (ближайший к отправителю), идентифицируемых этим адресом;
- **Multicast (one-to-any)** – идентифицирует множество интерфейсов (обычно принадлежащих различным узлам); пакет, посылаемый по multicast-адресу должен быть доставлен на все интерфейсы, идентифицируемые этим адресом.



- Соответственно broadcast трансляция осуществляется через multicast адрес: `FF02::1 (all-host-group)`.

IPv6 Features (cont):

- IPv6 jumbograms – payload («полезная нагрузка») в IP протоколах ограничена 64 Кибибайтами (кибибайт = 2^{10} байт), что не всегда удобно, поэтому в IPv6 присутствует jumbo payload option, позволяющая хостам обмениваться пакетами большего размера. При этом из-за ограничений TCP и UDP пакетов, требуется подстраивание под транспортный уровень, то есть специальные механизмы для такой передачи (rfc2675).
- Безопасность на сетевом уровне (механизмы аутентификации и шифрования на уровне IP-пакетов).
- Изменено представление необязательных полей заголовка.
- Упрощен стандартный заголовок IP-пакета.
- Введены метки потоков данных.

Адреса IPv6

- 128 бит.
- Unicast, Anycast, Multicast.
- Обычно IP-адрес записывают как последовательность из восьми двухбайтовых чисел x:x:x:x:x:x:x:x, например:
  1080:0:0:0:8:800:200C:417A unicast-address
  FF01:0:0:0:0:0:0:43 multicast-address
  0:0:0:0:0:0:0:1 адрес loopback-интерфейса
- Кроме того, нулевые элементы можно опустить, вставляя вместо них :: (можно использовать только один раз, как правило, последовательность наибольшей длины)
- Маска IPv6, точно так же как и в IPv4, состоит из непрерывной последовательности единиц (в двоичном представлении), завершаемой непрерывной последовательностью нулей. Маска также может быть записана с использованием суффикса ::, например:
  1080:0:0:0:8::

IP address types:

Тип IP-адреса определяется по нескольким ведущим битам IP-адреса. Например, мультикастовый адрес начинается с октета **0xff**, адреса, выделенные в зависимости от географического положения узла, начинаются с бит **100**, а адреса, выделяемые провайдером – с **010**. Все типы IP-адресов перечислены в RFC 1884.

IPv6-адрес, содержащий 96 ведущих нулевых бит, на самом деле является **IPv4-адресом**, инкапсулированным в адрес IPv6 за исключением случая, когда это – адрес loopback-интерфейса (::1).

Link-Local-адреса используются только в пределах локальной физической подсети для идентификации узлов исключительно этой подсети. Они начинаются с последовательности **1111111010** и содержат только идентификатор интерфейса в подсети. Пакеты с этими адресами никогда не должны передаваться в другие подсети.

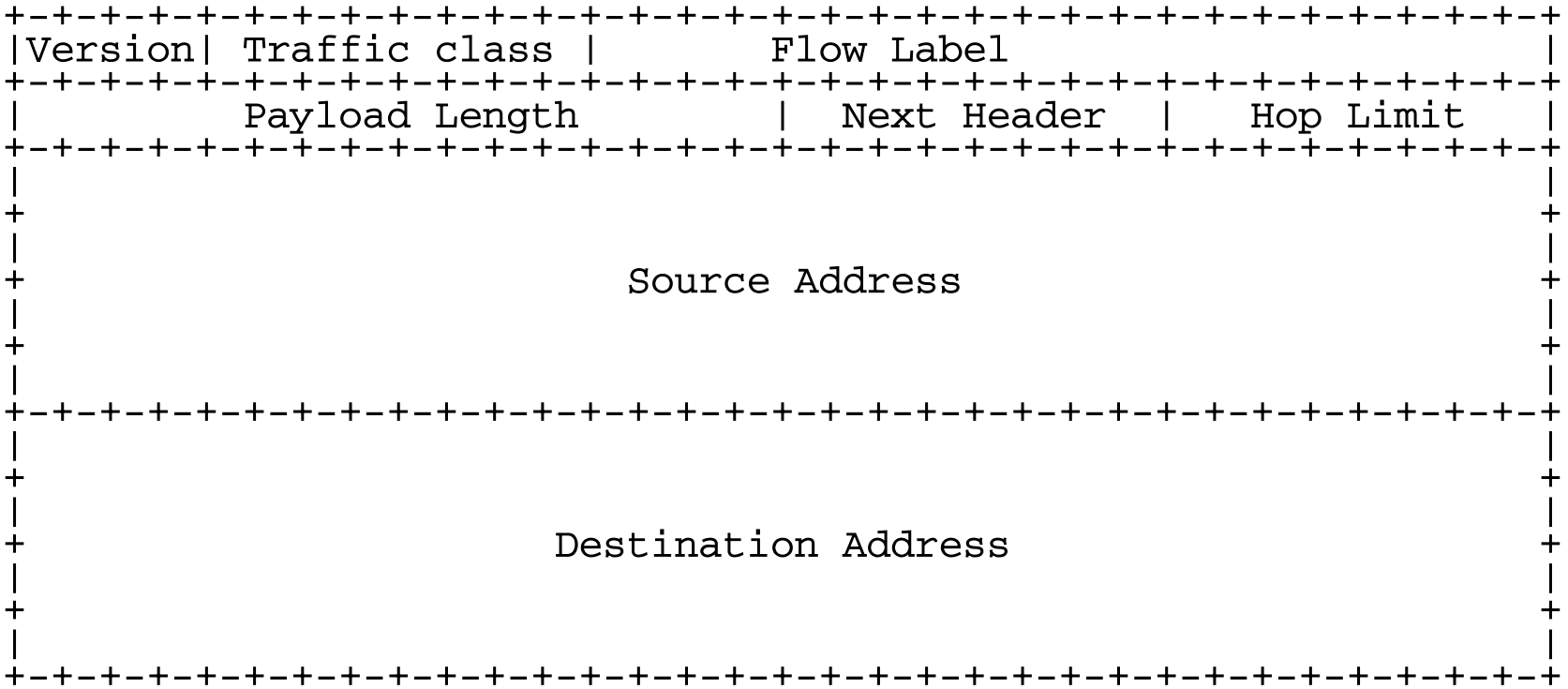
Site-Local-адреса начинаются с последовательности **1111111011**, содержат идентификатор подсети и идентификатор интерфейса в ней, но не могут быть переданы за пределы множества подсетей, из которых состоит сеть организации, их присвоившей.

Multicast-адреса начинаются с **11111111**.

Адрес **FF01:0:0:0:0:0:0:1** обозначает «все узлы в пределах области действия».

IPv6 Header Format

The IPv6 header, although longer than the IPv4 header, is considerably simplified. A number of functions that were in the IPv4 header have been relocated in extension headers or dropped. [Deering94b]



* **Version** - Internet Protocol version number. IPng has been assigned version number 6. (4-bit field)

* **Flow Label** - This field may be used by a host to label those packets for which it is requesting special handling by routers within a network, such as non-default quality of service or "real-time" service. (28-bit field)

* **Payload Length** - Length of the remainder of the packet following the IPv6 header, in octets. To permit payloads of greater than 64K bytes, if the value in this field is 0 the actual packet length will be found in an Hop-by-Hop option. (16-bit unsigned integer)

* **Next Header** - Identifies the type of header immediately following the IPv6 header. The Next Header field uses the same values as the IPv4 Protocol field (8-bit selector field)

* **Hop Limit** - Used to limit the impact of routing loops. The Hop Limit field is decremented by 1 by each node that forwards the packet. The packet is discarded if Hop Limit is decremented to zero. (8-bit unsigned integer)

* **Source Address** - An address of the initial sender of the packet. (128 bit field)

* **Destination Address** - An address of the intended recipient of the packet (possibly not the ultimate recipient, if an optional Routing Header is present). (128 bit field)

Traffic Class

Обеспечение приоритезации пакетов обеспечивается маршрутизаторами на основе поля приоритета. Множество значений этого поля разделено на два подмножества:

- от 0 до 7 — трафик с контролем перегрузки (например, протокол TCP снижает трафик при получении сигнала перегрузки);
- от 8 до 15 — трафик без контроля перегрузки (приложения реального времени с постоянной скоростью).

