Xpress Transfer Protocol Tutorial

Tim Strayer
Bert Dempsey
Alfred Weaver

Computer Science Report No. TR-93-04
January 7, 1993

The Xpress Transfer Protocol

Alfred C. Weaver
W. Timothy Strayer
Bert J. Dempsey

Computer Networks Laboratory
Department of Computer Science
University of Virginia

Outline

Motivation, justification, and history of XTP

Protocol procedures, packet structures, and packet formats

Network addressing, encapsulation, and multicast

The XTP Engine

Discussion

Background

What	design, implement, and evaluate a new transport protocol
Who	international group of researchers and implementors in the U.S., Canada, and Europe
Why	traditional transport protocols designed for an era when bandwidth was low and error rates were high
Why now	fiber optics, FDDI, gigabit networks, advances in VLSI technology, etc., represent a fundamental paradigm shift for transport protocol designers

Participants

AMD
Artel/NASA
Boeing
Concurrent
Crosfield Electronics
DY-4
E-Systems
IBM
Intel
Intergraph
Interphase
Linotype-Hell AG
Lockheed
SBE
Scitex
Unisys
Xerox

Participants

Admiralty Research Establishment
Bowman Gray School of Medicine
Concordia University
Electronics and Telecommunications Research Institute
Laboratoire de Genie Informatique
Johns Hopkins University
National Center for Supercomputer Applications
Naval Ocean Systems Center
Naval Surface Warfare Center
Superconducting Supercollider Lab
University of Kentucky
University of Massachusetts
University of Melbourne
University of Virginia

Group Leader

Dr. Greg Chesson

Chief Scientist, R&D Division
Silicon Graphics Inc.

Mandate

(1) to understand the transport problem in a fundamental way by examining what functionality and performance must be provided by a modern transport protocol

(2) to devise a new transport protocol which could meet those objectives

(3) to make the design public in support of open systems, and to reflect the comments, suggestions, and recommendations of the larger protocol community

(4) to implement the protocol in software (for versatility and experimentation) and in hardware (for performance)

Network Subsystem Processing

Conventional approach is to make the host support both application programs and network processing

Alternative is to move network processing onto a front-end processor

Network processing includes

- protocol processing (multiple layers)
- data movement
- buffer management
- operating system interface
- application program interface
- transmitting to and receiving from the physical network
- device handling

Caveats

Off-host network processing is not a guaranteed win

Considerations:

- speed of backplane bus
- front-end processor's contention for bus
- number of data copies required to move information
- number of hardware interrupts generated
- shared memory vs. message passing architecture
- efficiency of host operating system
- subtle interactions when using a *device driver* to control a peripheral

Distributed Systems

The future lies with *distributed systems*

New services are demanded

- rapid request/response operations
- reliable datagrams
- multicast
- latency control
- intra-protocol scheduling
- more flexible error control
- high performance (high throughput, low latency)

Performance

Performance requirements are application-dependent

Trend is toward sub-millisecond delivery of *transport*-layer services — not just *datalink*-layer services

More integration of communications services with the application process

— Conclusion: we were more likely to succeed if we designed a new protocol rather than tinkered with an older one

New effort is called *Xpress Transfer Protocol (XTP)*

No Religious Arguments!

No intention to belittle previous efforts

No religious argument for or against extant protocols

Designing a new transport protocol is primarily an academic challenge

If successful, then there are secondary commercial possibilities

Goal is to reuse what exists and is good, and to augment those ideas with new services and mechanisms

Influential Protocols

XTP acknowledges the work which has gone before:

Transmission Control Protocol (TCP)
ISO Transport Protocol (TP4)
Delta-t
NETBLT
GAM-T-103
Versatile Message Transaction Protocol (VMTP)
Universal Receiver Protocol (URP)
Datakit

Design Philosophy

(1) acknowledge and reuse extant good ideas

(2) augment previous work with new services

(3) provide a set of orthogonal protocol options

(4) separate policy from mechanism

(5) enable fast implementation through intelligent design

(6) encourage VLSI implementation

Functional Advantages

- large data pipeline size
- rate and burst control
- rich priority structure
- out-of-band data
- selectable error control options
- selectable flow control options
- policy vs. mechanism
- multicast

Performance Advantages

- header/trailer protocol
- fixed-length, fixed-position control fields
- connection setup
- address translation
- transfer layer architecture
- retransmission
- acknowledgement control
- data alignment
- VLSI implementation

XTP: The Xpress Transfer Protocol

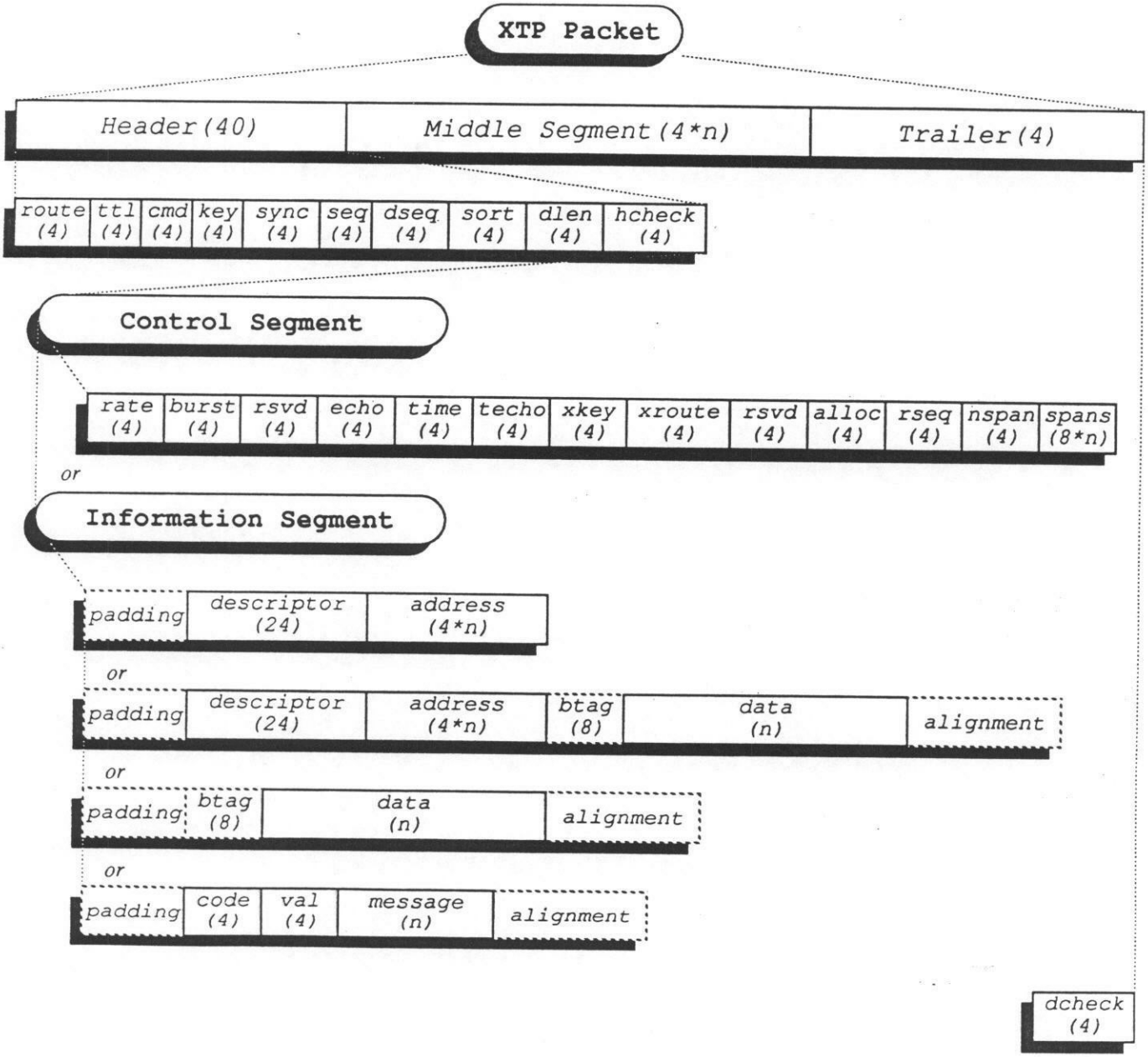
Book's table of contents:

- Foundations of the Xpress Transfer Protocol
- Network Concepts and the XTP Architectural Model
- Influential Protocols
- Protocol Procedures
- Packet Structures
- Packet Formats
- Addressing and Encapsulation
- Multicast
- The XTP Engine

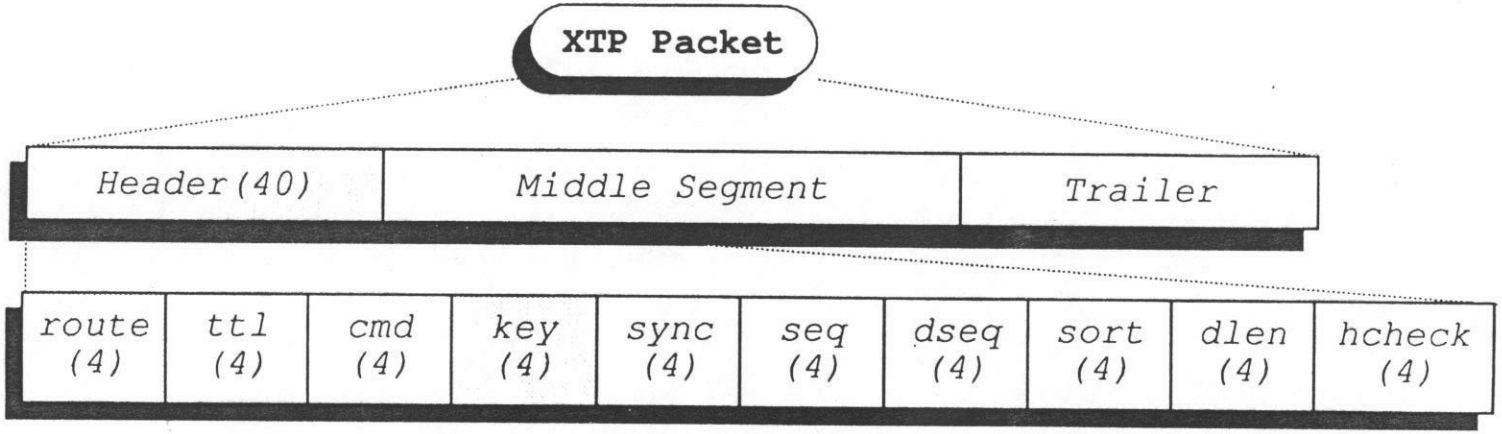
Protocol Procedures

Overview of XTP Packet Structure
Overview of Packet Options
Overview of Packet Formats
Association Management Procedures
Path Management Procedures
Data Transfer Procedures
Flow Control Procedures
Rate Control Procedures
Error Control Procedures

Packet Structure Overview



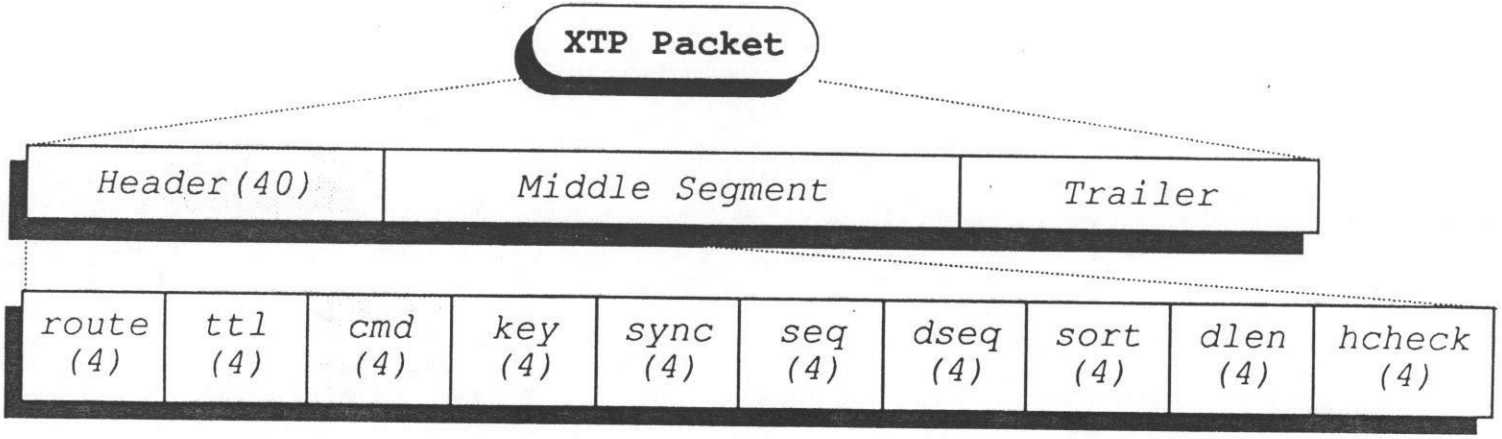
Packet Header



Specifies How To Process A Packet:

- Steering Information
- Format and Service Options
- Length and Sequencing
- Priority Information
- Validity Checks

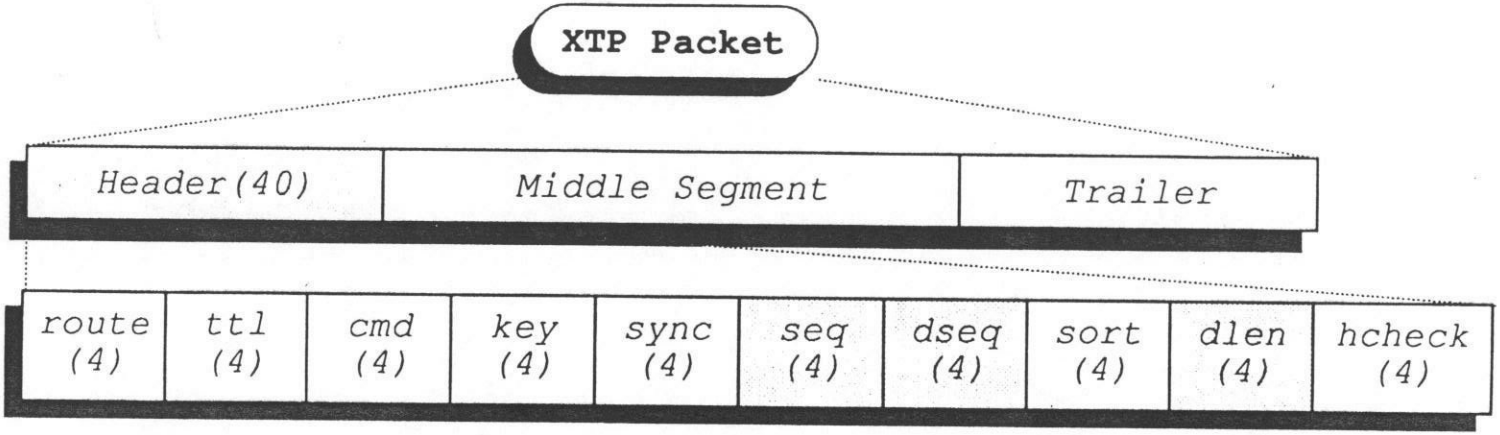
Packet Header



Specifies How To Process A Packet:

- Steering Information
- Format and Service Options
- Length and Sequencing
- Priority Information
- Validity Checks

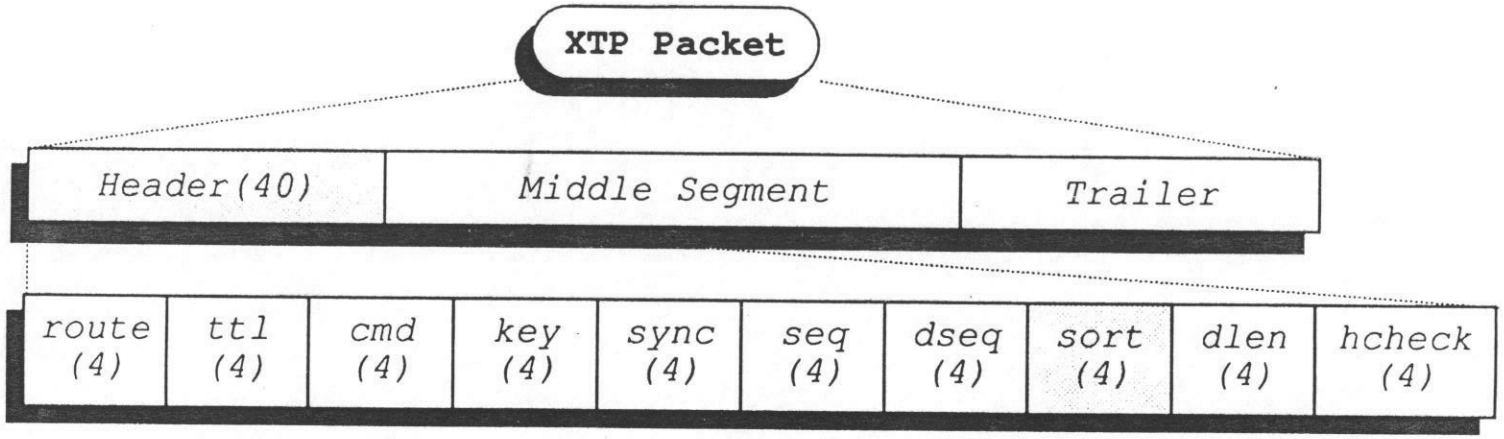
Packet Header



Specifies How To Process A Packet:

- Steering Information
- Format and Service Options
- Length and Sequencing
- Priority Information
- Validity Checks

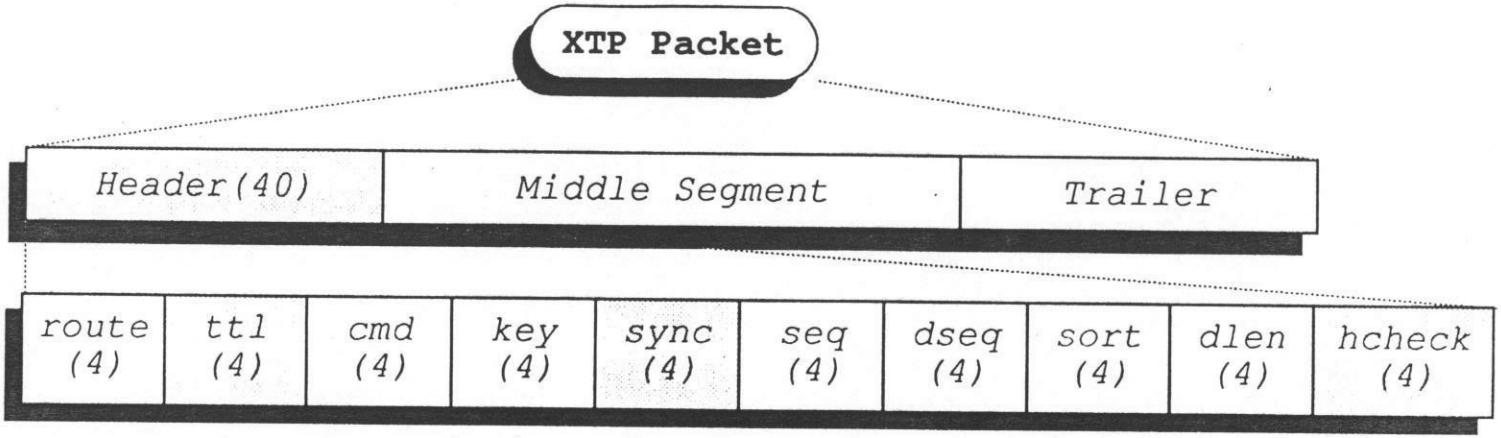
Packet Header



Specifies How To Process A Packet:

- Steering Information
- Format and Service Options
- Length and Sequencing
- Priority Information
- Validity Checks

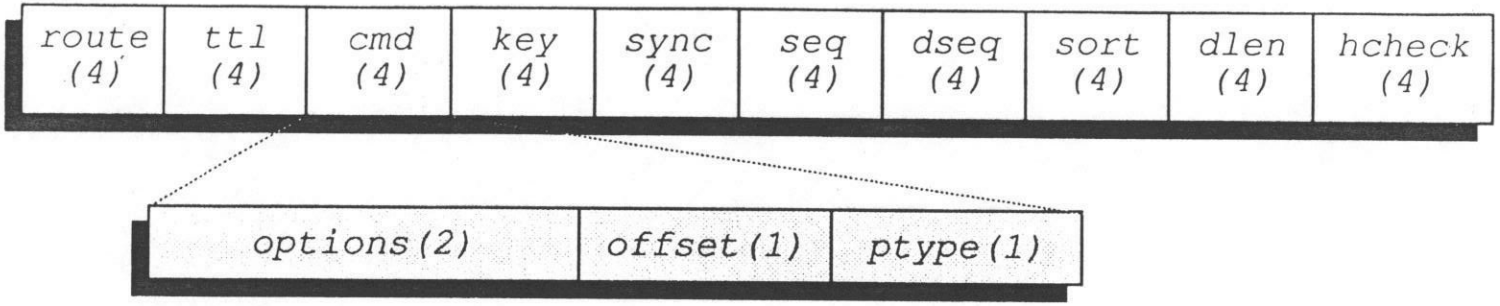
Packet Header



Specifies How To Process A Packet:

- Steering Information
- Format and Service Options
- Length and Sequencing
- Priority Information
- Validity Checks

Packet Header—The Command Field



Options field used to:

- Modify protocol processing
- Enable/disable various protocol processing modes
- Indicate the presence of some information
- Direct packet's receiver to do something

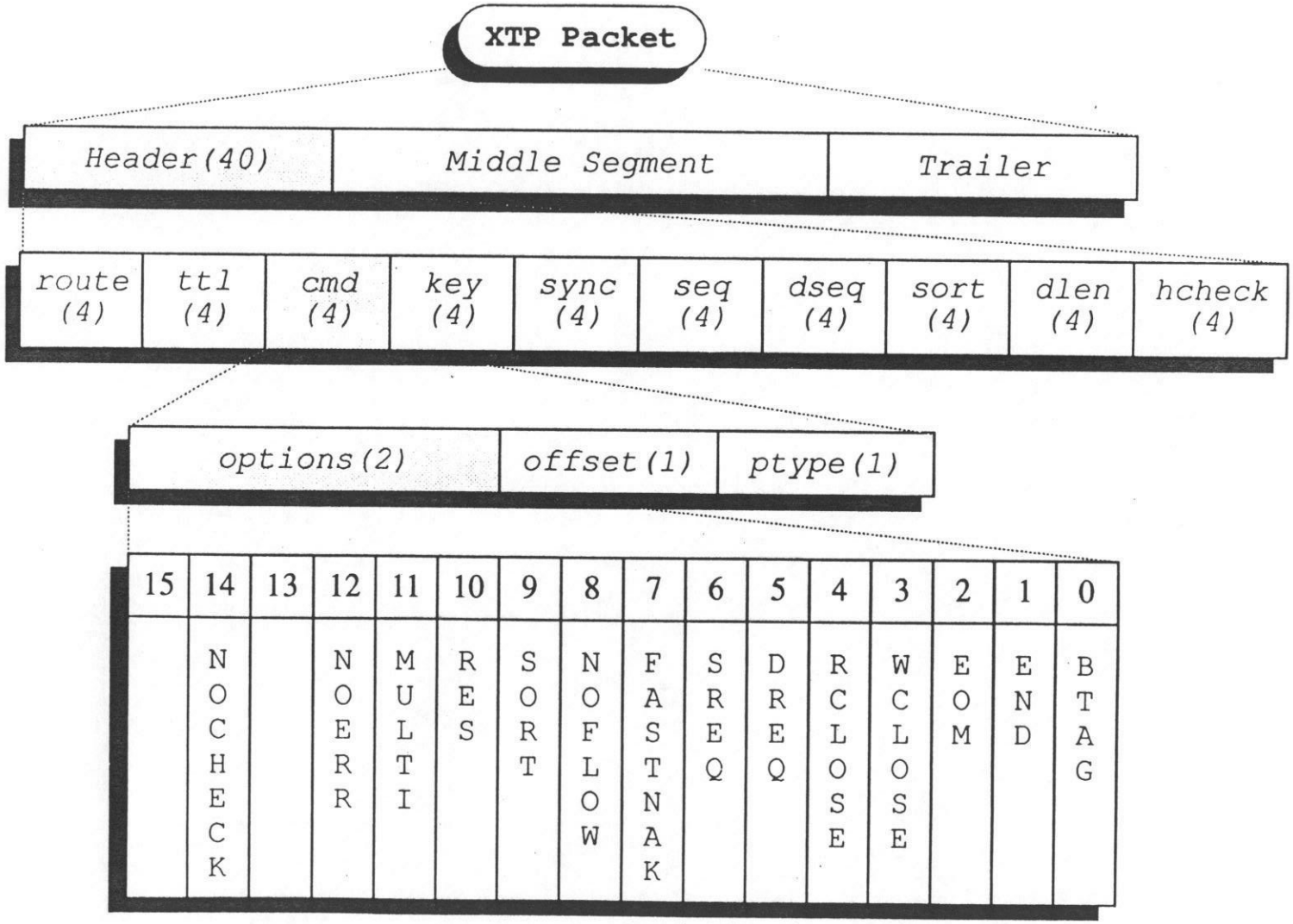
Offset field

- Information segment starting point

Ptype field

- Packet type information
- Version number
- Packet format

Command Field—Options



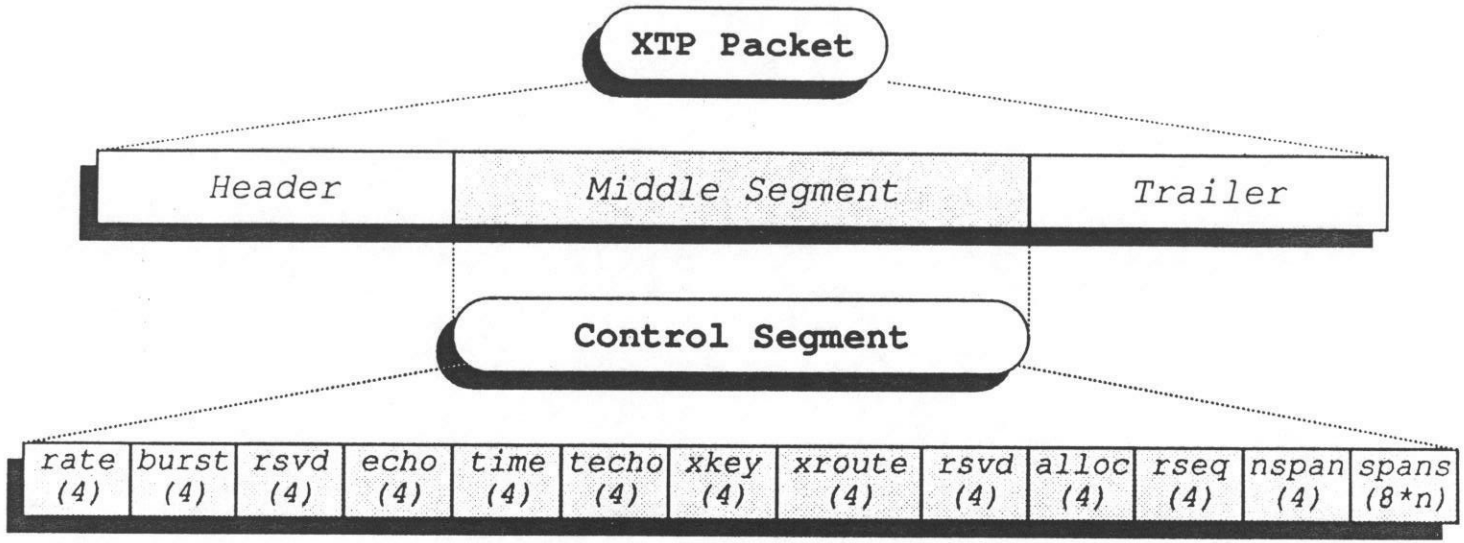
Protocol Options

NOCHECK	Turns off checksum over Middle Segment
NOERR	Turns off error control (no-error mode)
MULTI	Indicates multicast association
RES	Indicates reservation mode
SORT	Indicates the presence of an ordering value
NOFLOW	Turns off flow control (no-flow mode)
FASTNAK	Indicates fast negative acknowledgement policy
SREQ	Status requested immediately
DREQ	Status requested after data has been delivered
RCLOSE	Signals close of READER process
WCLOSE	Signals close of WRITER process
EOM	Marks end of message
END	Indicates end of association
BTAG	Indicates the presence of out-of-band data

Packet Formats

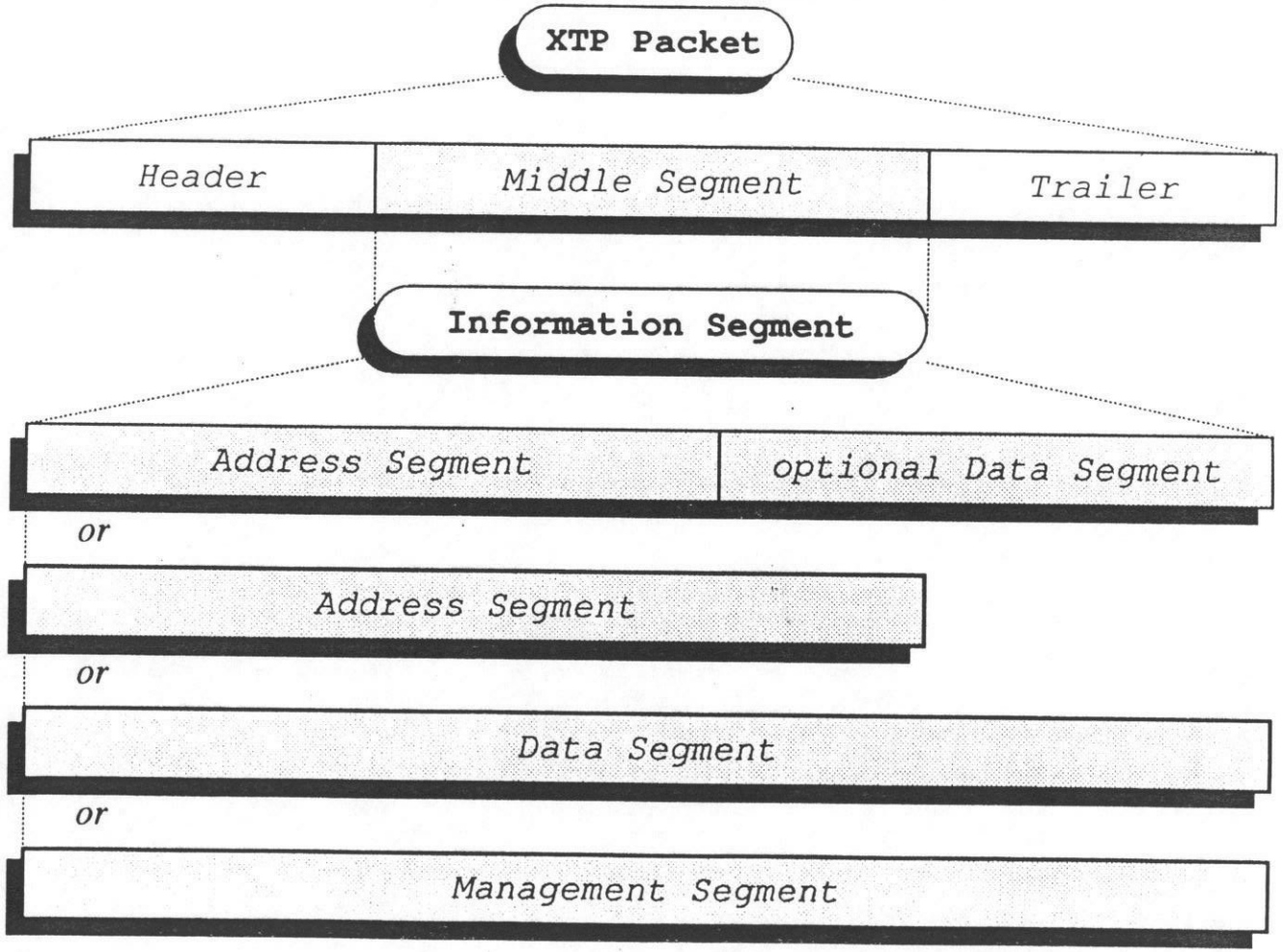
FIRST	Used to establish an association, may carry data
DATA	Carries user data only
CNTL	Carries control information and status
PATH	Used to rethread a path
DIAG	Used for fault notification
MAINT	Used for network maintenance (not yet defined)
MGMT	Used for network management (not yet defined)
ROUTE	Used to release a path
RCNTL	Used to exchange path-specific information

Middle Segment — Control

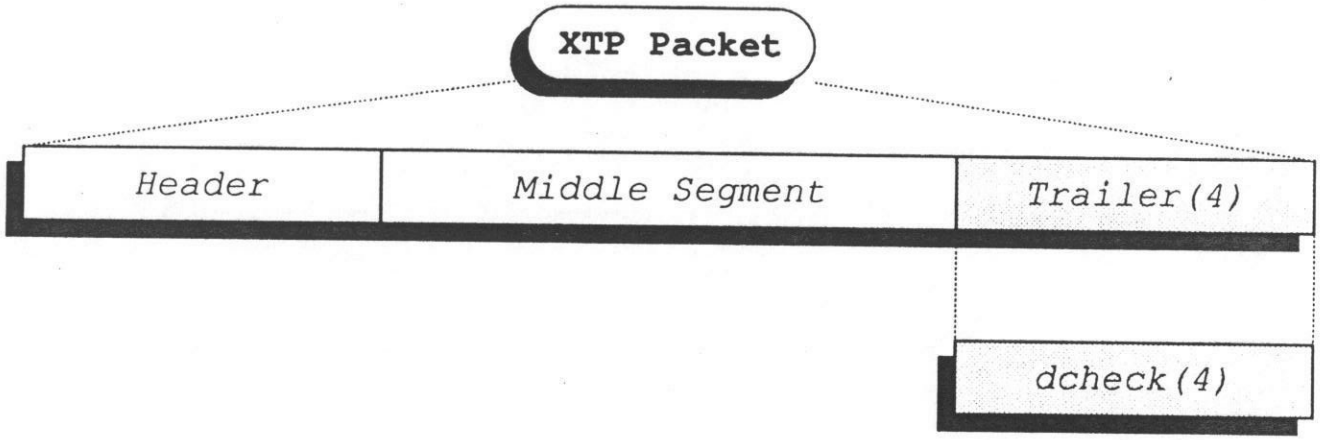


- Flow control parameters
- Rate control parameters
- Error control parameters
- Other protocol state information

Middle Segment — Information



Packet Trailer



- Contains only a validity check over middle segment
- A 32-bit concatenation of two 16-bit checksums
  - — first two bytes: straight XOR
  - — second two bytes: rotated XOR
- Notice position
  - — at end of packet for "flow-through processing"

Protocol Procedures

- Association Management Procedures
  - Establishment
  - Maintenance
  - Termination
- Path Management Procedures
  - Establishment
  - Maintenance
  - Release
- Data Transfer Procedures
- Control Procedures
  - Flow Control
  - Rate Control
  - Error Control

Association Management

Association Establishment

- Context State Machine
- Key and Route Values
- Packet Exchange

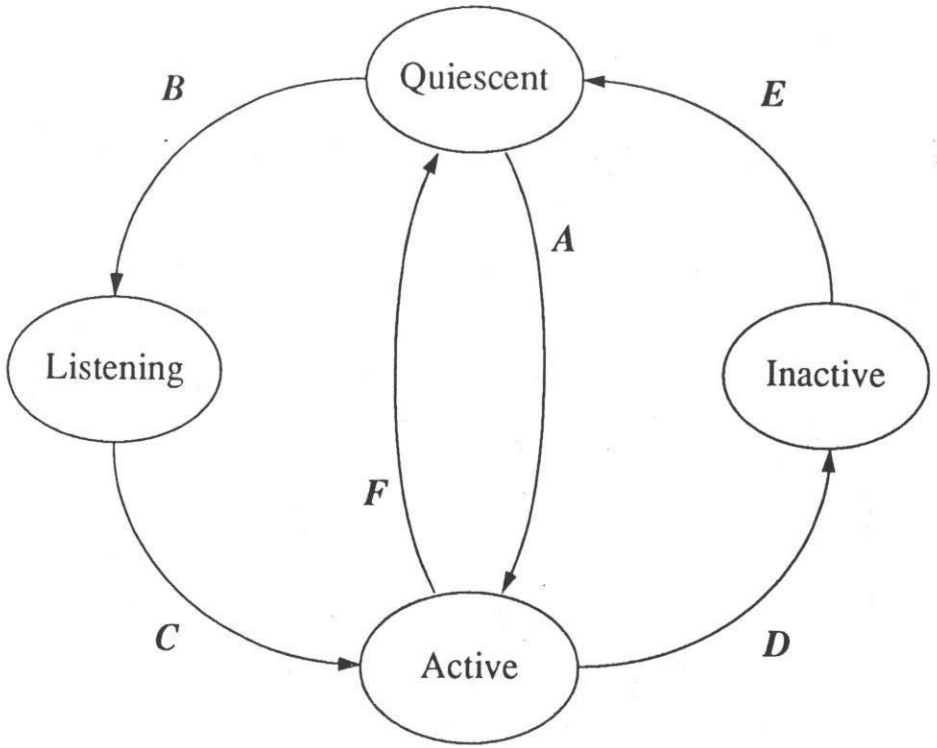
Association Maintenance

- Full Context Lookup Procedure
- Use of Return Key
- Exchange Key Procedure

Association Termination

- Association Termination State Machine
- Independent Graceful Close
- Abbreviated Graceful Close
- Forced Close
- Abort

Context State Machine



A: Context activated—caused by an **output** command

B: Context "listens" for FIRST packet—caused by an **input** command

C: Context activated—caused by receipt of FIRST packet

D: Close of READER and WRITER

E: Association terminated

F: Association aborted

- Each context starts in Quiescent state
- Context moves to Active state as commands are issued and packets exchanged

Key Values and Route Values

A *key* value identifies a context

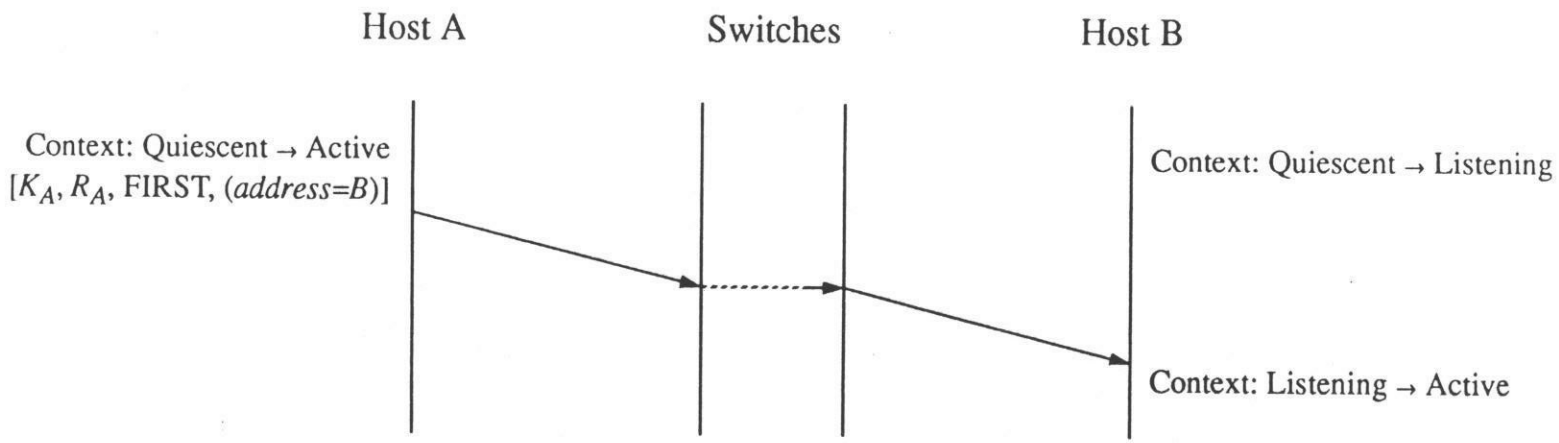
- Each *key* value created within a host must be unique
- The FIRST packet carries the *key* value associated with the context sending the FIRST packet (the Initiating Context)
- Subsequent packets from sender also carry the *key* value