Разработчики IPv6 рекомендуют использовать для определённых категорий приложений, управляющих сигналами перегрузки, следующие коды приоритета:

Код приоритета	Назначение	Пример
0	Нехарактеризованный трафик	
1	Заполняющий трафик	Сетевые новости
2	Несущественный трафик	Электронная почта
3	Резерв	
4	Существенный трафик	FTP, HTTP, NFS
5	Резерв	
6	Интерактивный класс	Telnet, SSH, Xwindow, IMs
7	Управляющий трафик	Протоколы маршрутизации, SNMP

Для приложений, скоростью передачи которых невозможно управлять, разработчики определяют, что большему значению кода соответствует более высокий приоритет. Для кода 8 установлено низшее значение приоритета, и он применяется для трафика, часть данных которого можно отбросить, например, потоковое видео высокого качества. Значение 15 следует применять для трафика с фиксированной скоростью, чувствительного к задержкам и потерям, например, трафик речевых кодеков с минимальной избыточностью.

Следует отметить, что трафик с возможностью управления перегрузками (0—7) и без неё (8—15) обрабатываются маршрутизаторами независимо, то есть не существует взаимного соответствия между приоритетами данных видов трафика.

Flow Label

Введение в протоколе IPv6 поля «Метка потока» позволяет значительно упростить процедуру маршрутизации однородного потока пакетов.
Поток это последовательность пакетов, посылаемых отправителем определённому адресату, при этом предполагается, что все пакеты данного потока должны быть подвергнуты определённой обработке. Характер данной обработки задаётся дополнительными заголовками.
Допускается существование нескольких потоков между отправителем и получателем.
Метка потока присваивается узлом-отправителем путём генерации псевдослучайного 24-битного числа. Все пакеты одного потока должны иметь одинаковые заголовки, обрабатываемые маршрутизатором.
При получении первого пакета с меткой потока маршрутизатор анализирует дополнительные заголовки, выполняет предписанные этими заголовками функции и запоминает результаты обработки (адрес следующего узла, опции заголовка переходов, перемещение адресов в заголовке маршрутизации и т. д.) в локальном кэше. Ключом для такой записи является комбинация адреса источника и метки потока. Последующие пакеты с той же комбинацией адреса источника и метки потока обрабатываются с учётом информации кэша без детального анализа всех полей заголовка.
Время жизни записи в кэше составляет не более 6 секунд, даже если пакеты этого потока продолжают поступать. При обнулении записи в кэше и получении следующего пакета потока, пакет обрабатывается в обычном режиме и для него происходит новое формирование записи в кэше. Следует отметить, что указанное время жизни потока может быть явно определено узлом отправителем с помощью протокола управления или опций заголовка переходов, и может превышать 6 секунд.

Концепция расширяемых заголовков

IPv6 определяет ещё несколько типов заголовков, используемых для обеспечения работы протокола, которые могут помещаться за описанным заголовком и содержать различные опции. Во всех этих заголовках первым полем являются идентификатор следующего заголовка (кроме No Next Header).

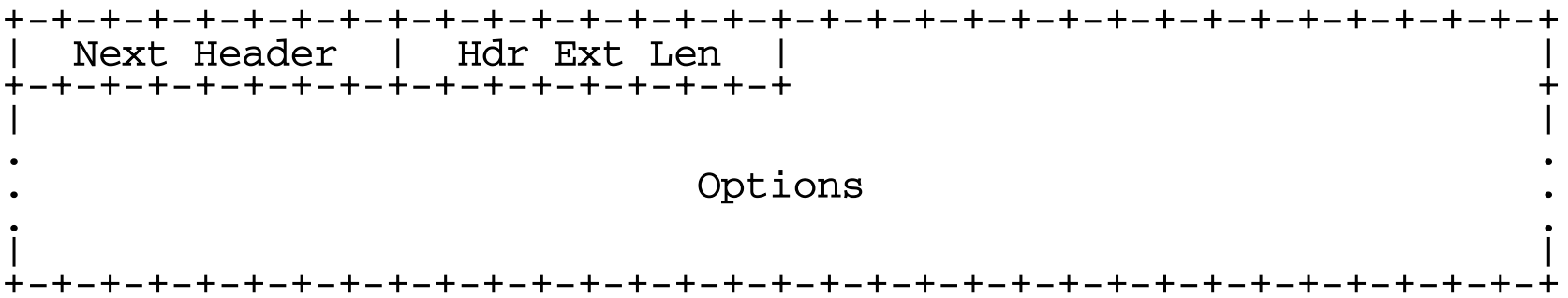
- Hop-by-Hop Options Header содержит опции, которые должны обрабатываться на всех промежуточных рутерах. Примерами таких опций могут служить опция Jumbo, которая служит для указания длины данных, если она превышает 65535 байт, и опция Router Alert, указывающая рутеру, что содержимое пакета может представлять для него интерес, несмотря на то, что пакет транзитный.
- Routing Header дает возможность отправителю указать список узлов, которые пакет должен посетить по дороге к конечному получателю. Этот заголовок является аналогом опции Loose Source and Record Route в IPv4.
- Fragment Header используется для фрагментирования датаграмм. В отличии от IPv4, операцию фрагментирования может осуществлять только отправитель. Заголовок содержит стандартные параметры фрагментирования: идентификатор датаграммы, смещение фрагмента и флаг <More Fragments>. Поскольку промежуточные узлы не могут фрагментировать пакеты, узел-отправитель должен поддерживать алгоритм для определения MTU всего пути или посылать очень маленькие пакетики (до 1280 байт), которые обязаны приниматься всеми узлами.
- Destination Options Header используется для передачи опций, важных только для получателя. Теоретически он может встречаться в пакете два раза: первый раз для “промежуточного” получателя (если в пакете присутствует Routing Header), а второй раз для “конечного” получателя.
- No Next Header – его присутствие говорит о том, что в пакете больше ничего нет (то есть, он не несёт в себе никаких данных транспортного уровня).

Последний заголовок (если только он не No Next Header) должен указывать на заголовок транспортного уровня (то есть содержать номер соответствующего транспортного протокола, скажем TCP или UDP).

Расширенный заголовок	Тип	Описание
Hop-by-Hop Options	0	Параметры, которые должны быть обработаны каждым транзитным узлом
Destination Options	60	Параметры которые должны быть обработаны только получателем
Routing	43	Позволяет отправителю определять список узлов, которые пакет должен пройти
Fragment	44	Заголовок содержит информацию по фрагментации пакета
Authentication Header (AH)	51	Содержит информацию, используемую для аутентификацию большей части пакета
Encapsulating Security Payload (ESP)	50	Осуществляет шифрование данных для безопасных подключений

Hop-by-Hop Option Header

The Hop-by-Hop Options header is used to carry optional information that must be examined by every node along a packet's delivery path. The Hop-by-Hop Options header is identified by a Next Header value of 0 in the IPv6 header, and has the following format:



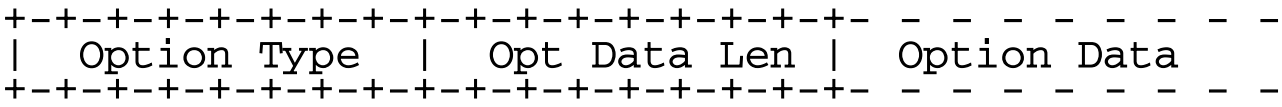
* Next Header - Identifies the type of header immediately following the Hop-by-Hop Options header. Uses the same values as the IPv4 Protocol field. (8-bit selector)

* Hdr Ext Len - Length of the Hop-by-Hop Options header in 8-octet units, not including the first 8 octets. (8-bit unsigned Integer)

* Options - Contains one or more TLV-encoded options. (Variable-length field, of length such that the complete Hop-by-Hop Options header is an integer multiple of 8 octets long.)

IPv6 Header Options

Two of the currently-defined extension headers -- the Hop-by-Hop Options header and the End-to-End Options header -- may carry a variable number of Type-Length-Value (TLV) encoded "options", of the following format:



* **Option Type** - identifier of the type of option (8-bit field)

* **Opt Data Len** - Length of the Option Data field of this option, in octets. (8-bit unsigned integer)