A *route* value identifies the path

- Each *route* value at an endpoint or switch must be unique
- The FIRST packet carries the route value created by the sender
- At the first switch, this *route* value is replaced by a *route* value created at the switch
- The path, then, is identified by the ordered set of *route* values between the two endpoints

The Initiated Context uses the *key* value and *route* value from the FIRST packet instead of an explicit address when sending packets in the return direction

Association Establishment Packet Exchange



Association Management

Association Establishment

- Context State Machine
- Key and Route Values
- Packet Exchange

Association Maintenance

- Full Context Lookup Procedure
- Use of Return Key
- Exchange Key Procedure

Association Termination

- Association Termination State Machine
- Independent Graceful Close
- Abbreviated Graceful Close
- Forced Close
- Abort

Association Maintenance

Must allow for subsequent packet exchanges after FIRST packet:

— *How to map packets to their destination context*

— *How to send packets in the return direction*

— *How to make the mappings more efficient*

Full Context Lookup

- Most general mapping of a packet to a context
- Unambiguous packet identification

Return Key

- What to put in the *key* field of packets sent in return direction

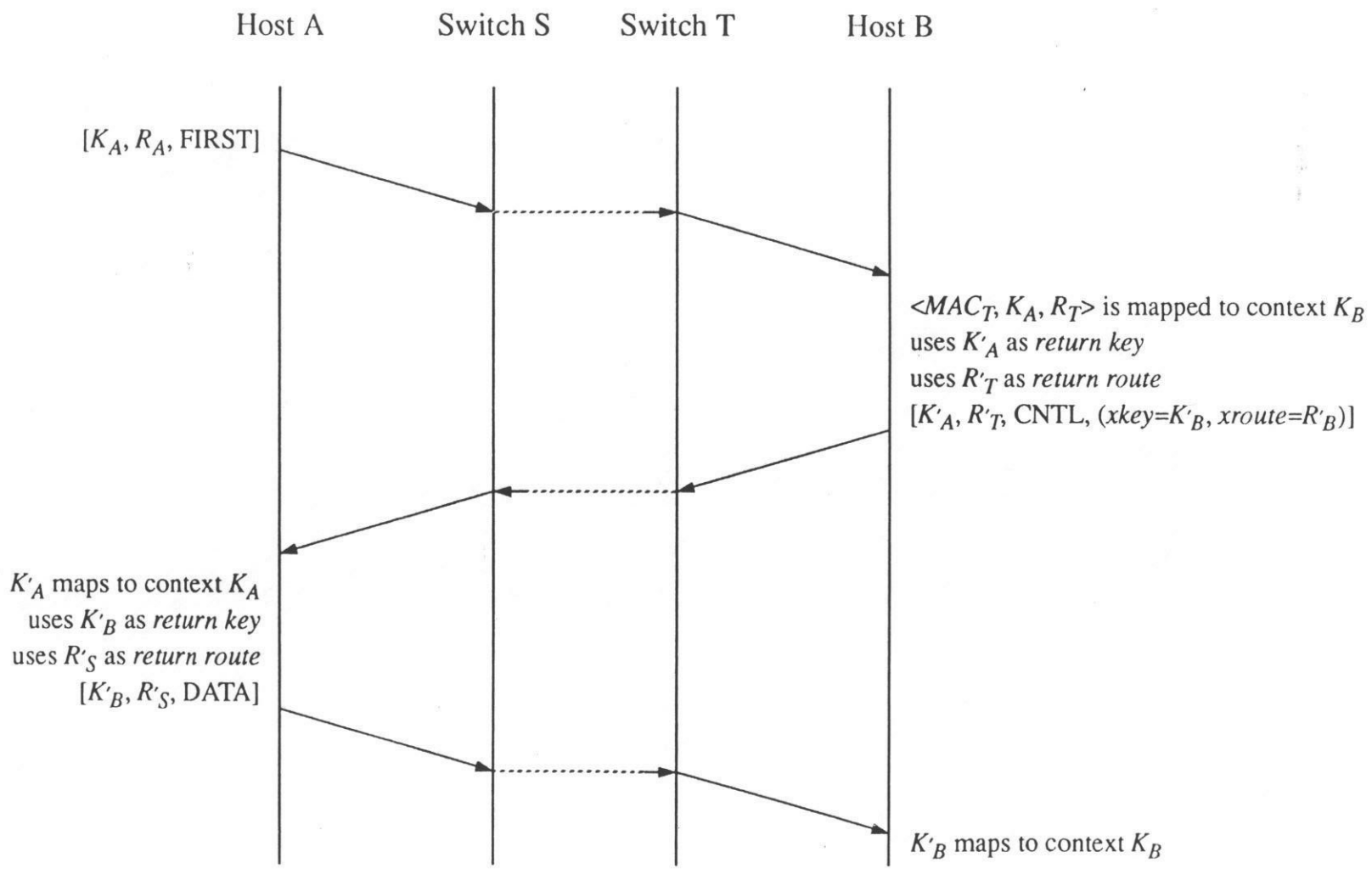
Key Exchange Procedure

- Initiated Context tells the Initiating Context which *key* value to use

Abbreviated Context Lookup

- Only possible with *return key* and after a key exchange

Key (and Route) Exchange Packet Exchanges



Association Management

Association Establishment

- Context State Machine
- Key and Route Values
- Packet Exchange

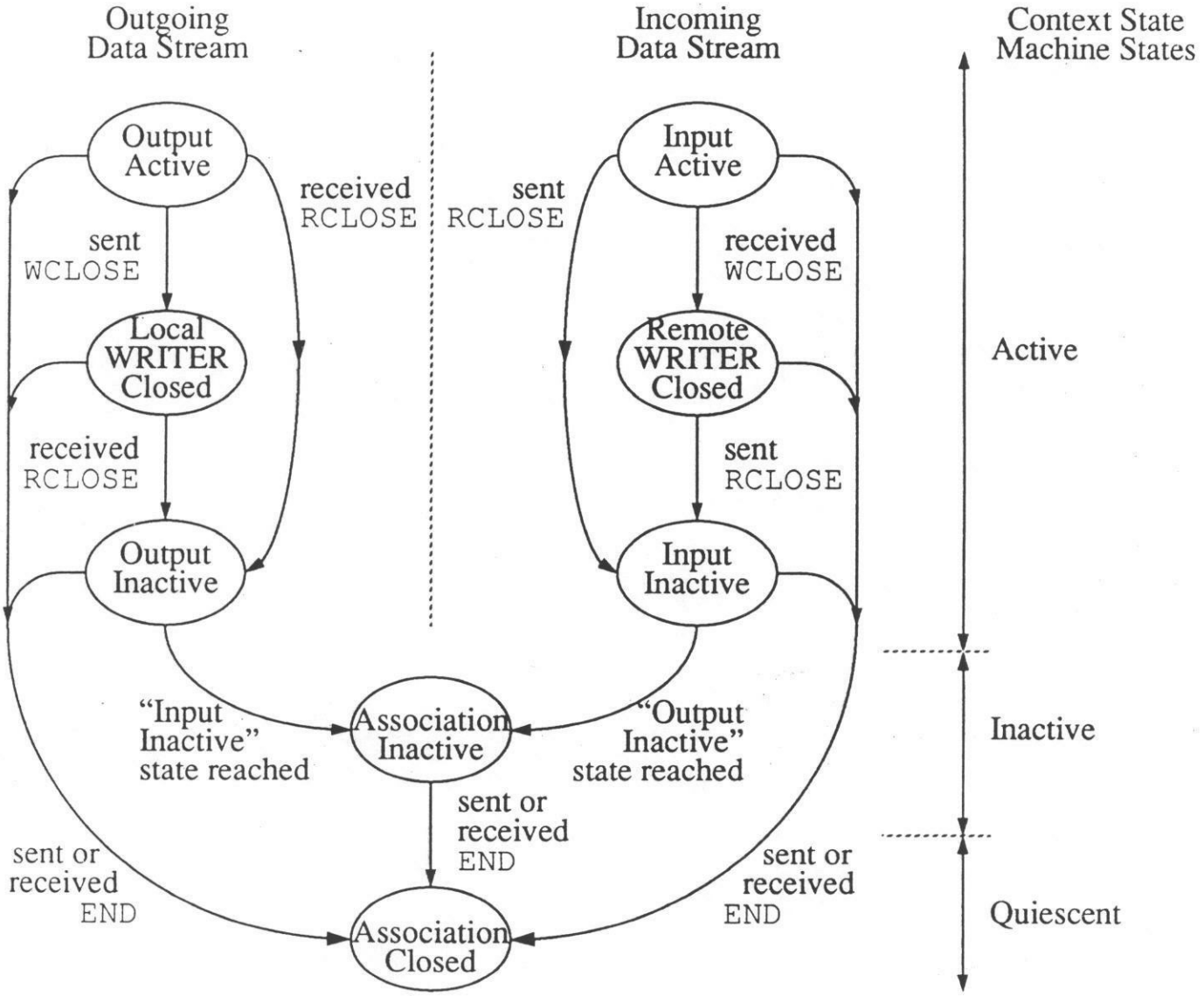
Association Maintenance

- Full Context Lookup Procedure
- Use of Return Key
- Exchange Key Procedure

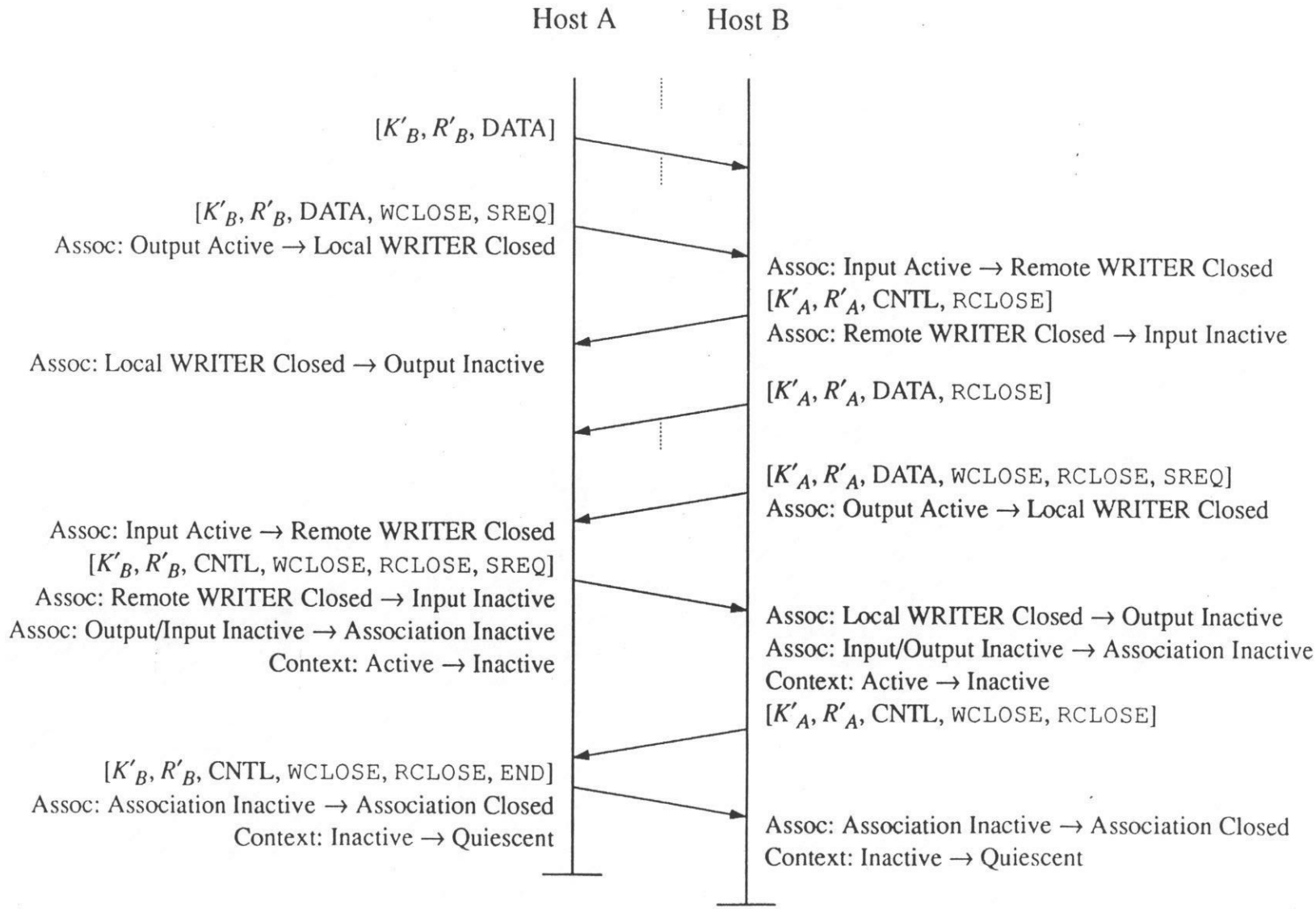
Association Termination

- Association Termination State Machine
- Independent Graceful Close
- Abbreviated Graceful Close
- Forced Close
- Abort

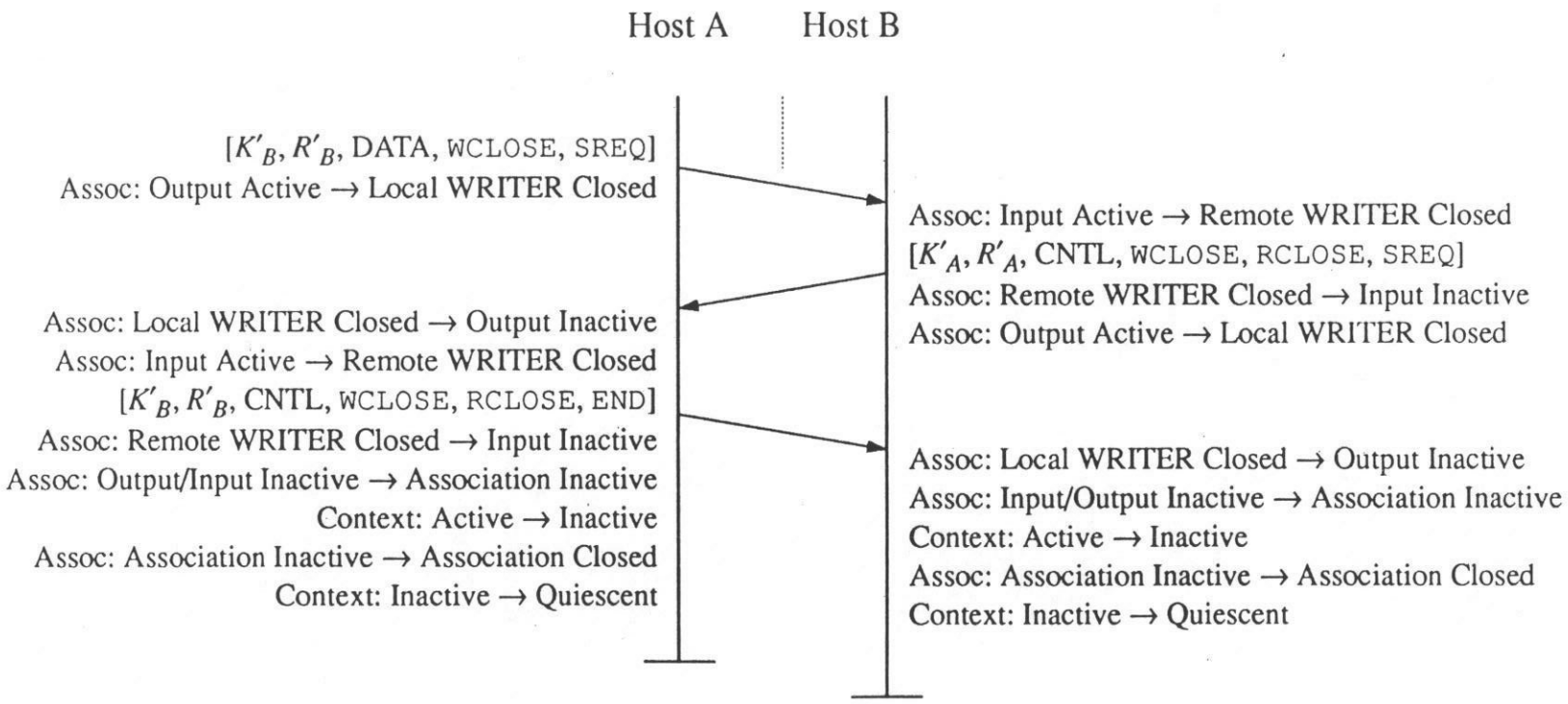
Association Terminaiton State Machine



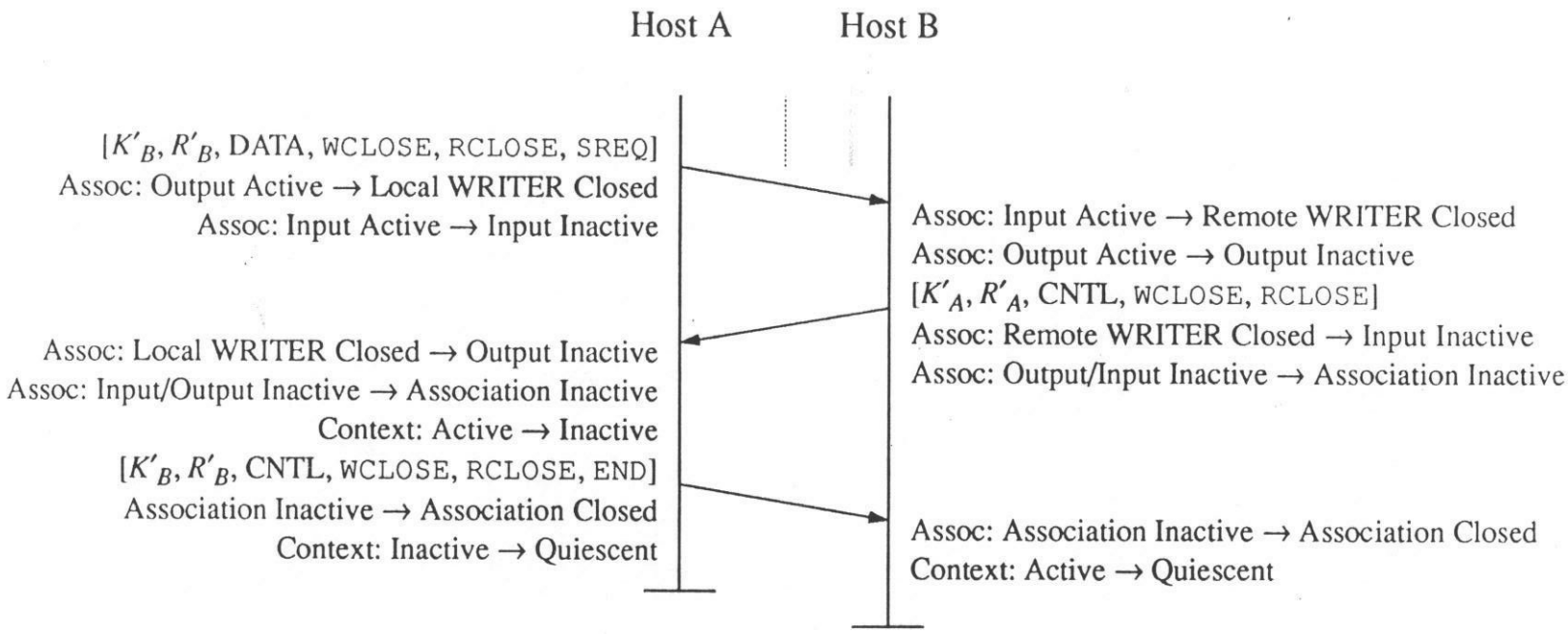
Independent Graceful Close



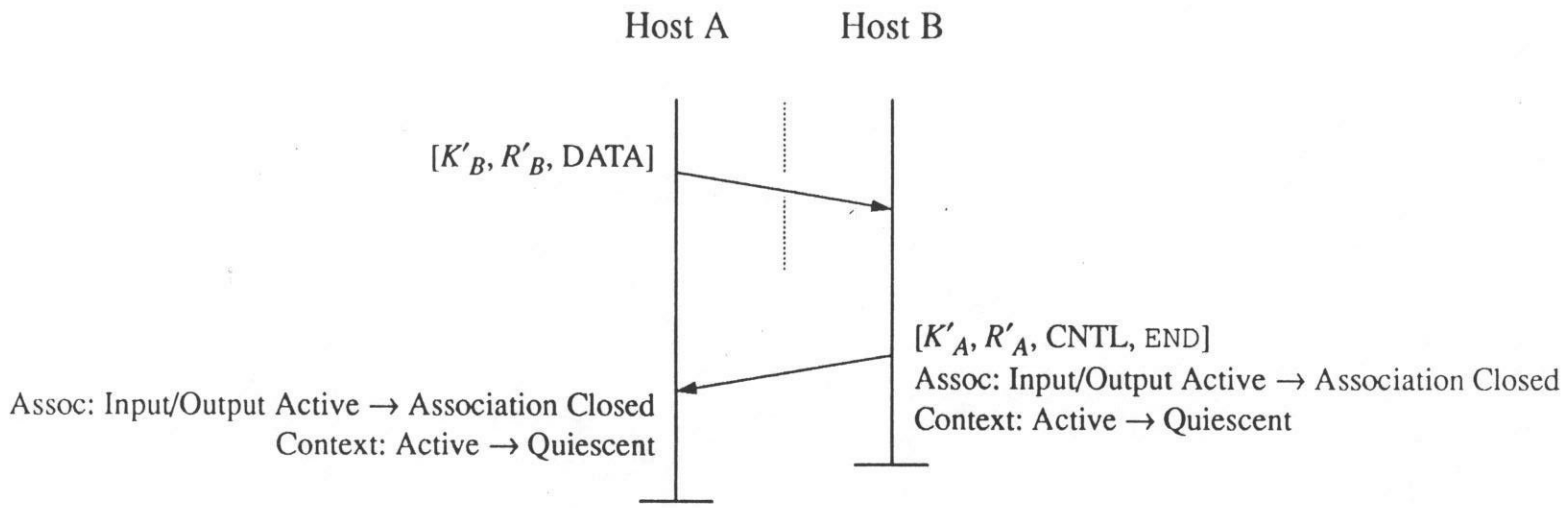
Abbreviated Graceful Close



Forced Close



Abort


Host A
Host B
[K'_B, R'_B, DATA]
[K'_A, R'_A, CNTL, END]
Assoc: Input/Output Active → Association Closed
Context: Active → Quiescent
Assoc: Input/Output Active → Association Closed
Context: Active → Quiescent

Protocol Procedures

Association Management Procedures
- Establishment
- Maintenance
- Termination

Path Management Procedures
- Establishment
- Maintenance
- Release

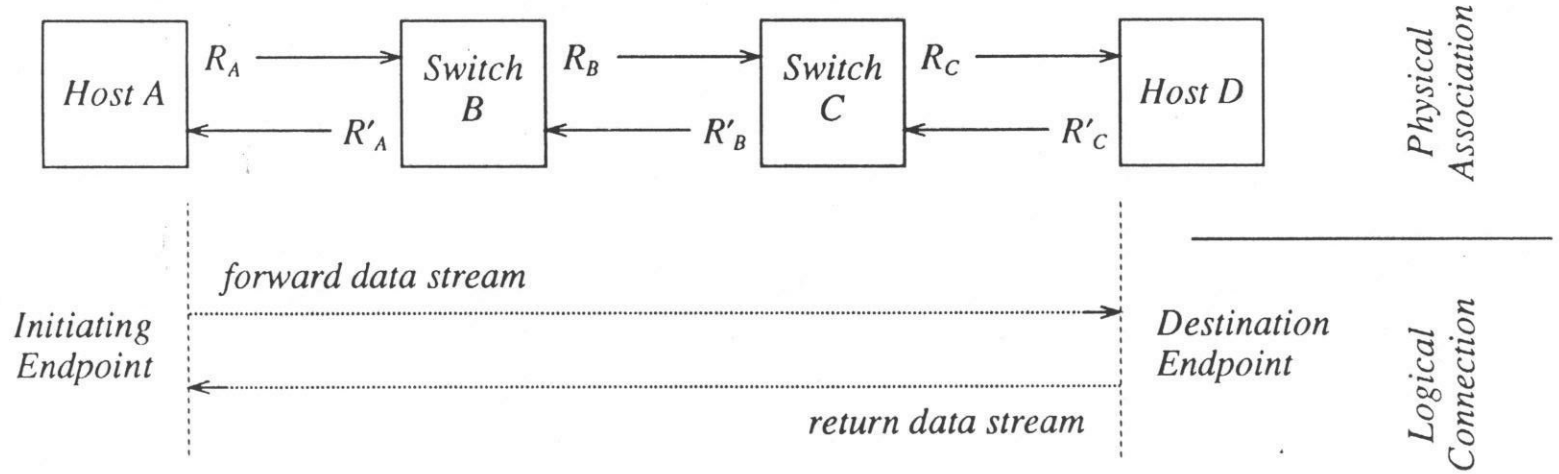
Data Transfer Procedures

Control Procedures
- Flow Control
- Rate Control
- Error Control

Path Management Procedures

Path Establishment

- Cut-through Routing: FIRST packet leaves a trail of references
- References: an ordered list of *route* values
- Recall: a *route* value is unique within host that creates it
- Example:



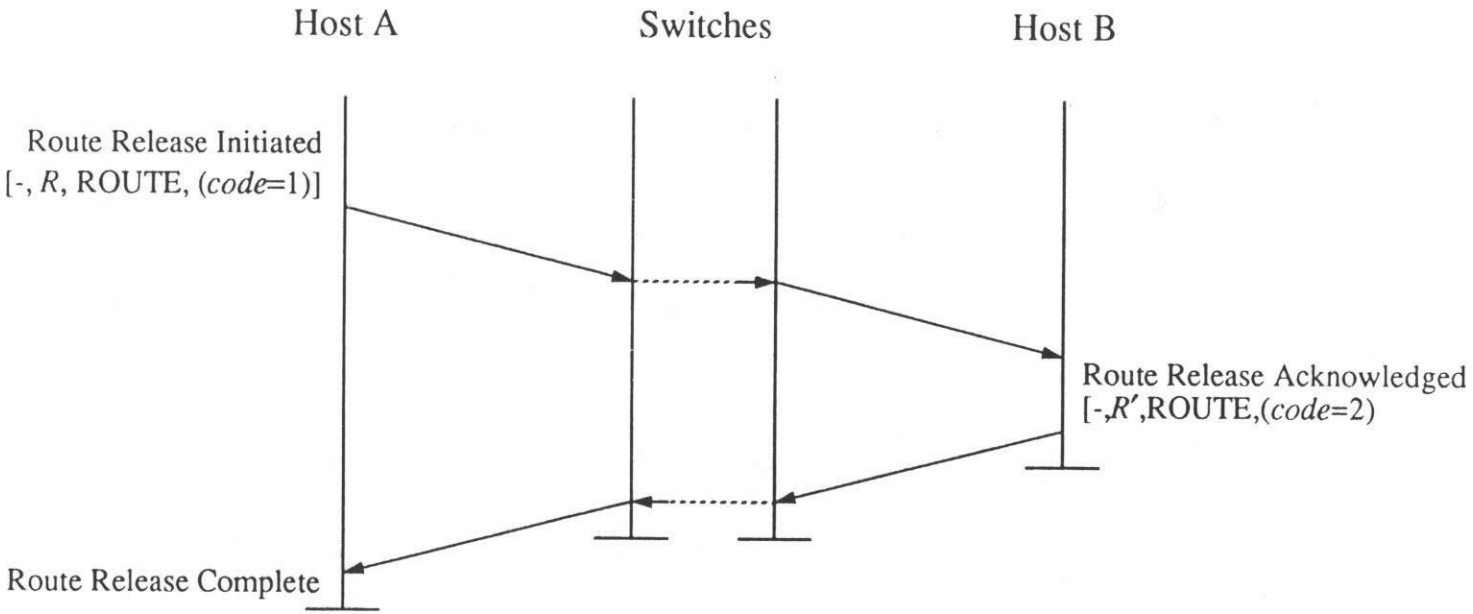
Path Management Procedures

Path Maintenance

- Route sharing
  - — two or more associations between this host and the same destination
  - — different *key* values, same *route* value
- Route migration
  - — when path becomes unusable or undesirable
  - — PATH packet rethreads a new path
- Route parameter adjustment
  - — RCNTL packets used to communicate *rate*, *burst* values
  - — this packet exchange is completely independent of association

Path Management Procedures

Path Release



- Release begun after long period without activity
- Any node on the path can initiate
- Two-way handshake

Protocol Procedures

Association Management Procedures
- Establishment
- Maintenance
- Termination

Path Management Procedures
- Establishment
- Maintenance
- Release

Data Transfer Procedures

Control Procedures
- Flow Control
- Rate Control
- Error Control

Data Transfer Procedures

Normal User Data Transfer

- User data is a parameter of an **output** command
- The data is placed in buffers in XTP implementation
- The data is segmented, placed into packets
- Each byte of the data is assigned a *sequence number*
- The data is reassembled at destination, placed in buffers
- The data is delivered to the user via an **input** command

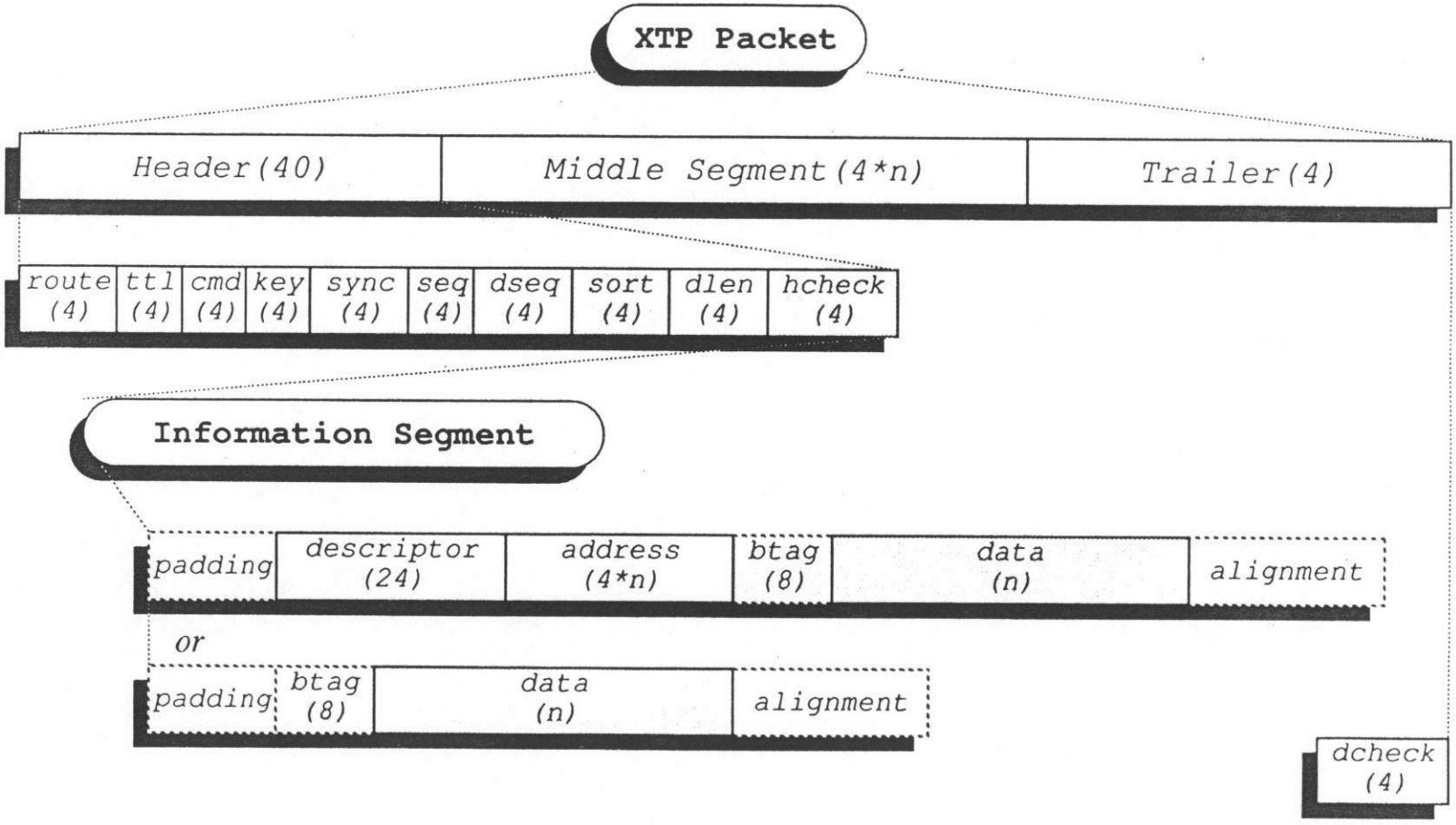
Out-of-Band Data Transfer

- Separate channel is provided for out-of-band data
- May be used as a higher-layer encapsulation mechanism

Data Ordering

- A ranking mechanism is provided
- Contexts are serviced from highest to lowest rank

Data Transfer Procedures—Packet Structure



Fields Used in Data Transfer Procedures

data field

- carries normal user data
- arbitrarily long (limited by packet size constraints)
- each byte of data is assigned a sequence number

btag field

- carries out-of-band data
- first 8 bytes of *data* field if `BTAG` bit is set

seq field

- value represents first sequence number assigned in this packet
- if FIRST packet it is first byte of addressing information
- if DATA packet it is first byte of *data* field

dlen field—count of all bytes in Middle Segment

sort field—32-bit priority field

dcheck field—checksum over Middle Segment

Protocol Procedures

Association Management Procedures
- Establishment
- Maintenance
- Termination

Path Management Procedures
- Establishment
- Maintenance
- Release

Data Transfer Procedures

Control Procedures
- Flow Control
- Rate Control
- Error Control

Flow Control Procedures

XTP provides a credit-based sliding window flow control algorithm

- volume from sender is limited by an *allocation* of sequence space
- lower edge of window moved via acknowledgements
- upper edge of window moved via new allocations

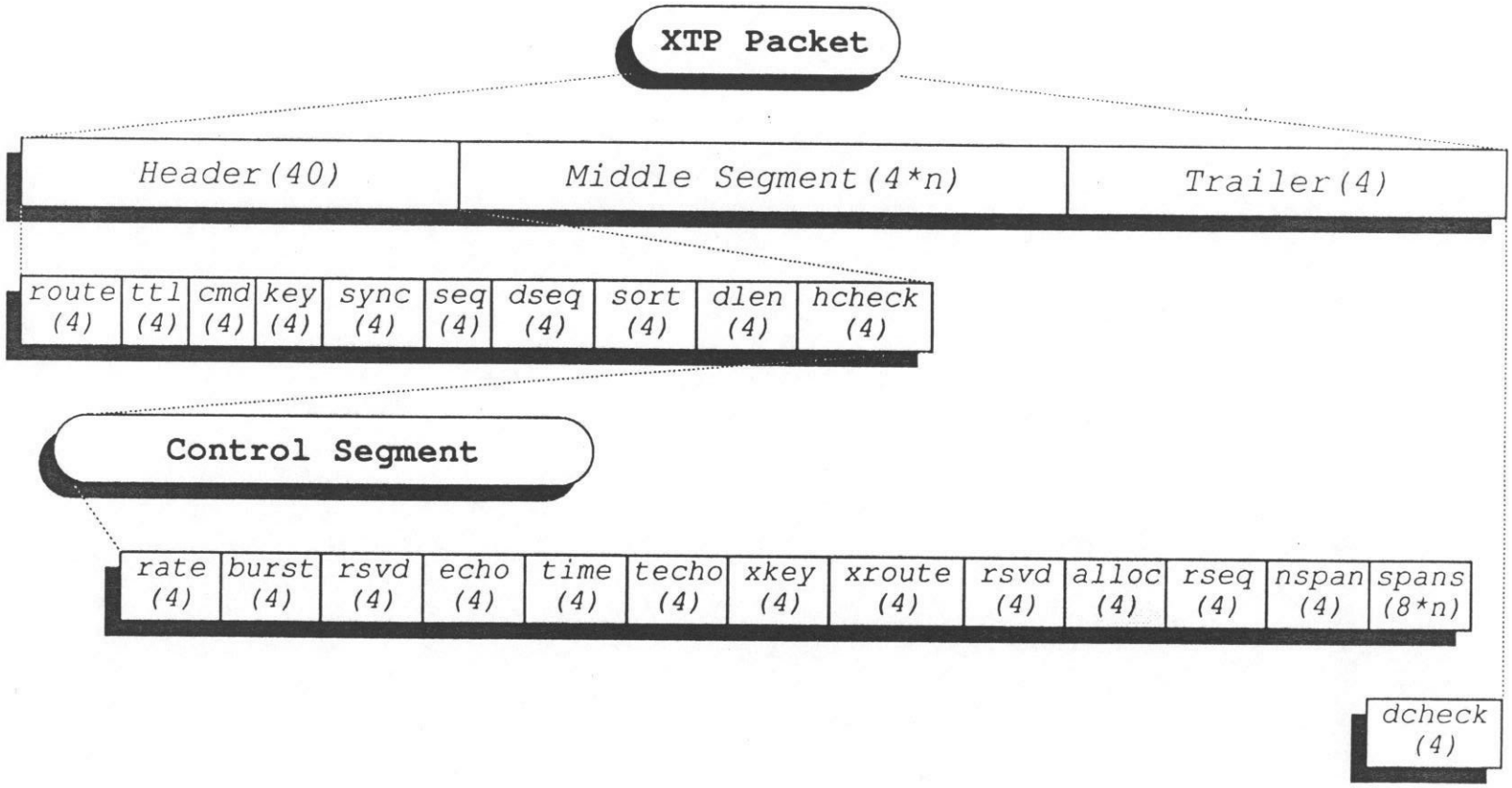
Reservation Mode

- enabled when `RES` bit is set
- forces receiver to use a conservative allocation policy

No-Flow Mode

- enabled when `NOFLOW` bit is set
- disables flow control restrictions
- receiver may reject this at association establishment

Fields Used in Flow Control Procedures



- *deq* field—last byte delivered to user—lower edge of window
- *alloc* field—a "do not exceed" value—upper edge of window