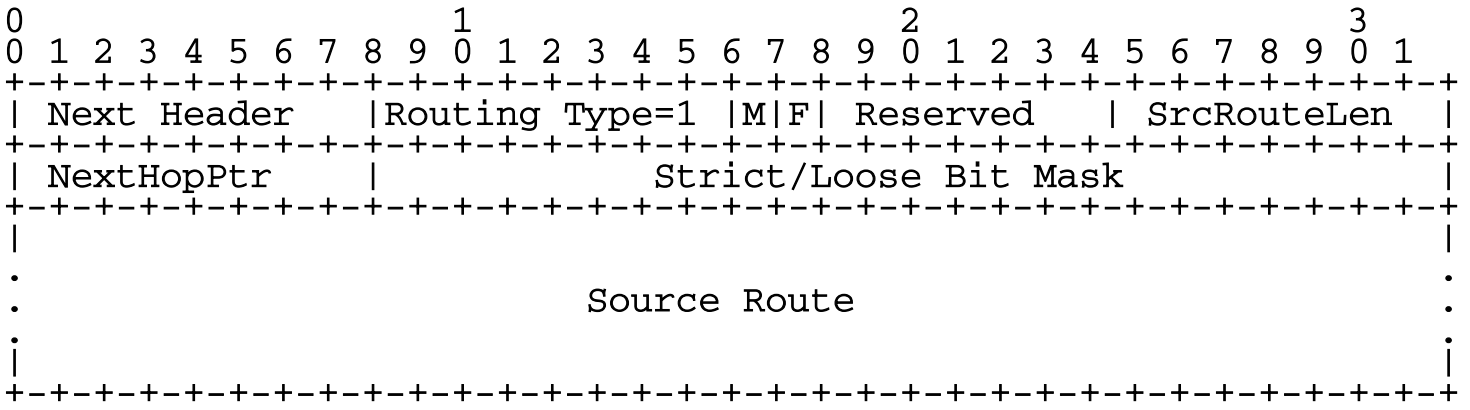
* **Option Data** - Option-Type-specific data. (Variable-length field)

The Option Type identifiers are internally encoded such that their highest-order two bits specify the action that must be taken if the processing IPv6 node does not recognize the Option Type:

00 - skip over this option and continue processing the header
01 - discard the packet
10 - discard the packet and send an ICMP Unrecognized Type message to the packet's Source Address, pointing to the unrecognized Option Type
11 - undefined.

Routing Header

The Routing header is used by an IPv6 source to list one or more intermediate nodes (or topological clusters) to be "visited" on the way to a packet's destination. This particular form of the Routing Header is designed to support SDRP. [Estrin94b]



* Next Header - Identifies the type of header immediately following the Routing Header. Uses the same values as the IPv4 Protocol field. (8-bit selector)

* Routing Type - Indicates the type of routing supported by this header. Value must be 1.

* MRE flag - Must Report Errors. If this bit is set to 1, and a router can not further forward a packet (with an incompletely traversed source route), as specified in the Source Route, the router must generate an ICMP error message. If this bit is set to 0, and a router can not further forward a packet (with an incompletely traversed source route), as specified in the Source Route, the router should not generate an ICMP error message.

* F flag - Failure of Source Route Behavior. If this bit it set to 1, it indicates that if a router can not further forward a packet (with an incompletely traversed source route), as specified in the Source Route, the router must set the value of the Next Hop Pointer field to the value of the Source Route Length field, so that the subsequent forwarding will be based solely on the destination address. If this bit is set to 0, it indicates that if a router can not further forward a packet (with an incompletely traversed source route), as specified in the Source Route, the router must discard the packet.

Routing Header (cont)

* Reserved - Initialized to zero for transmission; ignored on reception.

* SrcRouteLen - Source Route Length - Number of source route elements/hops in the SDRP Routing header. Length of SDRP routing header can be calculated from this value (length = SrcRouteLen * 16 + 8) This field may not exceed a value of 24. (8 bit unsigned integer)

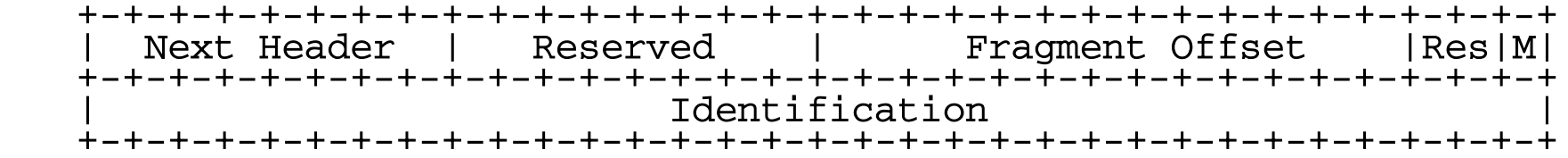
* NextHopPtr - Next Hop Pointer- Index of next element/hop to be processed; initialized to 0 to point to first element/hop in the source route. When Next Hop Pointer is equal to Source Route Length then the Source Route is completed. (8 bit unsigned integer)

* Strict/Loose Bit Mask - The Strict/Loose Bit Mask is used when making a forwarding decision. If the value of the Next Hop Pointer field is N, and the N-th bit in the Strict/Loose Bit Mask field is set to 1, it indicates that the next hop is a Strict Source Route Hop. If this bit is set to 0, it indicates that the next hop is a Loose Source Route Hop. (24 bit bitpattern)

* Source Route - A list of IPv6 addresses indicating the path that this packet should follow. A Source Route can contain an arbitrary intermix of unicast and cluster addresses. (integral multiple of 128 bits)

Fragment Header

The Fragment header is used by an IPv6 source to send payloads larger than would fit in the path MTU to their destinations. (Note: unlike IPv4, fragmentation in IPv6 is performed only by source nodes, not by routers along a packet's delivery path) The Fragment header is identified by a Next Header value of 44 in the immediately preceding header, and has the following format:



* Next Header - Identifies the type of header immediately following the Fragment header. Uses the same values as the IPv4 Protocol field. (8 bit selector)

* Reserved, Res - Initialized to zero for transmission; ignored on reception.

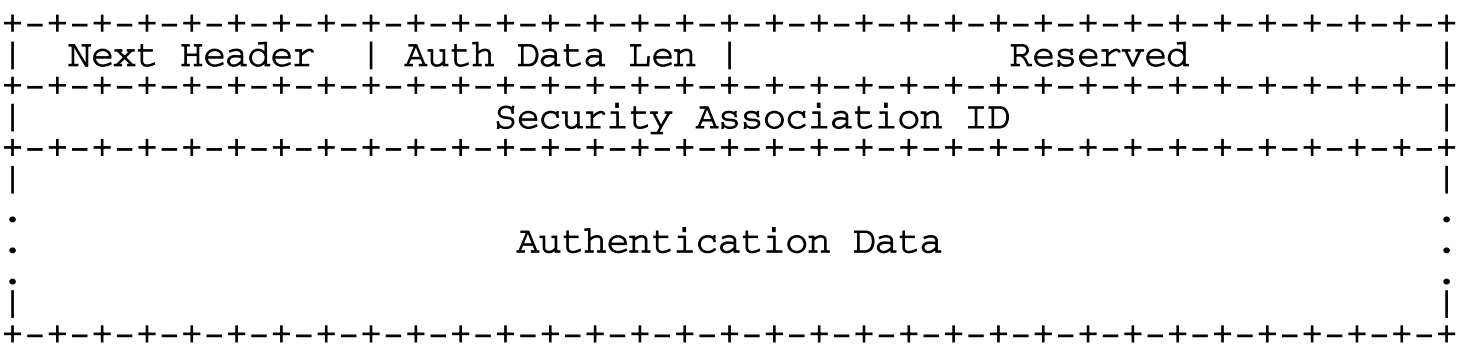
* Fragment Offset - The offset, in 8-octet units, of the following payload, relative to the start of the original, unfragmented payload. (13-bit unsigned integer)

* M flag - 1 = more fragments; 0 = last fragment.

* Identification - A value assigned to the original payload that is different than that of any other fragmented payload sent recently with the same IPv6 Source Address, IPv6 Destination Address, and Fragment Next Header value. (If a Routing header is present, the IPv6 Destination Address is that of the final destination.) The Identification value is carried in the Fragment header of all of the original payload's fragments, and is used by the destination to identify all fragments belonging to the same original payload. (32 bit field)

Authentication Header

The Authentication header is used to provide authentication and integrity assurance for IPv6 packets. Non-repudiation may be provided by an authentication algorithm used with the Authentication header, but it is not provided with all authentication algorithms that might be used with this header. The Authentication header is identified by a Next Header value of 51 in the immediately preceding header, and has the following format:



* Next Header - Identifies the type of header immediately following the Authentication header. Uses the same values as the IPv4 Protocol field. (8-bit selector)

* Auth Data Len - Length of the Authentication Data field in 8-octet units. (8-bit unsigned integer)

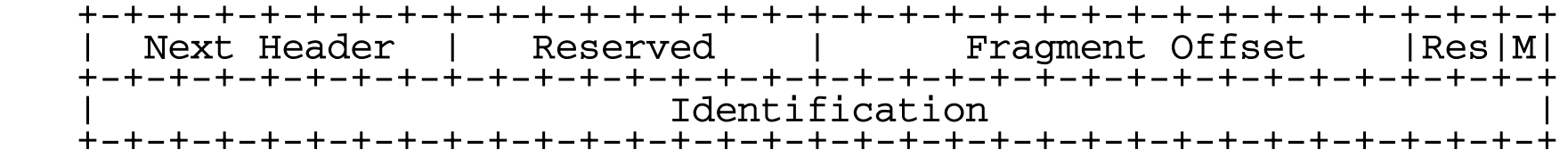
* Reserved - Initialized to zero for transmission; ignored on reception.