Rate Control Procedures

Attribute of the path, not the association

Regulates data producer so as not to overrun consumer

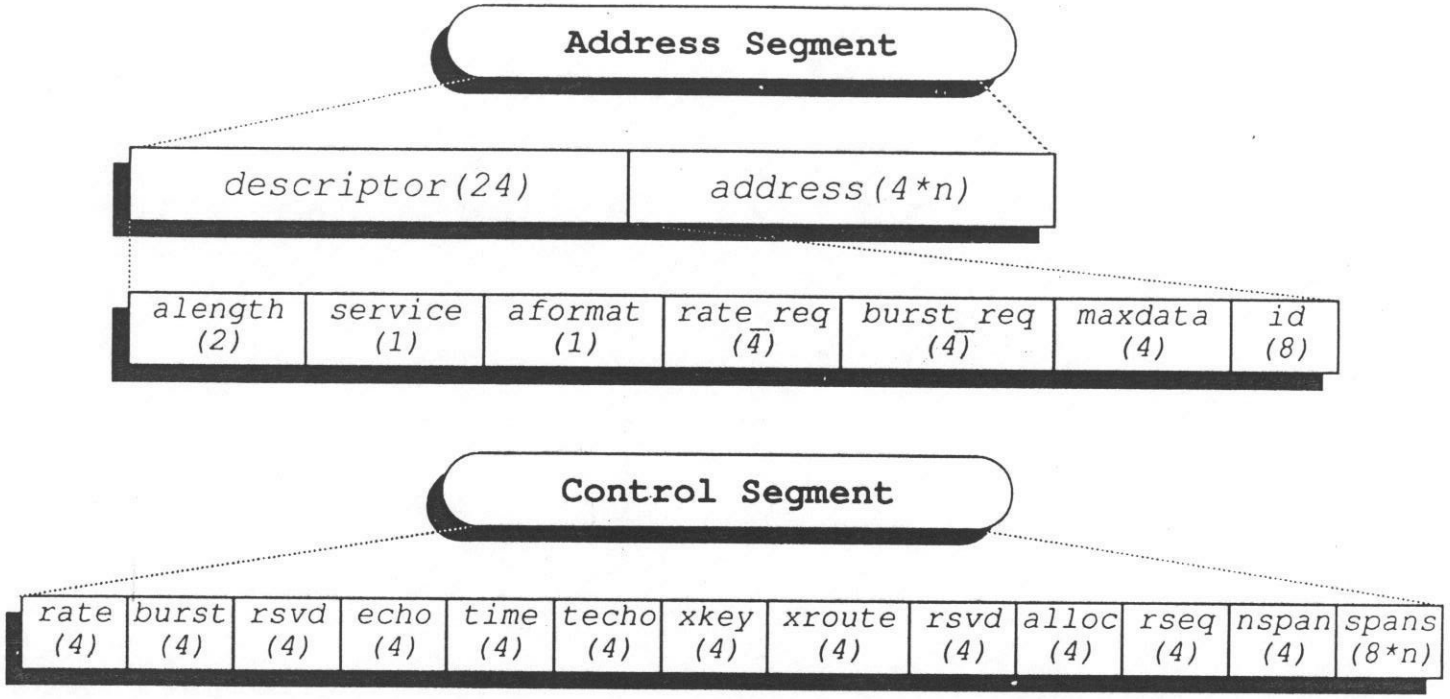
Parameters are a composite of all nodes on the path

- *rate*—number of bytes per unit time that can be consumed
- *burst*—number of bytes that can be handled in one group

Special values of *rate* and *burst*:

- *rate* = 0 → transmission halted
- *burst* = 0 → transmission unconstrained

Fields Used in Rate Control Procedures

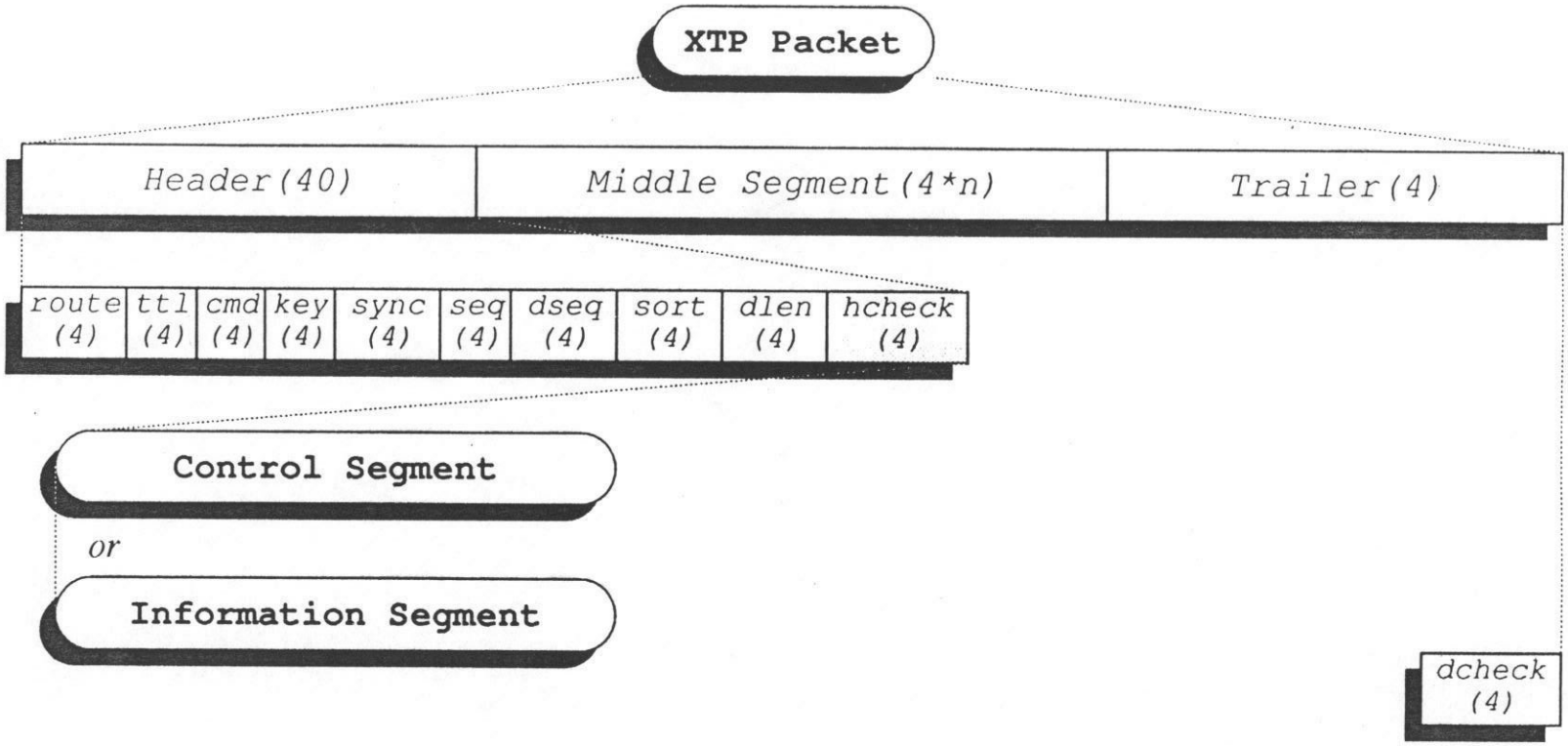


- *rate_req* and *burst_req* fields—suggestions to receiver (FIRST packet)
- *rate* and *burst* fields—actual parameters (CNTL and RCNTL packets)

Error Control Procedures

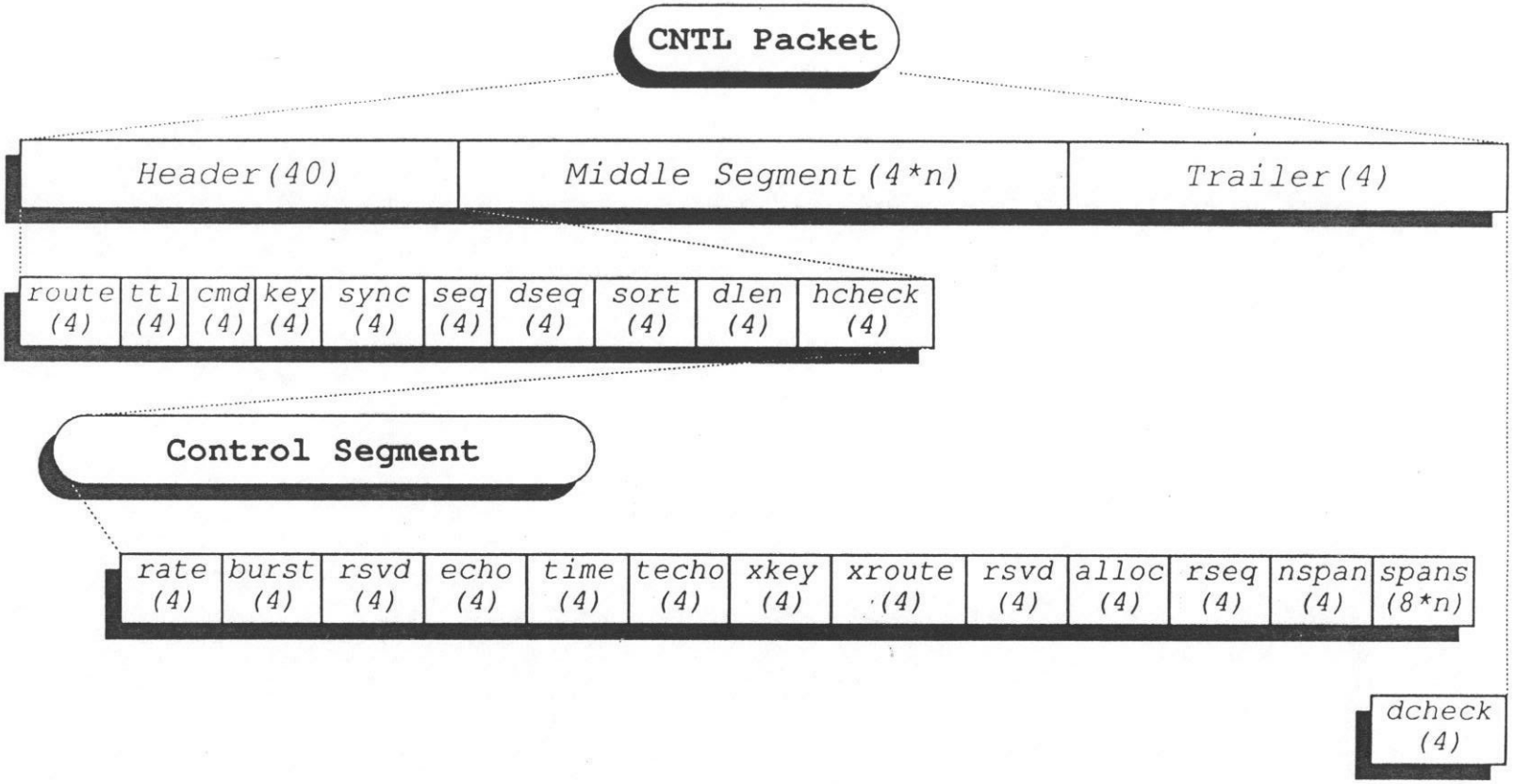
Checksum
Acknowledgements and Retransmissions
Synchronizing Handshake
Time-to-Live
Error Notification

Checksum



- *hcheck* field —checksum over all header fields (except *hcheck*)
  —can not be disabled
- *dcheck* field —checksum over all Middle Segment fields
  —may be disabled with NOCHECK

Acknowledgements and Retransmissions



- *rseq* field—first byte not contiguously received
- *nspan* field—number of spans present in *spans* field
- *spans* field—sequence number pair that represents contiguously received data

Acknowledgements and Retransmissions

Go-back-*n* policy

- Retransmit from last contiguously received byte of data
- Simple and easy but not necessarily efficient
- The value in *rseq* field gives starting point

Selective retransmission

- Retransmit only the gaps of missing data
- May be cheaper to concentrate only on what is missing
- Gaps are derived from report of received data in *spans* list

Fast negative acknowledgement

- Enabled when `FASTNAK` bit is set by transmitter
- Causes receiver to generate a CNTL packet as soon as out of order data is received

No-error mode

- Enabled when `NOERR` bit is set by transmitter
- Causes receiver to always set *rseq* field to highest sequence number seen and ignore gaps

Synchronizing Handshake

Procedure to ensure that endpoints are synchronized

Returning CNTL packet's *sync* value must match *sync* value sent

Used to ensure accurate roundtrip time estimates

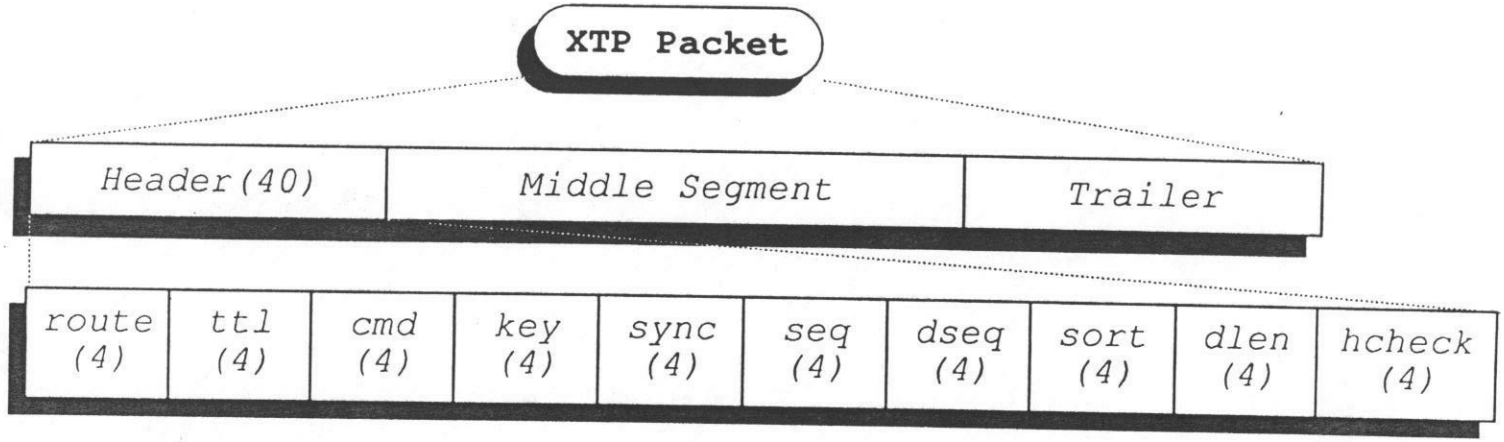
Used when WTIMER expires

- No response to an `SREQ`
- Either other endpoint is dead or WTIMER is too small
- Backoff algorithm retries synchronizing handshake for a larger WTIMER value
- Repeats backoff for a bounded amount of time

Used when CTIMER expires

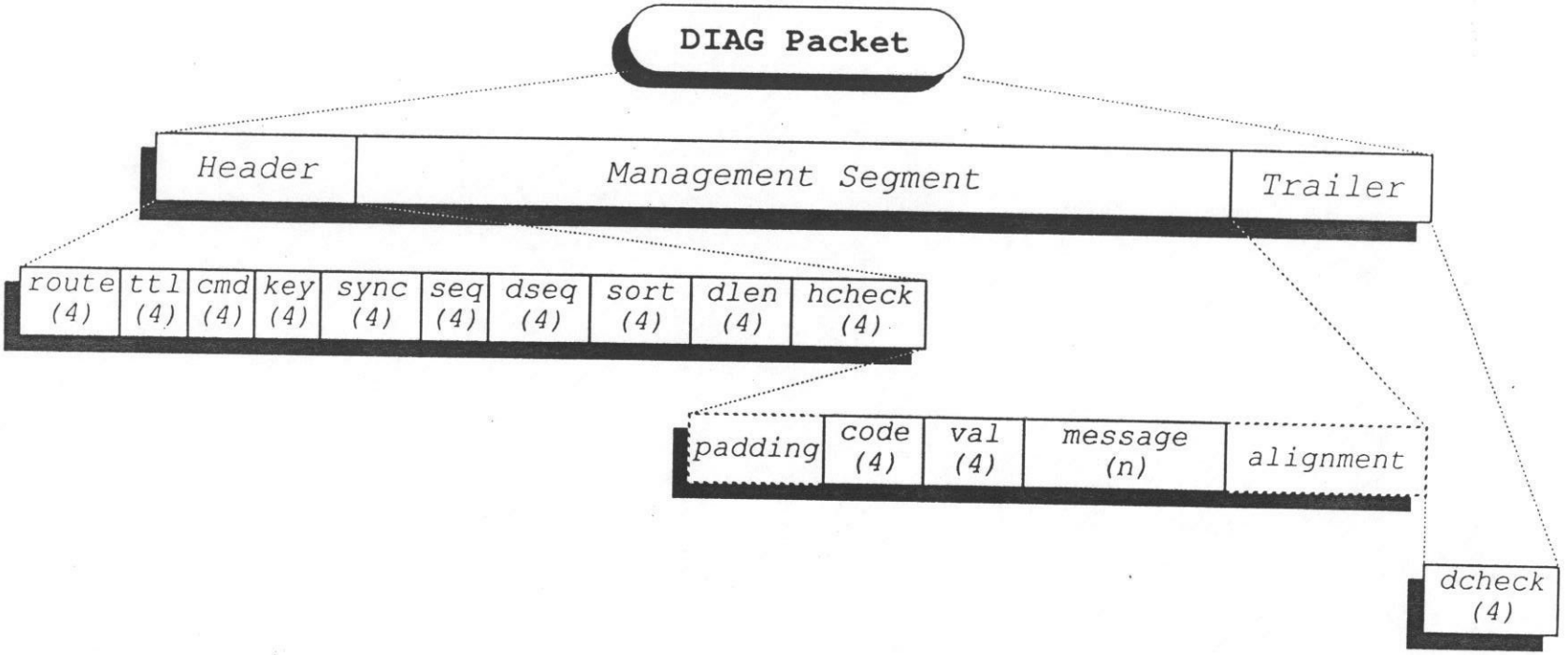
- Attempt to revive a dormant association
- Will "ping" other endpoint for a bounded amount of time before aborting
- Repeats backoff for a bounded amount of time

Time-to-Live



- Restricts lifetime of a packet
- Value of *ttl* field in units of 100 nanoseconds
- Putting a value of 0 in *ttl* turns off this mechanism

Error Notification



- Provides timely notification of errors
- Code field specifies which kind of error
- *val* field further modifies or holds *code*-specific value
- *message* field holds indication of error for log file

Summary

Protocol had to be powerful and lightweight

- procedures are built from a small state vector
- procedures are regular and orthogonal
- protocol is highly configurable
- economies and optimizations are exploited

Result: One streamlined protocol can support many different services and paradigms

Next!