* Security Assoc. ID - When combined with the IPv6 Source Address, identifies to the receiver(s) the pre-established security association to which this packet belongs. (32 bit field)

* Authentication Data - Algorithm-specific information required to authenticate the source of the packet and assure its integrity, as specified for the pre-established security association. (Variable-length field, an integer multiple of 8 octets long.)

Fragment Header

The Fragment header is used by an IPv6 source to send payloads larger than would fit in the path MTU to their destinations. (Note: unlike IPv4, fragmentation in IPv6 is performed only by source nodes, not by routers along a packet's delivery path) The Fragment header is identified by a Next Header value of 44 in the immediately preceding header, and has the following format:



* Next Header - Identifies the type of header immediately following the Fragment header. Uses the same values as the IPv4 Protocol field. (8 bit selector)

* Reserved, Res - Initialized to zero for transmission; ignored on reception.

* Fragment Offset - The offset, in 8-octet units, of the following payload, relative to the start of the original, unfragmented payload. (13-bit unsigned integer)

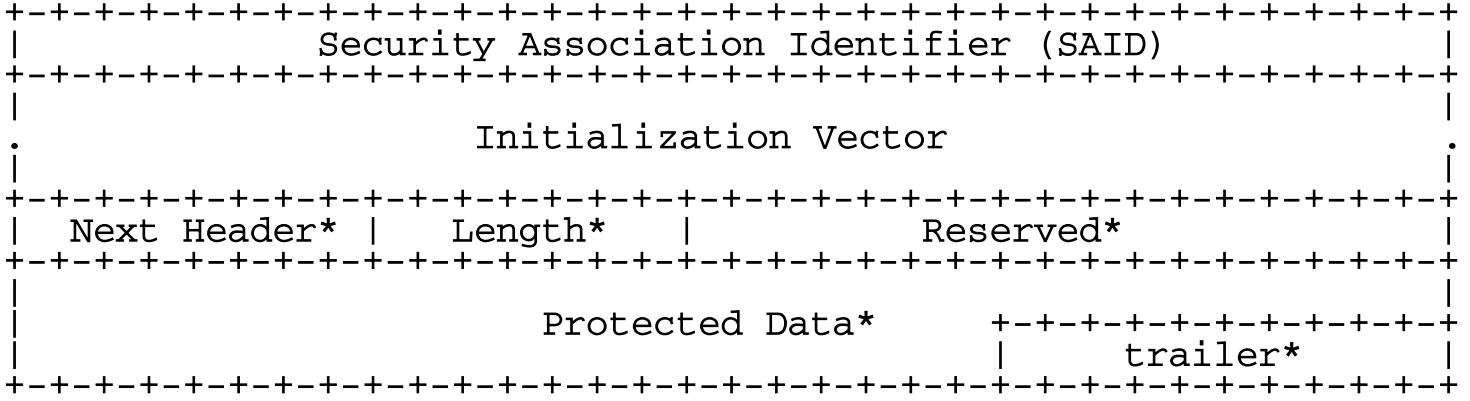
* M flag - 1 = more fragments; 0 = last fragment.

* Identification - A value assigned to the original payload that is different than that of any other fragmented payload sent recently with the same IPv6 Source Address, IPv6 Destination Address, and Fragment Next Header value. (If a Routing header is present, the IPv6 Destination Address is that of the final destination.) The Identification value is carried in the Fragment header of all of the original payload's fragments, and is used by the destination to identify all fragments belonging to the same original payload. (32 bit field)

Privacy Header

The Privacy Header seeks to provide confidentiality and integrity by encrypting data to be protected and placing the encrypted data in the data portion of the Privacy Header. Either a transport-layer (e.g., UDP or TCP) frame may be encrypted or an entire IPv6 datagram may be encrypted, depending on the user's security requirements. This encapsulating approach is necessary to provide confidentiality for the entire original datagram. If present, the Privacy Header is always the last non-encrypted field in a packet.

The Privacy Header works between hosts, between a host and a security gateway, or between security gateways. This support for security gateways permits trustworthy networks to exist without the performance and monetary costs of security, while providing security for traffic transiting untrustworthy network segments.



*encrypted

Privacy Header (cont)

* Security Association Identifier (SAID) - Identifies the security association for this datagram. If no security association has been established, the value of this field shall be 0x0000. A security association is normally one-way. An authenticated communications session between two hosts will normally have two SAIDs in use (one in each direction). The receiving host uses the combination of SAID value and originating address to distinguish the correct association. (32 bit value)

* Initialization Vector - This field is optional and its value depends on the SAID in use. For example, the field may contain cryptographic synchronization data for a block oriented encryption algorithm. It may also be used to contain a cryptographic initialization vector. A Privacy Header implementation will normally use the SAID value to determine whether this field is present and, if it is, the field's size and use. (presence and length dependent on SAID)

* Next Header - encrypted - Identifies the type of header immediately following the Privacy header. Uses the same values as the IPv4 Protocol field. (8 bit selector)

* Reserved - encrypted - Ignored on reception.

* Length - encrypted - Length of the Privacy Header in 8-octet units, not including the first 8 octets. (8-bit unsigned integer)

* Protected Data - encrypted - This field may contain an entire encapsulated IPv6 datagram, including the IPv6 header, a sequence of zero or more IPv6 options, and a transport-layer payload, or it may just be a sequence of zero or more IPv6 options followed by a transport-layer payload. (variable length)

* trailer (Algorithm-dependent Trailer) - encrypted - A field present to support some algorithms which need to have padding (e.g., to a full cryptographic block size for block-oriented encryption algorithms) or for storage of authentication data for use with a encryption algorithm that provides confidentiality without authentication. It is present only when the algorithm in use requires such a field. (presence and length dependent on SAID)

End-to-End Option Header

The End-to-End Options header is used to carry optional information that needs to be examined only by a packet's destination node(s). The End-to-End Options header is identified by a Next Header value of TBD in the immediately preceding header, and has the same format as the Hop-by-Hop Option Header except for the ability to exclude an option from the authentication integrity assurance computation.

DNS (domain name system):

IPv6 адреса представлены в DNS AAAA-адресами (или как их еще называют quad-A) (для IPv4 это были просто A-адреса).
В связи с появлением IPv6 DNS претерпел следующие изменения:
определена запись для хранения соответствия имени домена и адреса;
определен домен для разрешения адресов в имя хоста;
переопределены запросы для разрешения адресов, с учетом поддержки обоих протоколов: как IPv4, так и IPv6.
Изменения разработаны так же с учетом совместимости с уже существующим оборудованием.

Запись для хранения IPv6 адресов представляет собой AAAA запись в сетевом порядке (high-order byte first). Определен домен ip6.arpa – предназначен для разрешения IPv6 адресов, хотя может использоваться и для других целей. Запись адресов на ip6.arpa выглядит так:

```
4321:0:1:2:3:4:567:89ab -> b.a.9.8.7.6.5.0.4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.0.1.2.3.4.IP6.ARPA.
IPV6DOMAIN.EXAMPLE.NET AAAA IN 86400 128 b.a.9.8.7.6.5.0.4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.0.1.2.3.4.IP6.ARPA.
```

Взаимодействие сетей IPv4 и IPv6 (transition mechanisms):

Пока IPv6 полностью не вытеснило IPv4, требуются механизмы взаимодействия этих сетей – для доступа IPv6 хостов к сервисам IPv4, для доступа IPv6 хостов и сетей к IPv6 сетям через IPv4. Рассмотрим механизмы, которые для этого существуют на данный момент.

Основные подходы/механизмы:

- dual stack – полная поддержка обоих протоколов на хостах и рутерах (понятно, что с разделением большей части кода, так как IPv6 это все же расширение IPv4). Большая часть современных реализаций IPv6 поддерживает именно этот механизм. Ранние тестовые реализации использовали независимые стеки IPv4 и IPv6.
- Тоннелинг IPv6 через IPv4 – механизм установления p2p тоннелей с помощью энкапсуляции IPv6 пакетов с помощью IPv4 заголовка. То есть IPv4 используется в качестве сетевого уровня (link layer) для передачи IPv6.
- Автоматический тоннелинг – конечные точки тоннеля определяются автоматически инфраструктурой рутера. Наиболее рекомендуемый механизм – 6to4. Конечные точки определяются по известному IPv4 anycast адрессу на удаленной стороне и включенному в IPv6 IPv4 адрессу на локальной стороне. В качестве примера можно привести Teredo – протокол туннелинга, использующий энкапсуляцию IPv6 пакетов в IPv4 UDP датаграммы. Его экспериментальные версии установлены в Windows XP SP2 IPv6 stack. IPv6, 6to4, Teredo доступны как дефолтовые в Windows Vista.
- Конфигурируемый тоннелинг − конечные точки конфигурируются явно оператором или тоннельным брокером. Обычно полученный тоннелинг проще отлаживать, чем автоматический, потому рекомендуется для больших сетей.