XTP Addressing
XTP Multicast

Roadmap to Addressing and Encapsulation in XTP

Addressing Information

Parametric Addressing

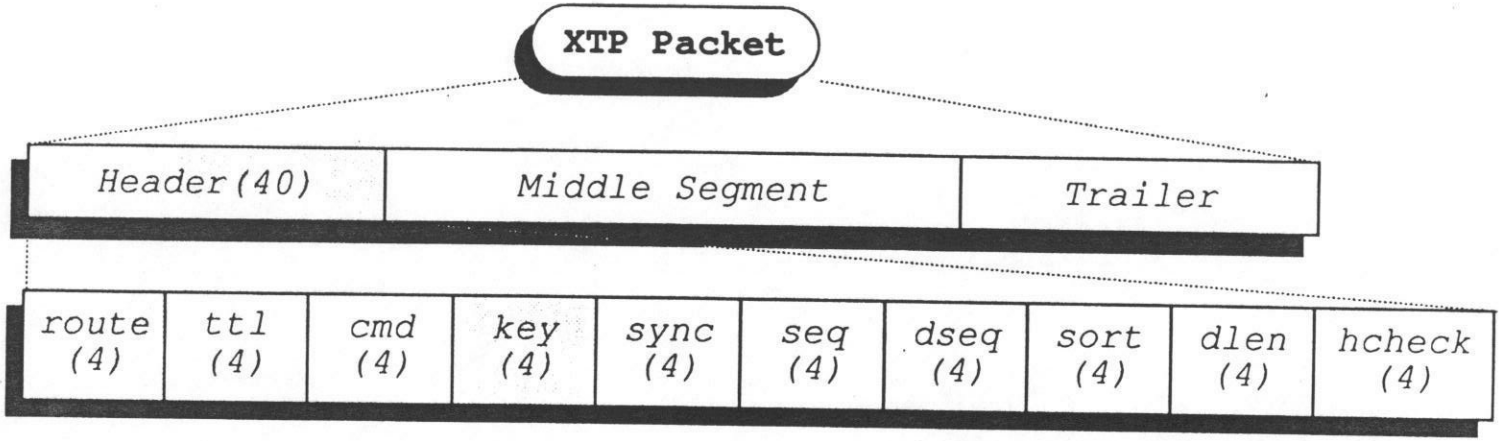
The *Address* Segment in FIRST (and PATH) Packets

Examples of Addressing Formats

Encapsulation

Examples of Encapsulating XTP Packets

Fields for Address Translation in Established Associations



- Packets in an established XTP association use lookup algorithms
- At end-systems the *full-context lookup* using the triple <key, route, MAC source-id> can be replaced by an *abbreviated lookup* (using only 32-bit *return key* value) in common scenarios
- Key-based lookup reduces the overhead of mapping packets to their destination contexts to a single-valued table lookup
- In switching nodes, lookups take place on (*return*) *route* values in a similar fashion to key-based lookup at end-systems

Addressing in XTP FIRST packets

FIRST packets carry full transfer layer addressing information that serves two purposes

- it enables the FIRST packet to establish a path through the network to the destination end-system
- it enables the FIRST packet to activate the remote endpoint at the destination end-system

At an intermediate switching node, this addressing information steers the packet to the destination network and sets up routing state for subsequent packets

At the destination end-system, this addressing information is used to match the FIRST packet to a context in the LISTEN state that has opened an address filter on the desired address

XTP's Parametric Addressing

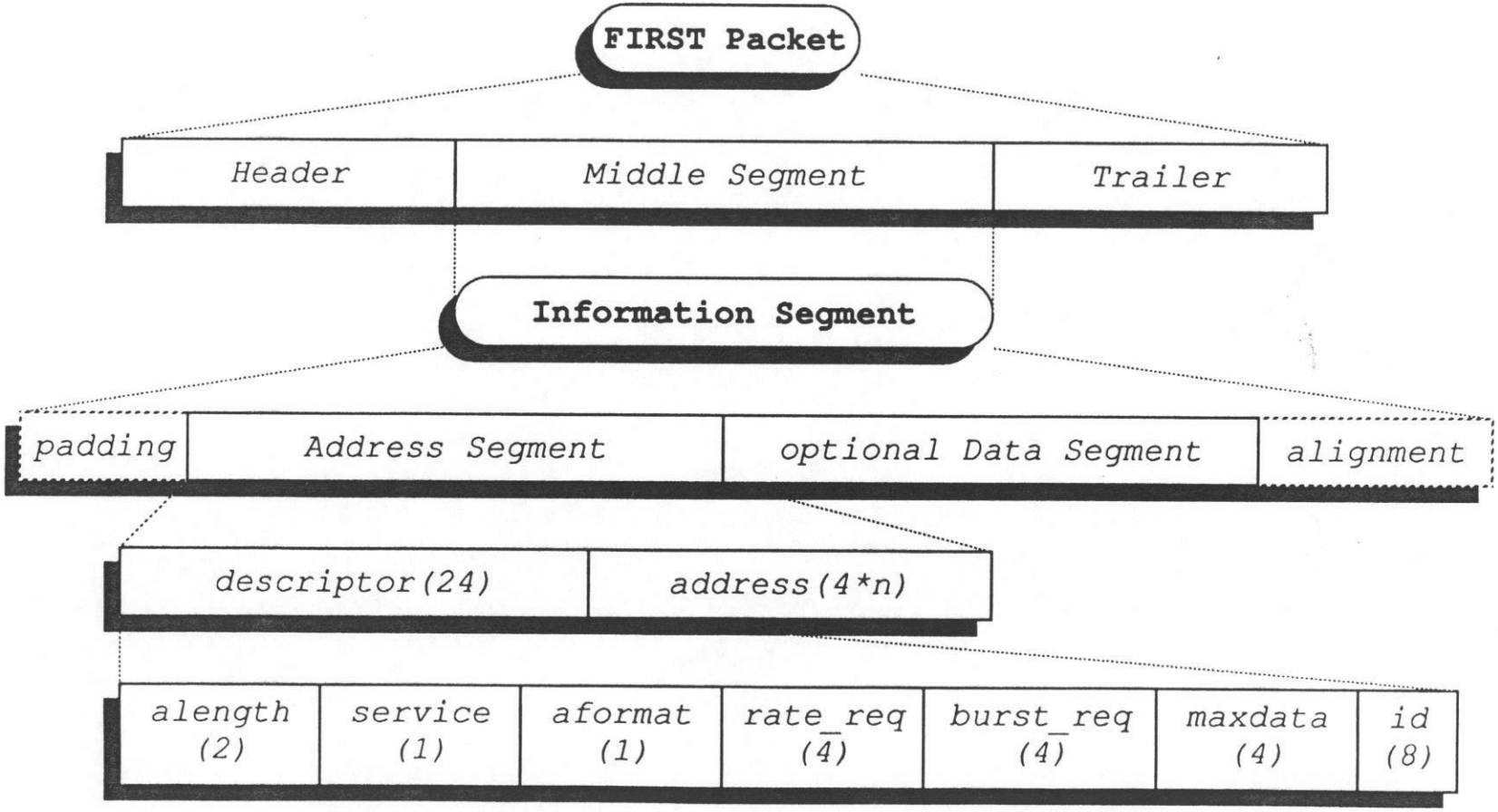
Instead of defining a new addressing scheme, XTP adopted a **parametric addressing** mechanism

The FIRST packet (and PATH packet) has an Address Segment with two components —

- a **descriptor** identifying the addressing domain being used
- an **address** in the format specified by the descriptor

This feature allows the use of existing addressing schemes along with their associated naming and routing functions

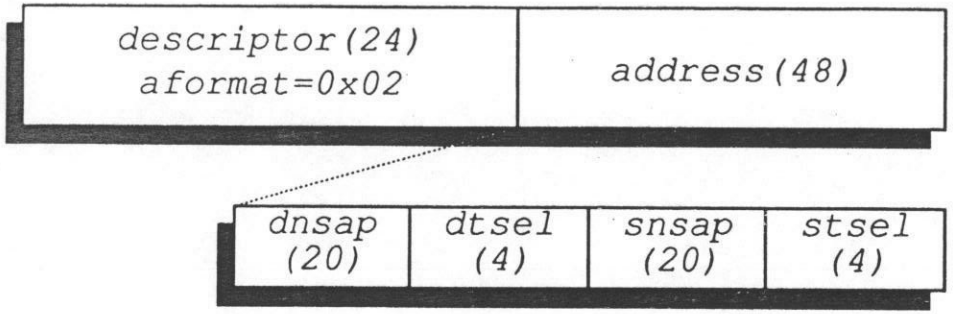
Address Segment in FIRST Packets



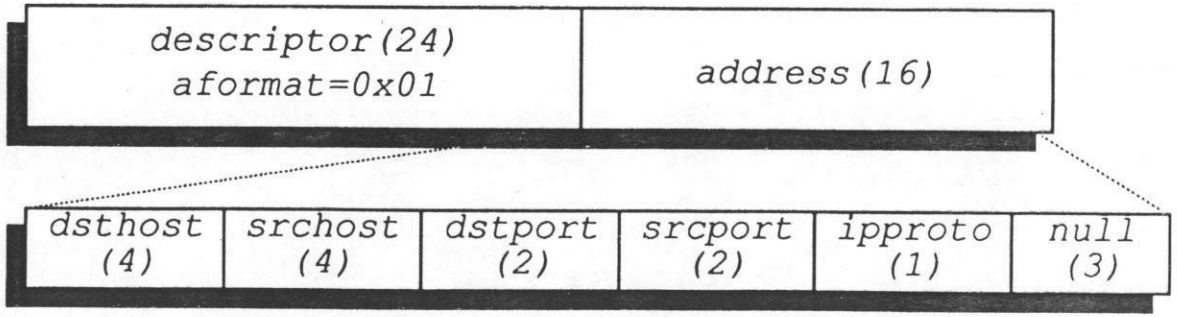
Addressing Schemes Defined for XTP Address Segments

aformat field value		Address Syntax
Decimal	Hex	
0	0x00	*No address*
1	0x01	*Internet Protocol addressing*
2	0x02	*ISO 8348 addressing*
3	0x03	*Xerox Network System addressing*
4	0x04	*IEEE 802-style source route address*
5	0x05	*MODSIM*
6	0x06	*MicroSoft NetBIOS*
7	0x07	*IP-style loose source route address*
8	0x08	*IP-style strict source route address*
9	0x09	*XTP direct addressing*
10	0x0A	*XTP experimental address type*
11	0x0B	*USAF embedded system address*

Two Examples of Addressing Formats



ISO 8348 Address Format



Internet Protocol Address Format

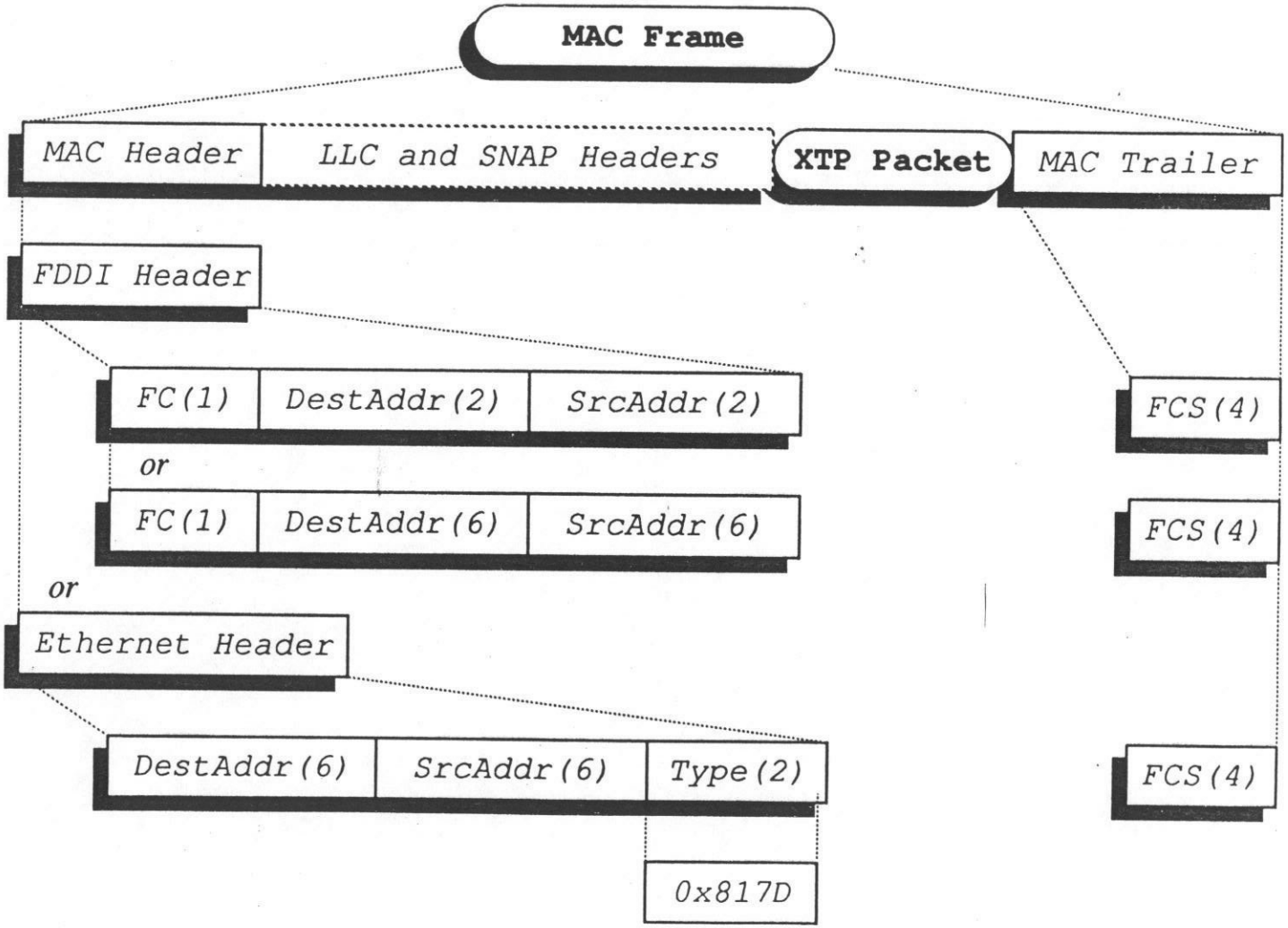
Encapsulation

XTP packets are encapsulated into the frames of an underlying protocol

The XTP Definition provides guidelines for encapsulation into Ethernet, FDDI, IEEE 802.5 Token Ring, and the use of Logical Link Control (LLC) headers

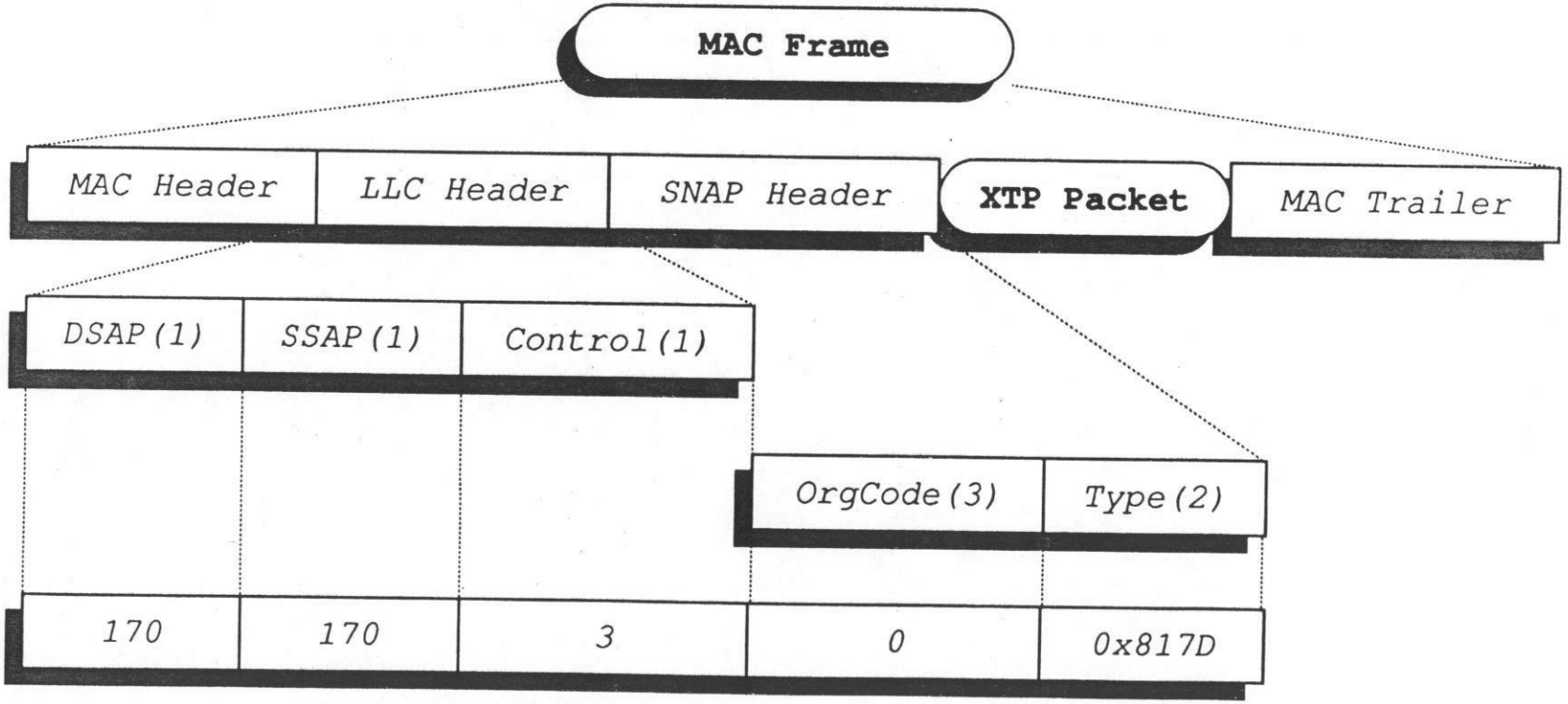
In addition, the XTP Definition specifies how an XTP packet can be encapsulated as the data portion of an Internet Protocol datagram

Encapsulation in Ethernet and in FDDI Frames

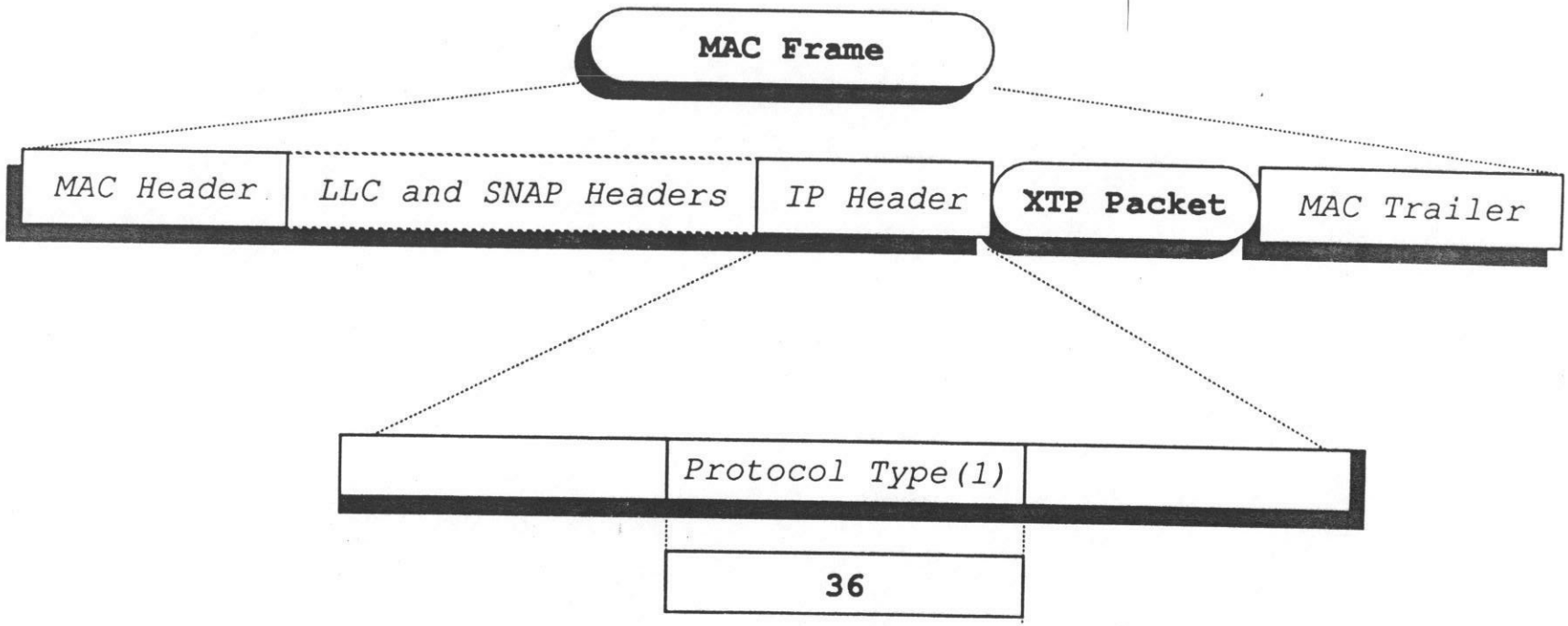


Encapsulation using IEEE 802.2 LLC with a SNAP Header

Logical Link Control (LLC) and Sub-Network Access Protocol (SNAP) can provide multiplexing service at the Data Link Layer, as per RFC 1103 for XTP encapsulation in FDDI frames



Tunneling through the Internet Protocol



XTP has been assigned the protocol type number decimal 36 for inclusion in the Protocol Type field of the IP header

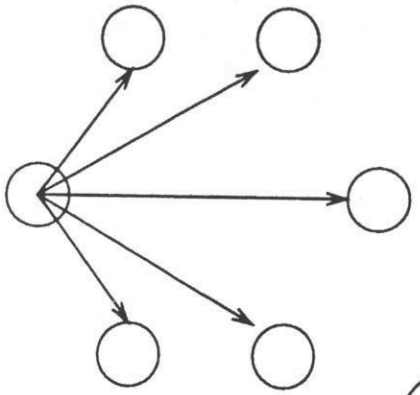
This is the only XTP-specific information in the IP Header