ICMPv6:

Очевидно, что элементы более или менее сложных сетей должны как-то реагировать на изменение ее состояния – исчезновение и появление новых станций, колебание нагрузки, и так далее. Для этой цели применяются служебные протоколы. Некоторые из них используются непосредственно для формирования таблицы маршрутизации (протоколы маршрутизации), другие служат для управления сетью в целом. Рассмотрим один из самых примитивных и старых протоколов, протокол, который должны поддерживать все – и хосты и рутеры – ICMP (Internet Control Message Protocol).
Строго говоря, сообщения этого протокола переносятся в IP-пакетах, и поэтому ICMP в стеке протоколов должен быть на ступеньку выше IP. Однако, поскольку сервис, предоставляемый этим протоколом, имеет смысл только в контексте IP-протокола, в большинстве операционных систем реализация ICMP входит в состав IP-модуля.

- ICMP предназначен для уведомления об ошибках при маршрутизации пакета и для выявления проблем в сети.
- ICMP-сообщения передаются в IP-пакетах с протоколом, равным 1. ICMP не нуждается в гарантированной доставке, однако его сообщения защищены контрольной суммой.
- Каждое сообщение содержит тип, дополнительный код, уточняющий этот тип, и некоторую дополнительную информацию (в зависимости от типа и кода), позволяющую получателю более адекватно обработать полученное сообщение.

1) ICMPv6 был специфицирован одновременно с IPv6 (как и для IPv4, этот протокол реально является частью реализации IP).
2) В нем более четко проведено разделение на сообщения об ошибках и информационные сообщения. (В частности, номера сообщений об ошибках нумеруются от 0 до 127, а информационные – с 128 до 255.)
3) Адрес отправителя в пакете IPv6, содержащем ICMP-сообщение, должен указываться согласно следующим правилам:

- Если ICMP-сообщение посылается в ответ на сообщение, пришедшее на один из unicast-адресов узла, адрес отправителя должен быть равен этому unicast-адресу.
- Если пакет предназначался multicast или anycast группе, то адресом отправителя должен быть адрес интерфейса, принадлежащего этой группе.
- Если адрес не принадлежит узлу, отправляющему сообщение, то адресом отправителя должен быть unicast-адрес того интерфейса узла, на котором возникла проблема или ошибка, чтобы отправителю легче было их диагностировать.
- Если три первых правила не срабатывают, то в качестве адреса отправителя берется адрес интерфейса, через который пакет отправляется.

1) При получении или посылке ICMP-сообщения узел должен руководствоваться следующими правилами:

- ICMP-сообщения об ошибках неизвестного типа должны передаваться на верхний уровень.
- Информационные ICMP-сообщения неизвестного типа должны уничтожаться.
- Каждое ICMP-сообщение об ошибке должно содержать как можно большую (допустимую MTU пути) часть пакета, вызвавшего ошибку. Получатель может использовать его для диагностирования проблемы.
- Сообщение ICMP об ошибке не должно посылаться в ответ на сообщение об ошибке.
- Кроме того, при посылке сообщений об ошибках следует ограничивать их частоту, чтобы не перегружать сеть.

ICMP type	Meaning
1	Destination Unreachable
2	Packet Too Big
3	Time Exceeded
4	Parameter Problem
128	Echo Request
129	Echo Reply
130	Group Membership Query
131	Group Membership Report
132	Group Membership Reduction
133	Router Solicitation
134	Router Advertisment
135	Neighbour Solicitation
136	Neighbour Advertisment
137	Redirect

Ошибки ICMP:

0 – Destination Unreachable

По каким-либо причинам не удалось доставить пакет узлу или приложению на этом узле. Для дополнительной диагностики используются коды:

0 – сеть недоступна;

1 – доступ к получателю административно запрещен;

2 – облавть действия адреса отправителя не соответствует местонахождению получателя (например, адрес отправителя Link Local, а пакет предназначен получателю из другой подсети);

3 – хост недоступен;

4 – на станции-получателе нет приложения, ожидающего этот пакет (аналог **port unreachable** в IPv4).

1 – Packet Too Big

Посылается, когда длина пакета превышает MTU выходного интерфейса и содержит MTU (чтобы отправитель мог скорректировать длину паекта до нужных размеров). С помощью этого сообщения для IPv6 реализуется Path MTU Discobvery-алгоритм.

2 – Time Exceed Message

Посылается, если исчерпался Hop Count пакета или на получателе превышено время ожидания фрагмента датаграммы.

3 – Parameter Problem Message

Уведомляет отправиталя о невозможности корректно обработать заголовок IP-пакета и содержит указатель на место, вызвавшее проблему.

Информационные сообщения ICMP:

128, 129 – Echo Request и Echo Reply

Эти сообщения используются для определения, присутствует ли станция в сети и каково качество связи между двумя станциями.
Сообщение Echo содержит некий идентификатор, порядковый номер и (иногда) дополнительные данные. Станция, получив ICMP-сообщение Echo Request, должна отправить полученный пакет назад, не изменяя его данных, а изменив только тип с Echo Request на Echo Reply и IP-заголовок. Таким образом, можно выяснить, есть ли в сети станция с указанным адресом и включена ли она. Послав серию пакетов и оценив процент потерявшихся (его можно определить, проанализировав идентификаторы и порядковые номера вернувшихся пакетов), можно судить, насколько хороша связь между станциями. С помощью ICMP-сообщений Echo Request и Echo Reply реализована хорошо известная программа ping.

137 – Redirect

Иногда возникает ситуация, когда промежуточный маршрутизатор посылает IP-пакет в тот же сетевой интерфейс, откуда он его принял (другому маршрутизатору). В этом случае, станции, ответственной за этот крюк, посылается сообщение Redirect, говорящее о том, что ее таблица маршрутизации некорректна. Получив это сообщение, станция анализирует дополнительную информацию (заголовок и первые 64 бита исходного IP-пакета и адрес маршрутизатора) и обновляет таблицу маршрутизации.

IPv6-to-IPv4 Stateless Translation mechanisms:

При доступе строго-IPv6 хостов к строго-IPv4 сервисам требуется прямая трансляция заголовков датаграмм, т. н. **Stateless IP/ICMP Translation(SIIT) –** механизм трансляции IPv6 пакетов в IPv4 и обратно. Описан в RFC2765.

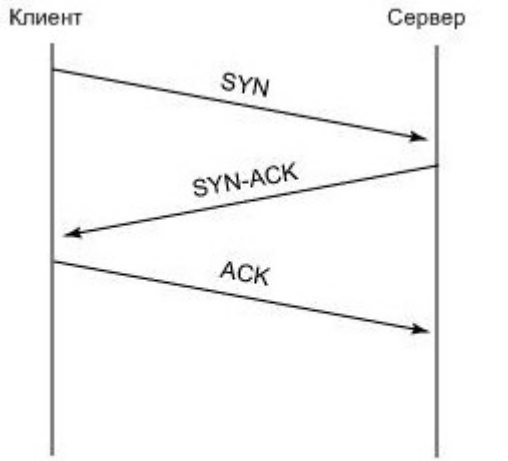
По сути SIIT описывает метод, с помощью которого рутер интерпретирует IPv4 заголовок и создает параллельный IPv6 заголовок с эквивалентной информацией и соответственно обратная операция из IPv6 в IPv4.

Протокол SCTP (RFC 3286)

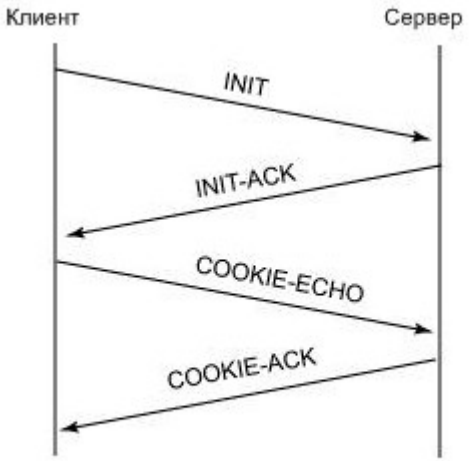
ST (RFC 1190, 1990) → ST2 (RFC 1819, 1995, IPver=5) → MPLS (2001) → SCTP (RFC 3286, 2002)

- множественная адресация;
- многопотоковая передача данных;
- безопасность устанавливаемого подключения от SYN-flood attack ;
- формирование кадров сообщений;
- настраиваемая неупорядоченная передача данных;
- поэтапное завершение работы.

4-way handshake



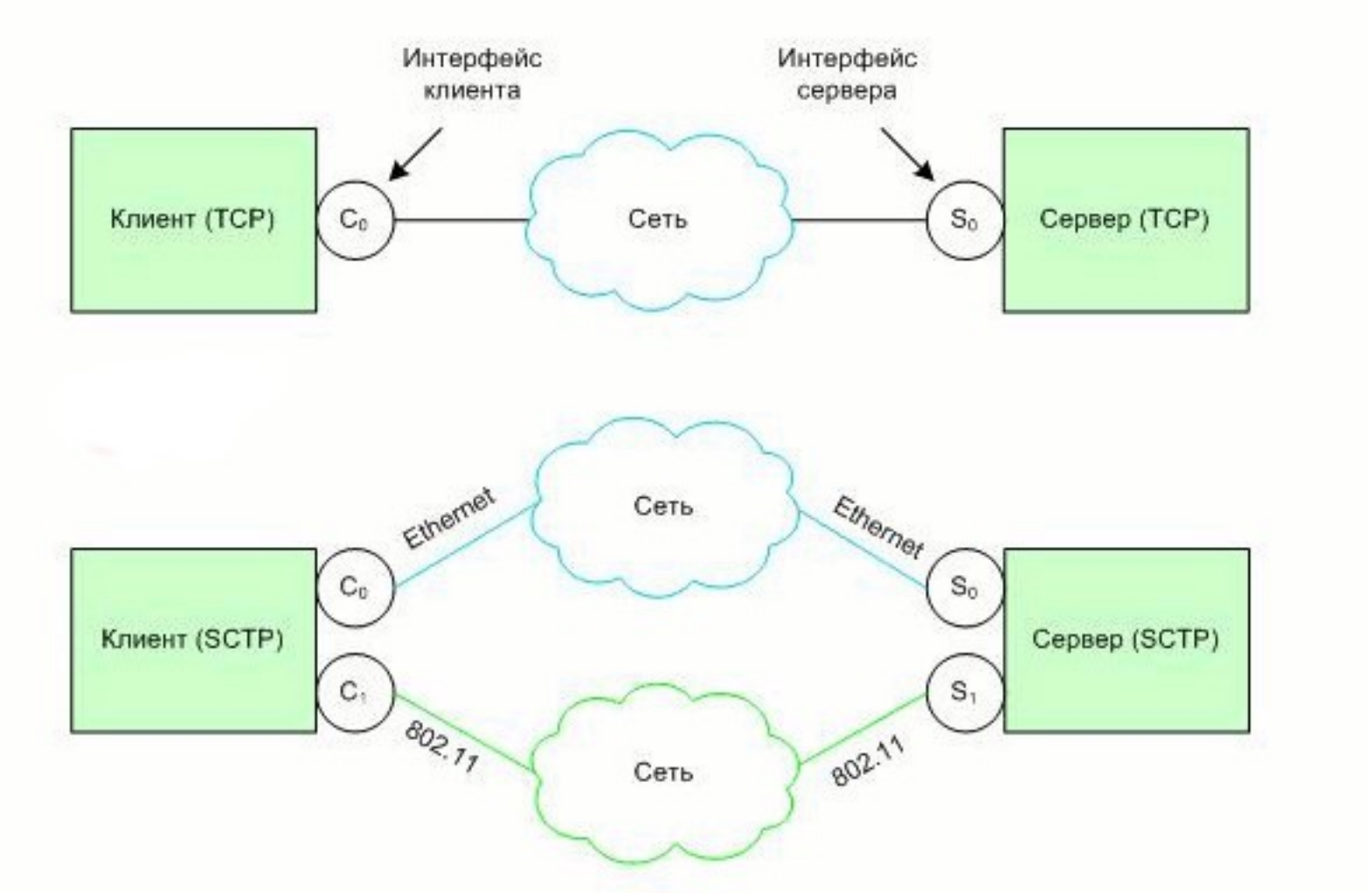
Процедура установления соединения (трехэтапное квитирование) в протоколе TCP



Процедура установления соединения (четырехэтапное квитирование) в протоколе SCTP

Синхронное завершение соединения

Многопоточная передача




Корреспондент
Поток 0
Поток 1
Поток 2
Поток 3
Корреспондент
Ассоциация протокола SCTP

Transport Layer Protocol Comparison

Параметр	UDP	TCP	SCTP
Установка соединения	No	Yes	Yes
Надежная передача	No	Yes	Yes
Сохранение границ сообщения	Yes	No	Yes
Упорядоченная доставка	No	Yes	Yes
Неупорядоченная доставка	Yes	No	Yes
Контрольные суммы данных	Yes	Yes	Yes
Размер контрольной суммы (бит)	16	16	32
Path MTU Discovery	No	Yes	Yes
Управление накоплением	No	Yes	Yes
Многопоточность	No	No	Yes
Поддержка множественных интерфейсов	No	No	Yes
Связка потоков	No	Yes	Yes

SCTP Message Format

Bits	0-7	8-15	16-23	24-31
0	Source Port		Destination Port	
32	Verification tag			
64	CRC 32			
96	Block 1 type	Block 1 flags	Block 1 Length	
128	Block 1 Data			
...				
...	Block N type	Block N flags	Block N Length	
...	Block N Data			

Verification Tag: 32 bits (unsigned integer)

The receiver of this packet uses the Verification Tag to validate the sender of this SCTP packet. On transmit, the value of this Verification Tag MUST be set to the value of the Initiate Tag received from the peer endpoint during the association initialization, with the following exceptions:

- A packet containing an INIT chunk MUST have a zero Verification Tag.
- A packet containing a SHUTDOWN-COMPLETE chunk with the T-bit set MUST have the Verification Tag copied from the packet with the SHUTDOWN-ACK chunk.
- A packet containing an ABORT chunk may have the verification tag copied from the packet which caused the ABORT to be sent. For details see Section 8.4 and 8.5.

An INIT chunk MUST be the only chunk in the SCTP packet carrying it.

Chunk Type: 8 bits (unsigned integer)

This field identifies the type of information contained in the Chunk Value field. It takes a value from 0 to 254. The value of 255 is reserved for future use as an extension field.

The values of Chunk Types are defined as follows:

ID Value	Chunk Type
0	- Payload Data (DATA)
1	- Initiation (INIT)
2	- Initiation Acknowledgement (INIT ACK)
3	- Selective Acknowledgement (SACK)
4	- Heartbeat Request (HEARTBEAT)
5	- Heartbeat Acknowledgement (HEARTBEAT ACK)
6	- Abort (ABORT)
7	- Shutdown (SHUTDOWN)
8	- Shutdown Acknowledgement (SHUTDOWN ACK)
9	- Operation Error (ERROR)
10	- State Cookie (COOKIE ECHO)
11	- Cookie Acknowledgement (COOKIE ACK)
12	- Reserved for Explicit Congestion Notification Echo (ECNE)
13	- Reserved for Congestion Window Reduced (CWR)
14	- Shutdown Complete (SHUTDOWN COMPLETE)
15 to 62	- reserved by IETF
63	- IETF-defined Chunk Extensions
64 to 126	- reserved by IETF
127	- IETF-defined Chunk Extensions
128 to 190	- reserved by IETF
191	- IETF-defined Chunk Extensions
192 to 254	- reserved by IETF
255	- IETF-defined Chunk Extensions

Chunk Type (cont)

Chunk Types are encoded such that the highest-order two bits specify the action that must be taken if the processing endpoint does not recognize the Chunk Type.

00 - Stop processing this SCTP packet and discard it, do not process any further chunks within it.

01 - Stop processing this SCTP packet and discard it, do not process any further chunks within it, and report the unrecognized parameter in an 'Unrecognized Parameter Type' (in either an ERROR or in the INIT ACK).