Roadmap to XTP Multicast Discussion

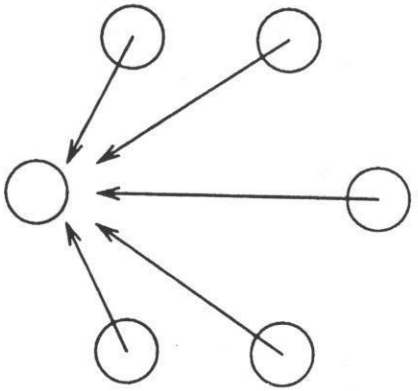
- Introduction to Group Communication
- XTP Multicast Design
  - Multicast Addressing
  - Multicast Association Establishment
  - Multicast In-progress Join Procedure
- XTP Multicast Heuristics
  - Slotting and Damping
  - Bucket Algorithm
  - Cloning
- Summary Remarks

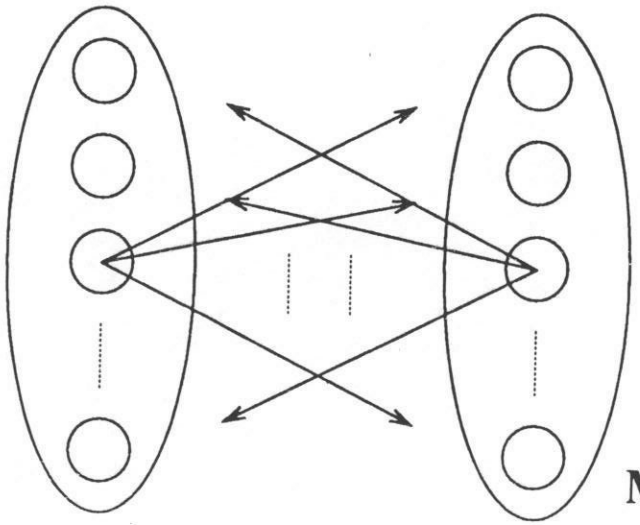
Group Communication Topologies

Point-to-Multipoint
(Multicast)



Multipoint-to-Point
(Concentration)





Multipoint-to-Multipoint

Multicast—Point-to-Multipoint Data Transfers

Medium access control (MAC) layer

- group addressing in a physical broadcast media
- increasing hardware support for group address filters

Network layer

- network layer endpoints across gateways

Transport layer

- transport layer endpoints and reliable multicast

Operating system level

- notion of distributed process groups not new
- convenient abstraction

Advantages of Multicast

Efficiency and speed relative to multiple serial transmissions

- Avoids overhead associated with pure broadcast
- Conserves resources at end-systems
- Eliminates traffic on network

Robustness of dynamic binding between the physical members of a group and the logical group address

- Locating a network service

Synchrony of multidestination delivery

Simplification of end-system design and application software

Multicast Applications

Resource location
Redundant servers
Replicated Remote Procedure Calls
Distributed agreement algorithms
Distributed simulations
Groupware
Synchronization
Industrial automation and process control
Real-time command and control
Teleconferencing
Multimedia traffic

XTP Multicast Design

Simple but powerful 1-to-N data delivery service

- Multicast with control algorithms
- Implementation hook for concentration

Integration of unicast/multicast in protocol

Scaleability—no association control procedures depend explicitly on the number of receivers in the multicast association

Room to take advantage of just-emerging work on multicast routing

Network Layer Multicast

Multicast through routers is not a well understood problem

Multicast considerations have always been a part of the XTP design, and protocol mechanisms exist within XTP to mesh routing procedures with existing multicast functions

Using Direct Addressing with some associated logic in the routers, XTP multicast can be defined across switches in closed networks

XTP can take advantage of the work just now coming together on Network Layer multicast protocols

Emerging Network Layer Multicast Protocols

IP Multicast

- connectionless, Host-to-Gateway defined (RFC 1112)
- Gateway-to-Gateway recently put forward (S. Deering)

ST-II

- connection-oriented with resource reservation for real-time services

X.6

- CCITT standards effort (features not yet firm)
- X.25 framework, connection-oriented

XTP Multicast—Addressing

Lookup Algorithms

- XTP key-based lookup algorithms work as with unicast with the exception of no key exchange procedure

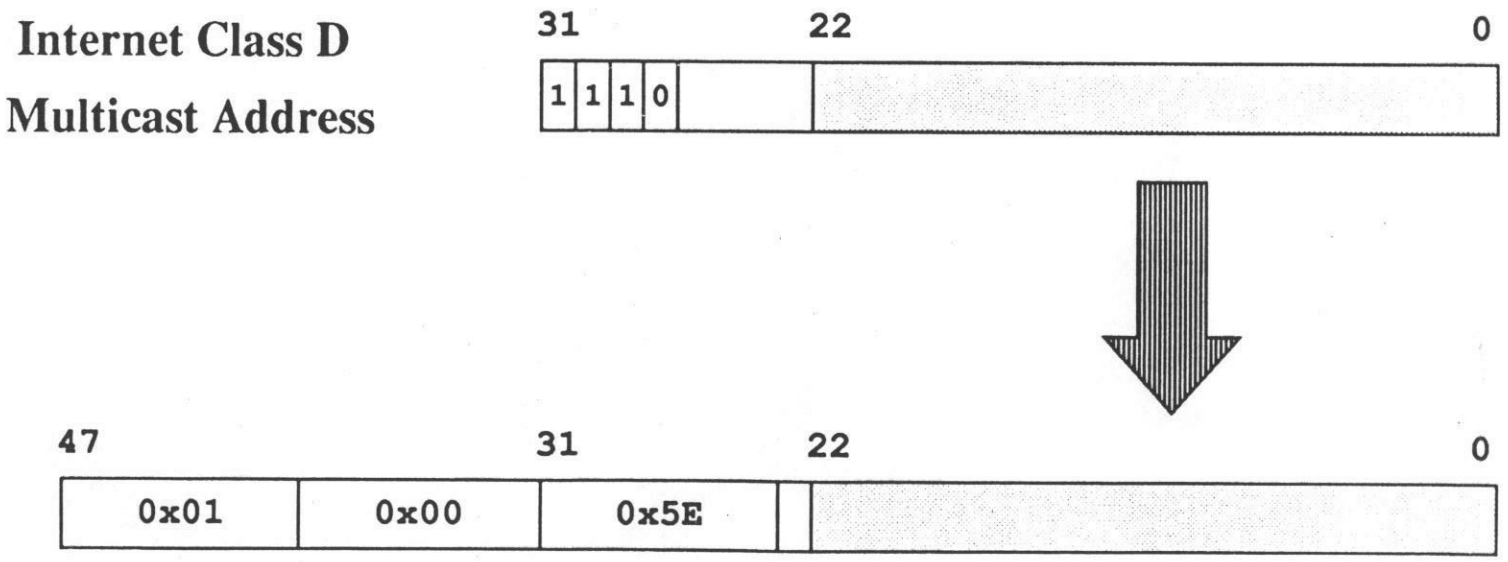
Addresses in Address Segments

- XTP Definition explicitly specifies either IP Group Addresses or Direct Addressing formats
- Direct Addressing format allows specialized schemes
- Parametric addressing allows for future group addressing formats

Generating MAC Group Addresses using RFC 1112

Mapping takes an Internet Class D Address into an Ethernet Group Address

Similar mapping for other 48-bit IEEE addresses



XTP Multicast—Association Establishment

Multicast Receivers open address filters on group address

Multicast Sender issues FIRST packet on group address with MULTI bit set

FIRST packet reception at each destination host invokes the XTP lookup algorithms to find the waiting Receiver context(s) on that host

Multicast association (the Sender-to-Receivers simplex data flow) is thus established

NOTES

All packets in multicast association, including CNTL packets issued by the Receivers, are sent to the group address

Multicast Sender can set only SREQ, not DREQ

Protocol Options in the cmd field of the XTP Header

Protocol Option	Description of Protocol Option
NOCHECK	Turns off the checksum over the Middle Segment
NOERR	Turns off error control—no-error mode
MULTI	Indicates that this is a multicast association
RES	Indicates that receiver is in reservation mode
SORT	Indicates that this packet is to be ordered by sort value
NOFLOW	Turns off flow control
FASTNAK	Indicates fast negative acknowledgement policy
SREQ	Indicates status requested immediately
DREQ	Indicates status requested after previous data delivered
RCLOSE	Signals the close of the READER process
WCLOSE	Signals the close of the WRITER process
EOM	Indicates end of message
END	Indicates end of association
BTAG	Indicates the presence of a btag field in the data field

XTP Multicast—In-Progress Join Procedure

XTP Multicast context wishing to join an existing active multicast association can issue a PATH packet on the group address with `key` == 0

All the Multicast Receivers receive and ignore this packet

When the Multicast Sender receives the PATH packet, it responds with a PATH packet whose Header contains the `key` and other state of the multicast association

When the joining context receives this PATH packet, it begins receiving data at the current sequence number

NOTES

Does not disturb the on-going multicast data transfer

Sender controls boundaries at which new members join

Roadmap to XTP Multicast Discussion

- Introduction to Group Communication
- XTP Multicast Design
  - Multicast Addressing
  - Multicast Association Establishment
  - Multicast In-progress Join Procedure
- **XTP Multicast Heuristics**
  - Slotting and Damping
  - Bucket Algorithm
  - Cloning
- Summary Remarks

XTP Multicast Heuristics

Heuristics are **not** part of the protocol

Heuristics **are** suggested algorithms for using protocol features to improve efficiency and effectiveness of multicast facility

Heuristics by their nature are implementation-specific

Four XTP Multicast Heuristics

- Slotting
- Damping
- Bucket Algorithm
- Cloning

Slotting and Damping—CNTL Packet Processing at Receivers

XTP Multicast Receivers issue CNTL packets on group address

Receivers can ignore CNTL packets from other Multicast Receivers (easily detected by highest bit in `key` field)

Alternatively, Multicast Receivers can process CNTL packets

Slotting and damping heuristics suggest a way to reduce CNTL packet traffic by having each Receiver examine other Receivers' CNTL packets

Basic Algorithm for Slotting and Damping

Multicast Receiver receives a packet from the Multicast Sender with the SREQ bit set.....

1. Construct a CNTL packet
2. Choose an delay interval of ***N*** slot-times /* **slotting** */
3. Examine any in-coming Receiver-generated CNTL packets
4. If any in-coming Receiver-generated CNTL packet renders the local CNTL packet *obsolete*, then dismiss the local CNTL packet /* **damping** */
5. After the ***N*** slot-times delay, if the local CNTL packet has not been dismissed, send it

Slotting and Damping

Slotting and damping are independent heuristics, but experimental evidence suggest that they are most effective when used together

Slotting increases the likelihood of packets being damped

These mechanisms address the problems of

- excessive CNTL packet traffic
- network access contention due to Receivers issuing their CNTL packets at the same time (*implosion*)

Both problems increase as the number of Receivers increases

XTP Multicast Sender—CNTL Packet Processing

XTP multicast operates in two Error Control Modes

1> Without active error control (no-error mode multicast)

- CNTL packets issued infrequently

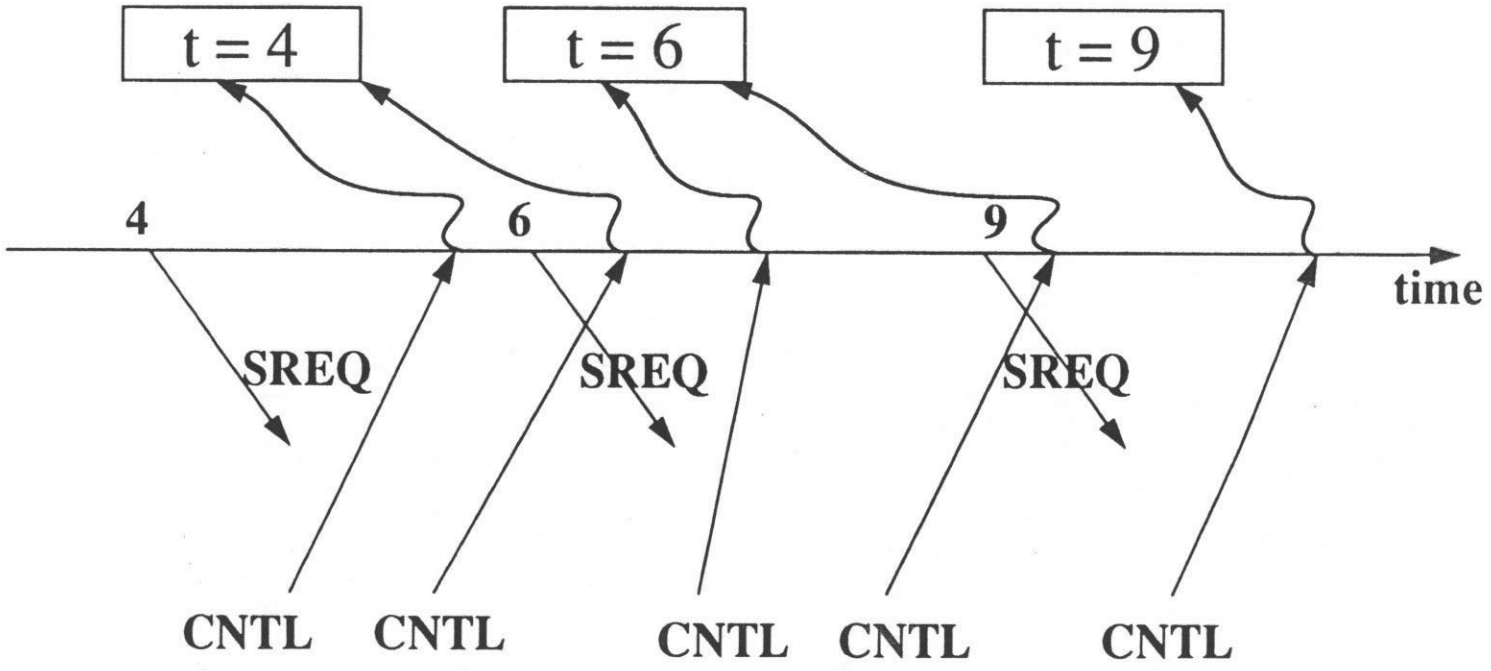
2> Active error control (semi-reliable multicast)

- CNTL packets needed more frequently in order to release transmit buffers, report packet loss, etc.

Important that Multicast Sender synthesize the stream of (possibly conflicting) CNTL packets returning from the Receiver set such that control algorithms can be effective

XTP Multicast —Bucket Algorithm Heuristic

Buckets at the Multicast Sender



Multicast Receivers

(NO SLIDE 32)

XTP Multicast —Bucket Algorithm Components

1. Which bucket holds incoming CNTL packet?
   Mechanism to match requests for status (SREQs) with status reports (CNTL packets from Multicast Receivers)

   SYNC/ECHO handshaking

2. How to incorporate CNTL packet into its bucket?
   Algorithm to synthesize the information contained in multiple CNTL packets

   Minimum of received values for alloc, rseq, dseq, techo

   Retransmission is go-back-n

   NOTE— An XTP implementation does not have to use the Bucket Algorithm and can, therefore, define a selective acknowledgment and retransmission multicast

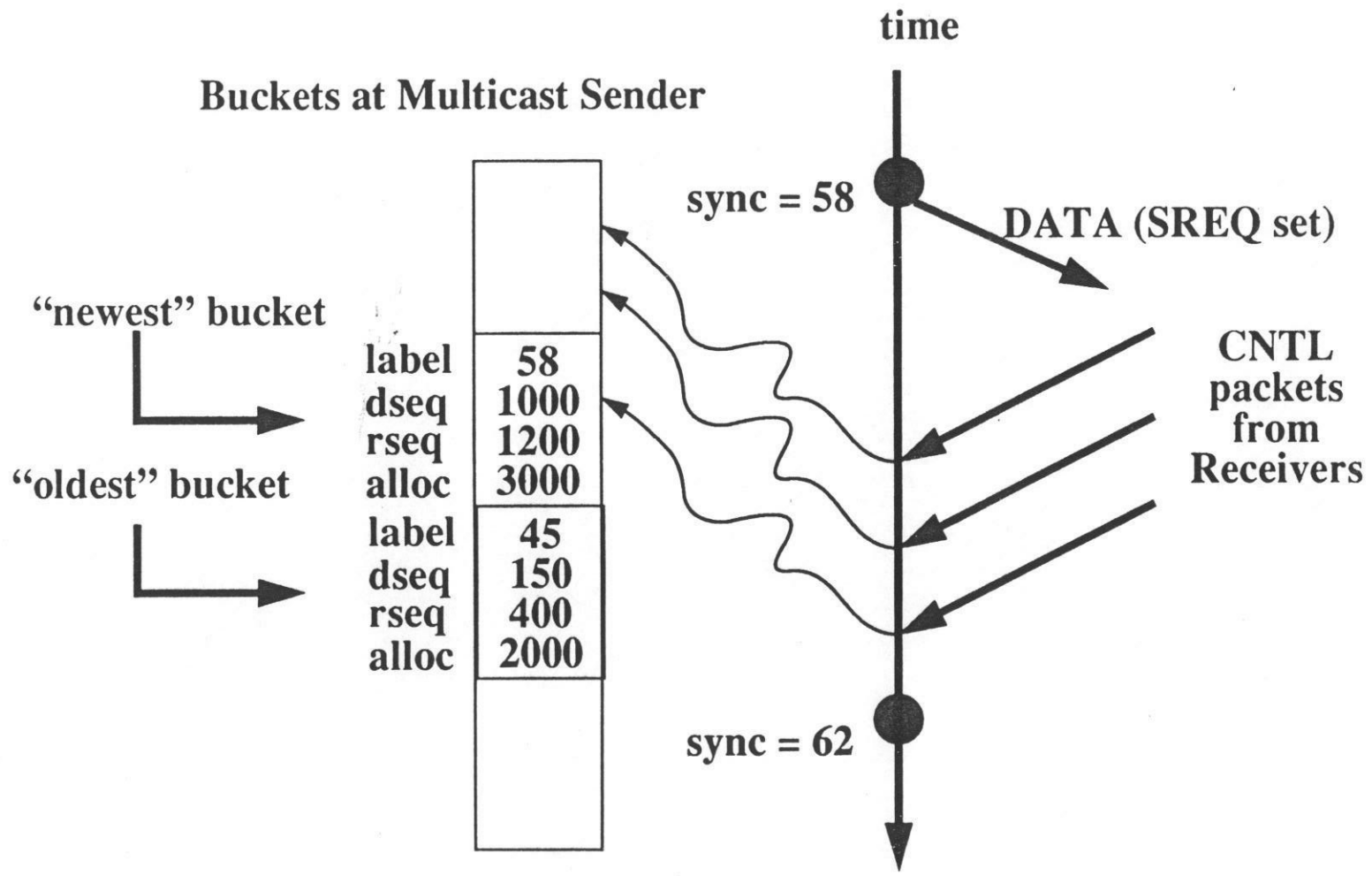
XTP Multicast —Bucket Algorithm Components (continued)

3. When to use accumulated CNTL information in a bucket?
   Technique for estimating delay between requesting status information from the Multicast Receivers and updating the Sender's state based on CNTL packets received from the Receivers

   Algorithm regulates delay by cycling through a set of buckets (data structures at the Multicast Sender holding state information derived from incoming CNTL packets)