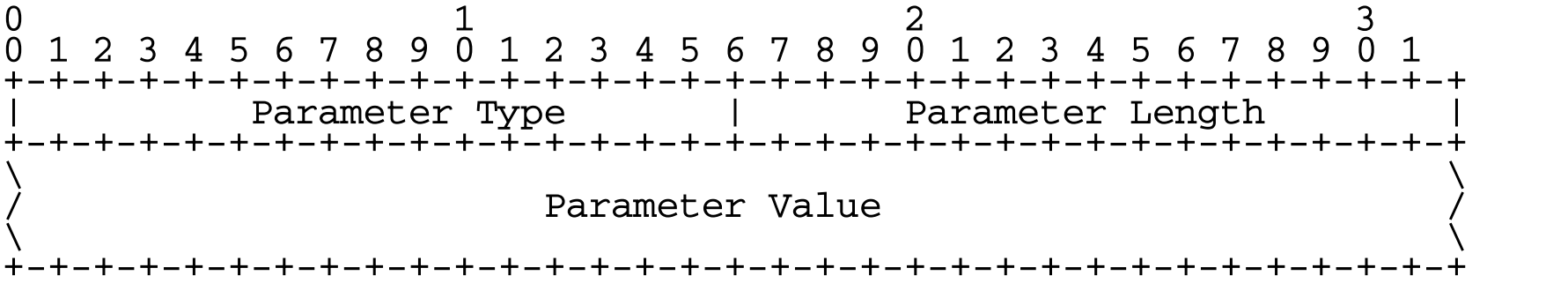
10 - Skip this chunk and continue processing.

11 - Skip this chunk and continue processing, but report in an ERROR Chunk using the 'Unrecognized Chunk Type' cause of error.

Note: The ECNE and CWR chunk types are reserved for future use of Explicit Congestion Notification (ECN).

Optional/Variable-length Parameter Format

Chunk values of SCTP control chunks consist of a chunk-type-specific header of required fields, followed by zero or more parameters. The optional and variable-length parameters contained in a chunk are defined in a Type-Length-Value format as shown below.



The Parameter Types are encoded such that the highest-order two bits specify the action that must be taken if the processing endpoint does not recognize the Parameter Type.

00 - Stop processing this SCTP packet and discard it, do not process any further chunks within it.

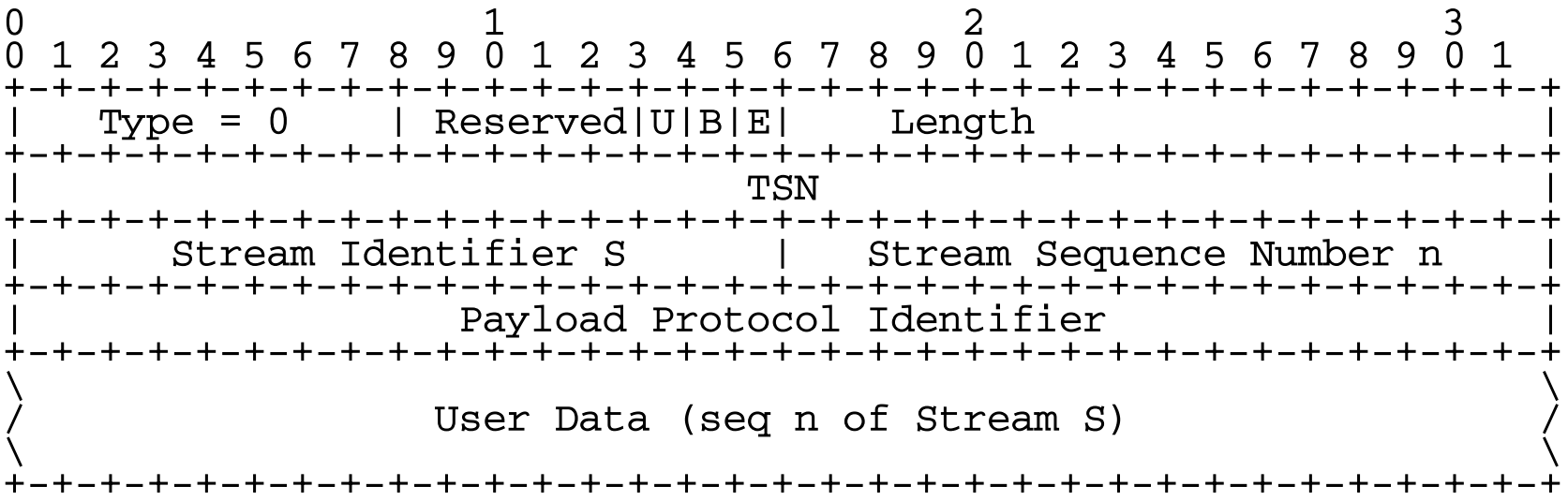
01 - Stop processing this SCTP packet and discard it, do not process any further chunks within it, and report the unrecognized parameter in an 'Unrecognized Parameter Type' (in either an ERROR or in the INIT ACK).

10 - Skip this parameter and continue processing.

11 - Skip this parameter and continue processing but report the unrecognized parameter in an 'Unrecognized Parameter Type' (in either an ERROR or in the INIT ACK).

Payload Data (DATA) (0)

The following format MUST be used for the DATA chunk:



U bit: 1 bit

The (U)nordered bit, if set to '1', indicates that this is an unordered DATA chunk, and there is no Stream Sequence Number assigned to this DATA chunk. Therefore, the receiver MUST ignore the Stream Sequence Number field.

After re-assembly (if necessary), unordered DATA chunks MUST be dispatched to the upper layer by the receiver without any attempt to re-order.

If an unordered user message is fragmented, each fragment of the message MUST have its U bit set to '1'.

B bit: 1 bit

The (B)eginning fragment bit, if set, indicates the first fragment of a user message.

E bit: 1 bit

The (E)nding fragment bit, if set, indicates the last fragment of a user message.

Payload Data (cont)

Length: 16 bits (unsigned integer)

This field indicates the length of the DATA chunk in bytes from the beginning of the type field to the end of the user data field excluding any padding. A DATA chunk with no user data field will have Length set to 16 (indicating 16 bytes).

TSN : 32 bits (unsigned integer)

This value represents the TSN for this DATA chunk. The valid range of TSN is from 0 to 4294967295 (2**32 - 1). TSN wraps back to 0 after reaching 4294967295.

Stream Identifier S: 16 bits (unsigned integer)

Identifies the stream to which the following user data belongs.

Stream Sequence Number n: 16 bits (unsigned integer)

This value represents the stream sequence number of the following user data within the stream S. Valid range is 0 to 65535.

When a user message is fragmented by SCTP for transport, the same stream sequence number MUST be carried in each of the fragments of the message.

Payload Data (cont)

Payload Protocol Identifier: 32 bits (unsigned integer)

This value represents an application (or upper layer) specified protocol identifier. This value is passed to SCTP by its upper layer and sent to its peer. This identifier is not used by SCTP but can be used by certain network entities as well as the peer application to identify the type of information being carried in this DATA chunk. This field must be sent even in fragmented DATA chunks (to make sure it is available for agents in the middle of the network).

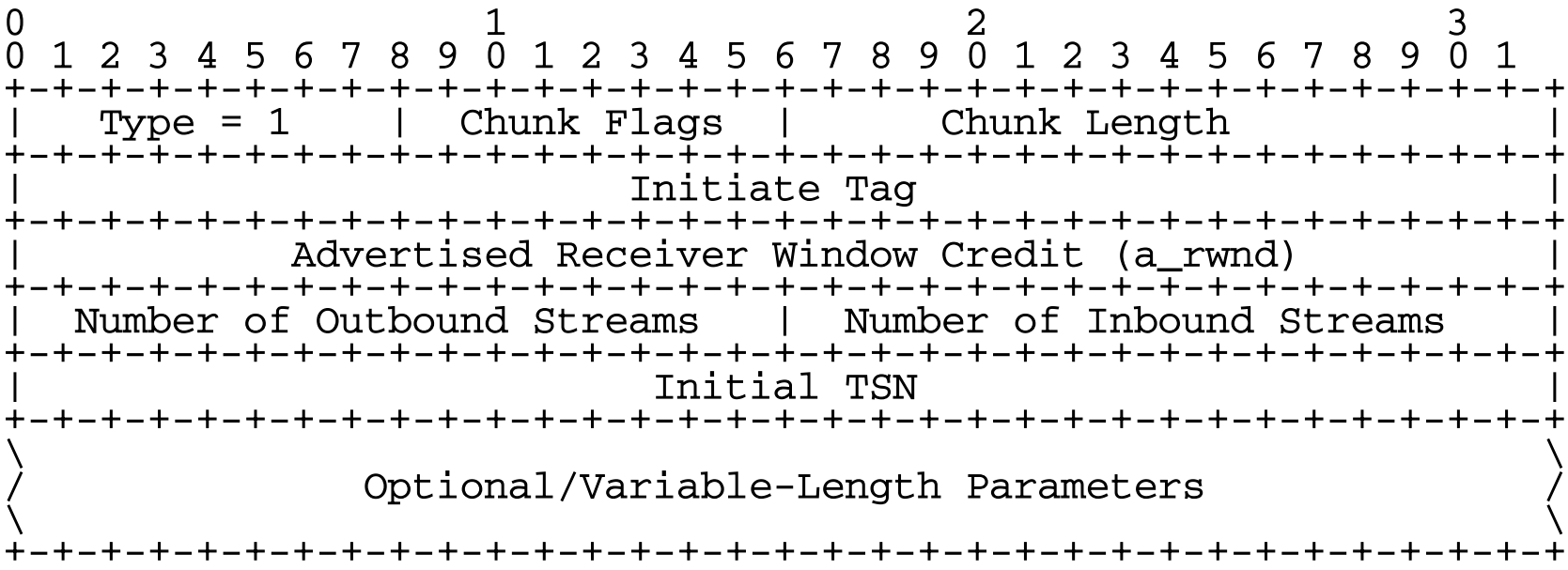
The value 0 indicates no application identifier is specified by the upper layer for this payload data.