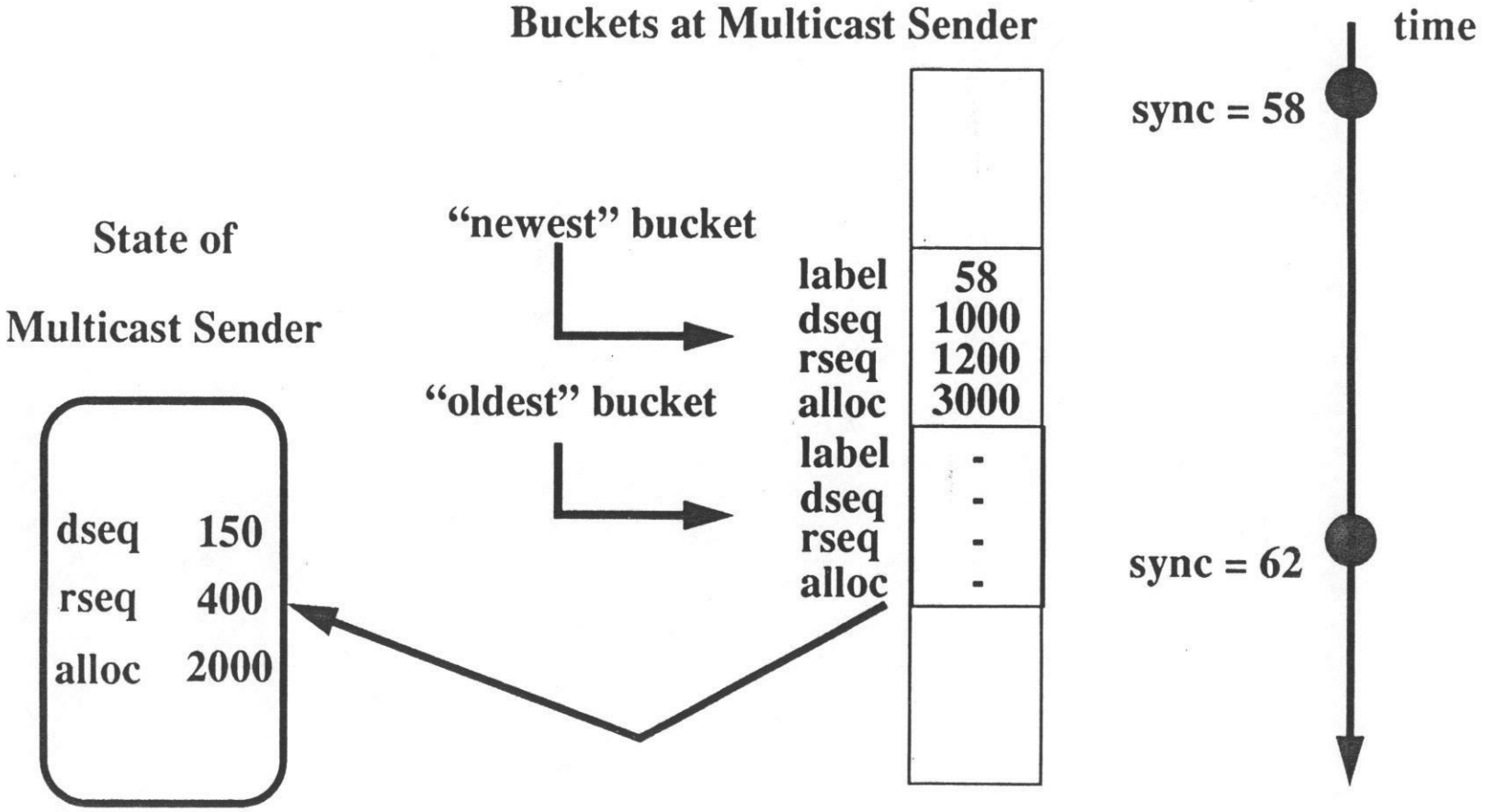
   Once all buckets are full, creating a new bucket means acting on the values in the oldest bucket; this "just-emptied" bucket is then initialized as the newest bucket

Bucket Algorithm—at time = 62 - Δ



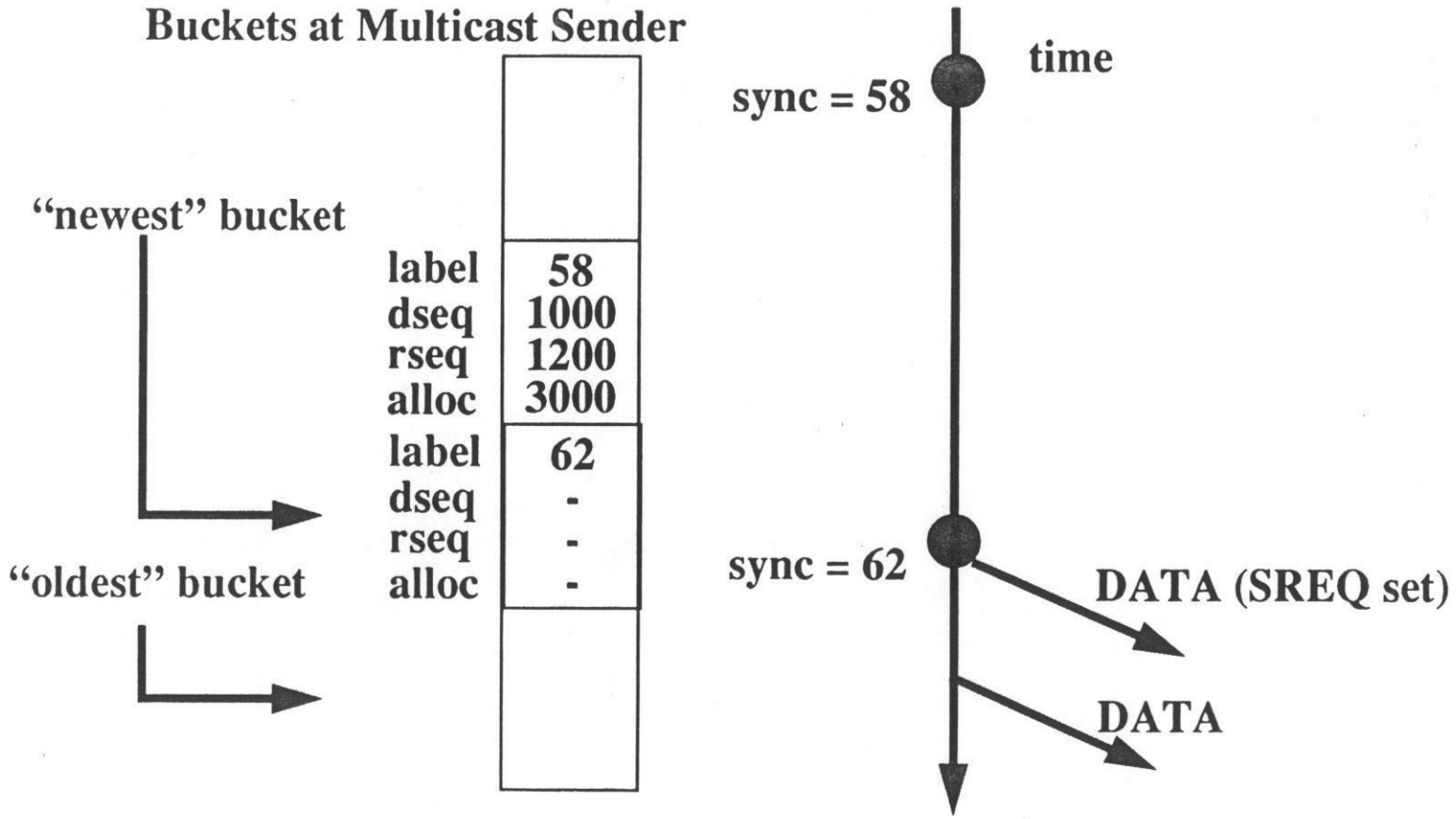
Bucket Algorithm—at time = 62 (Step 1)

Step 1— Emptying the oldest bucket (update Sender's state)



Bucket Algorithm—at time = 62 (Step 2)

Step 2—Initialize new bucket and send packet with SREQ



Summary of Bucket Algorithm

Reliability based on active receivers that do not lag too far behind (semi-reliable)

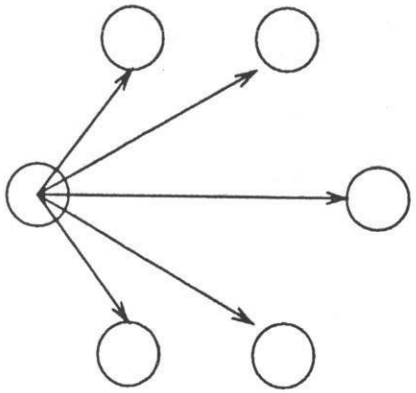
Explicit trade-offs between buffer space requirements and error tolerance can be made if the XTP user can choose the number of buckets to be used and the switch-time intervals

- More buckets and longer switch-times imply less likelihood of dropping receivers, but also more time between requesting the Receivers' State and updating the Multicast Sender's state based on the perceived Receivers' State

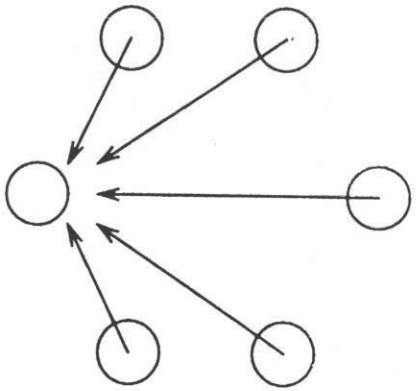
Any solution to control algorithms for streaming multicast in XTP must address the issues solved by the Bucket Algorithm

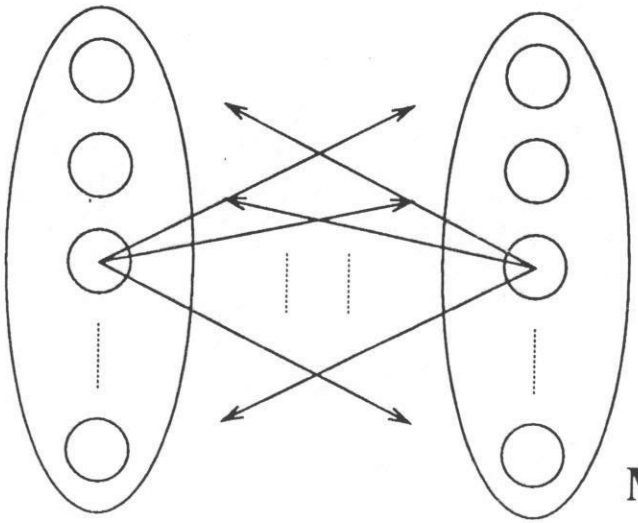
Group Communication Topologies

Multicast



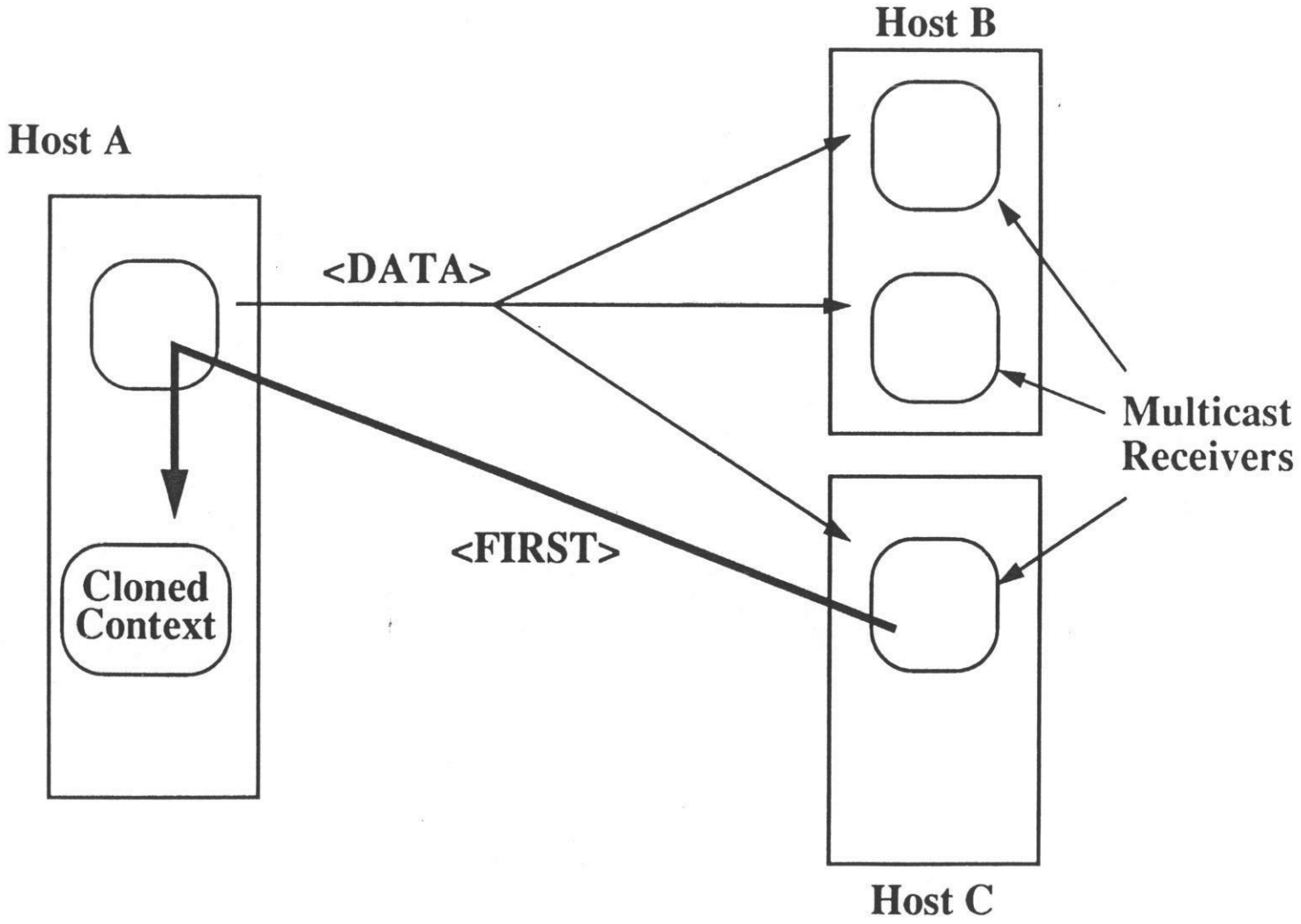
Concentration



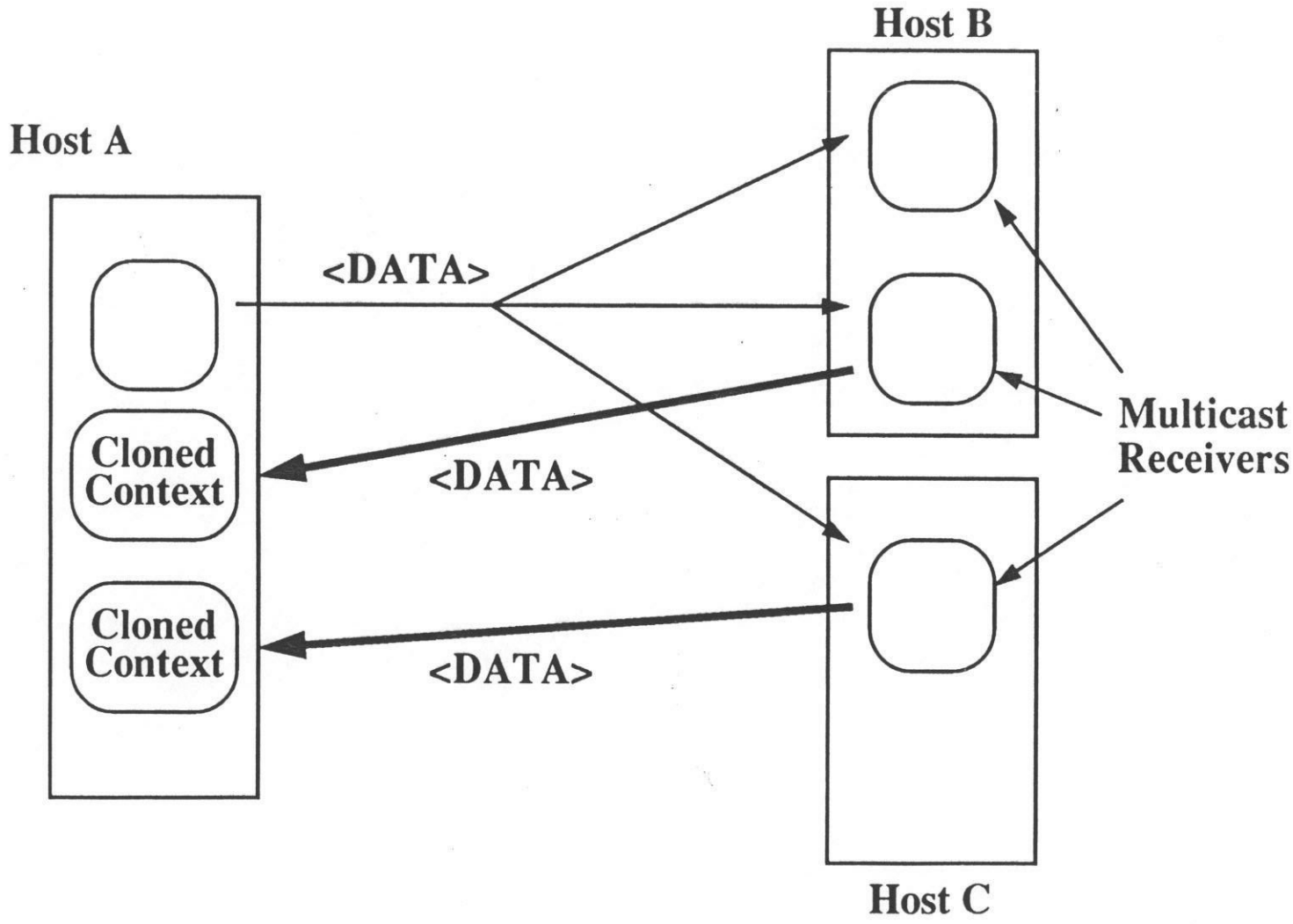


Multipoint-to-Multipoint

XTP Multicast—Cloning Heuristic



XTP Multicast—Cloning Heuristic



XTP Multicast—Summary

XTP provides a multicast facility largely orthogonal to the other features of the protocol

Semi-reliable multicast delivers data reliably to the set of receivers which are active for the duration of the distribution and which do not "lag too far behind"

In-progress join procedure permits dynamic groups

XTP heuristics include

- slotting and damping for reducing traffic of large groups
- bucket algorithm for trading off buffer space, error tolerance, and responsiveness during semi-reliable transfer
- cloning for efficiently supporting concentration

Experience

XTP currently operational on:

- Ethernet, token ring, FDDI
- DOS, pSOS, pSOS+, VRTX, AIX, Unix
- Intel 80x86, Motorola 680x0, RS/6000, SUN-4

Applications:

- file transfer
- digitized image transfer
- multimedia
- low-latency messaging
- multicast
- multi-channel telephone system

Performance

Measurements are user-to-user with transport services active

50 MHz Intel 80486, EISA bus

- 38 Mbits/sec on FDDI

IBM RS/6000 model 320J under AIX

- 5 Mbits/sec on Ethernet

25 MHz Motorola 68020, VMEbus, pSOS

- 14 Mbits/sec on FDDI

25 MHz Intel 80386, ISA bus, ZEK

- 1.9 Mbits/sec on token ring
- 4.7 Mbits/sec on Ethernet

Standardization

ANSI and ISO formed study groups to generate a *High Speed Transport Protocol* (HSTP)

XTP was submitted as a suggestion

HSTP is a modified XTP with somewhat reduced functionality and with network layer separated from transport layer

ANSI has approved two documents for submission to ISO:

- High Speed Transport Services (HSTS)
- High Speed Transport Protocol (HSTP)

Standardization is probably 2-4 years away

The XTP Engine

Idea is to separate application processing from protocol processing

An attached processor to run the protocol is one obvious approach

On the Navy's DTC-2 (SUN Sparcstation) we put a Motorola 167 (25 MHz 68040 + 8 MB memory) and a Network Peripherals FDDI interface on the VMEbus backplane

Currently conducting performance measurements

Advantage: free host for user applications

Disadvantages:

- three node path from Sparc to 68040 to FDDI
- extra data copies
- extra contention for backplane
- interaction of two operating systems

The XTP Engine

New idea is to integrate XTP and FDDI on one board

Possible targets:

- Interphase 5211 (Motorola 68040)
- SBC (Motorola 68030)
- CMC (AMD 29000)
- other "intelligent" FDDI boards

Next goal:

- an end-to-end model which predicts performance (throughput, latency) when parameterized to reflect a given architecture