User Data: variable length

This is the payload user data. The implementation MUST pad the end of the data to a 4 byte boundary with all-zero bytes. Any padding MUST NOT be included in the length field. A sender MUST never add more than 3 bytes of padding.

Initiation (INIT) (1)

This chunk is used to initiate a SCTP association between two endpoints. The format of the INIT chunk is shown below:



The INIT chunk contains the following parameters. Unless otherwise noted, each parameter MUST only be included once in the INIT chunk.

Initiation (cont)

Fixed Parameters	Status
Initiate Tag	Mandatory
Advertised Receiver Window Credit	Mandatory
Number of Outbound Streams	Mandatory
Number of Inbound Streams	Mandatory
Initial TSN	Mandatory

Variable Parameters	Status	Type Value
IPv4 Address (Note 1)	Optional	5
IPv6 Address (Note 1)	Optional	6
Cookie Preservative	Optional	9
Reserved for ECN Capable (Note 2)	Optional	32768 (0x8000)
Host Name Address (Note 3)	Optional	11
Supported Address Types (Note 4)	Optional	12

Note 1: The INIT chunks can contain multiple addresses that can be IPv4 and/or IPv6 in any combination.

Note 2: The ECN capable field is reserved for future use of Explicit Congestion Notification.

Note 3: An INIT chunk MUST NOT contain more than one Host Name address parameter. Moreover, the sender of the INIT MUST NOT combine any other address types with the Host Name address in the INIT. The receiver of INIT MUST ignore any other address types if the Host Name address parameter is present in the received INIT chunk.

Note 4: This parameter, when present, specifies all the address types the sending endpoint can support. The absence of this parameter indicates that the sending endpoint can support any address type.

The Chunk Flags field in INIT is reserved and all bits in it should be set to 0 by the sender and ignored by the receiver. The sequence of parameters within an INIT can be processed in any order.

Initiation (cont)

Initiate Tag: 32 bits (unsigned integer)

The receiver of the INIT (the responding end) records the value of the Initiate Tag parameter. This value MUST be placed into the Verification Tag field of every SCTP packet that the receiver of the INIT transmits within this association.

The Initiate Tag is allowed to have any value except 0. See Section 5.3.1 for more on the selection of the tag value.
If the value of the Initiate Tag in a received INIT chunk is found to be 0, the receiver MUST treat it as an error and close the association by transmitting an ABORT.

Advertised Receiver Window Credit (a_rwnd): 32 bits (unsigned integer)

This value represents the dedicated buffer space, in number of bytes, the sender of the INIT has reserved in association with this window. During the life of the association this buffer space SHOULD not be lessened (i.e. dedicated buffers taken away from this association); however, an endpoint MAY change the value of a_rwnd it sends in SACK chunks.

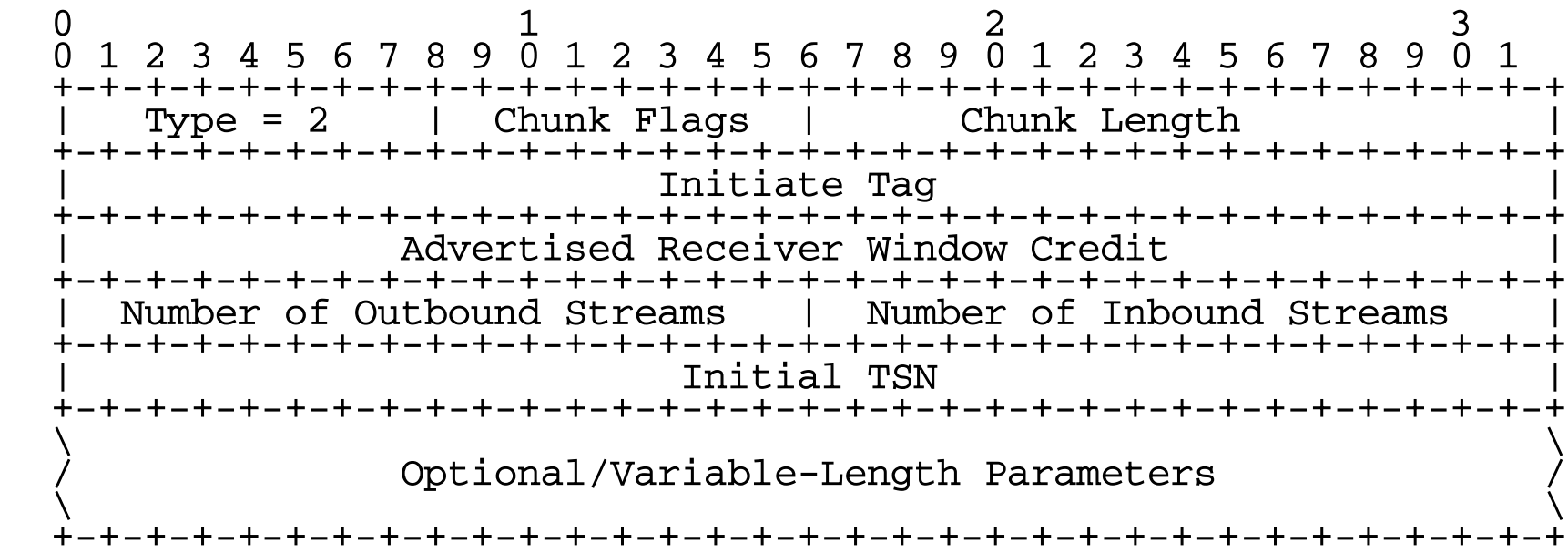
Number of Outbound Streams (OS): 16 bits (unsigned integer) (...)

Number of Inbound Streams (MIS) : 16 bits (unsigned integer) (...)

Initial TSN (I-TSN) : 32 bits (unsigned integer)

Defines the initial TSN that the sender will use. The valid range is from 0 to 4294967295. This field MAY be set to the value of the Initiate Tag field.

Initiation Acknowledgement (INIT ACK) (2):



Initiate Tag: 32 bits (unsigned integer)

The receiver of the INIT ACK records the value of the Initiate Tag parameter. This value MUST be placed into the Verification Tag field of every SCTP packet that the INIT ACK receiver transmits within this association. (…)

Advertised Receiver Window Credit (a_rwnd): 32 bits (unsigned integer)

This value represents the dedicated buffer space, in number of bytes, the sender of the INIT ACK has reserved in association with this window. During the life of the association this buffer space SHOULD not be lessened (i.e. dedicated buffers taken away from this association).

Number of Outbound Streams (OS): 16 bits (unsigned integer)(...)

Number of Inbound Streams (MIS) : 16 bits (unsigned integer)(...)

Initial TSN (I-TSN) : 32 bits (unsigned integer)

Defines the initial TSN that the INIT-ACK sender will use. The valid range is from 0 to 4294967295. This field MAY be set to the value of the Initiate Tag field.

Fixed Parameters	Status
Initiate Tag	Mandatory
Advertised Receiver Window Credit	Mandatory
Number of Outbound Streams	Mandatory
Number of Inbound Streams	Mandatory
Initial TSN	Mandatory

Variable Parameters	Status	Type Value
State Cookie	Mandatory	7
IPv4 Address (Note 1)	Optional	5
IPv6 Address (Note 1)	Optional	6
Unrecognized Parameters	Optional	8
Reserved for ECN Capable (Note 2)	Optional	32768 (0x8000)
Host Name Address (Note 3)	Optional	11

Note 1: The INIT ACK chunks can contain any number of IP address parameters that can be IPv4 and/or IPv6 in any combination.

Note 2: The ECN capable field is reserved for future use of Explicit Congestion Notification.

Note 3: The INIT ACK chunks MUST NOT contain more than one Host Name address parameter. Moreover, the sender of the INIT ACK MUST NOT combine any other address types with the Host Name address in the INIT ACK. The receiver of the INIT ACK MUST ignore any other address types if the Host Name address parameter is present.

(...)

Спасибо за внимание!