コンパクトな集合表現による マルチキャスト経路制御の効率化について

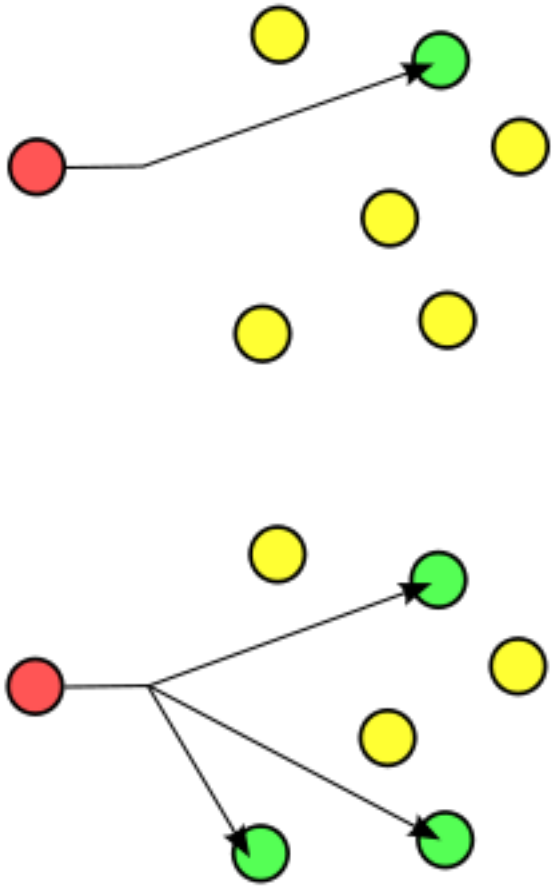
Takeru INOUE (NTT Network Innovation Labs.)

Our goal

- We are
  - Investigating innovative networking technologies for future networks
  - In an early stage of the research
- What's the future network like?
  - I don't know :-)
  - We assume
    - There will be trillions of data sources subscribed by billions of devices
      - Data sources can be Twitters, robots, sensors, and so on
      - Devices can be mobile phones, cars, home appliances, and so on
    - The data streams will be distributed to the devices
    - The devices will timely provide smart services like navigation, notification, alerting
  - As a result, we need a scalable multicast infrastructure that support a huge number of groups
    - To distribute the data streams
- Our challenges
  - To invent a new protocol to distribute such myriad of groups efficiently
  - To implement the protocol and conduct experiments

What's multicast transmission?

- Unicast transmission
  - Send messages to a single destination host
  - Internet *was* designed for unicast transmission
  - e.g., telnet, ftp
- Multicast transmission
  - Send messages to a group including multiple hosts
  - Internet *is* used for information retrieval (multicast transmission)
  - e.g., file download, live streaming, subscribe tweets



Internet must be re-designed for multicast tra

How to implement unicast routing?

- Create a routing table on each router
  - The table has several entries for *destinations*
- Route messages based on the tables
  - e.g., message from D to A
    - D sends the message on link 2, and C sends it on link 1

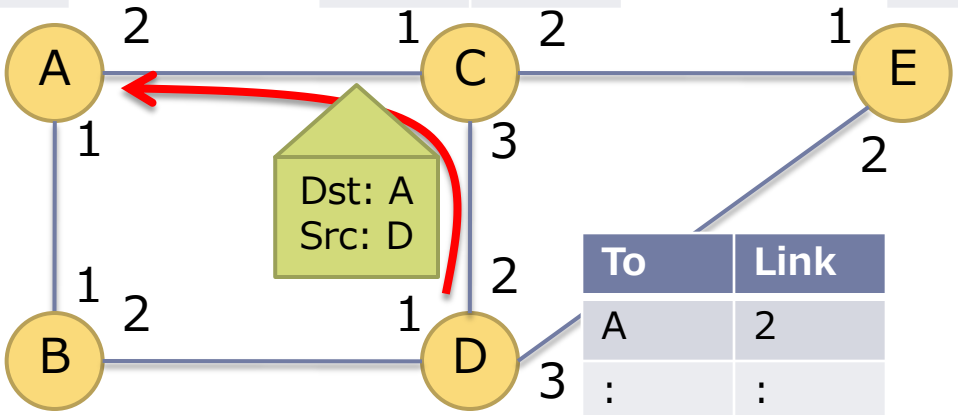
A's routing table

To	Link
B	1
C-E	2

To	Link
A	1
:	:

To	Link
A	1
:	:

Route aggregation reduces # of routing entries



To	Link
A	1
:	:

To	Link
A	2
:	:

# of routing entries < # of destinations

How to implement multicast routing?

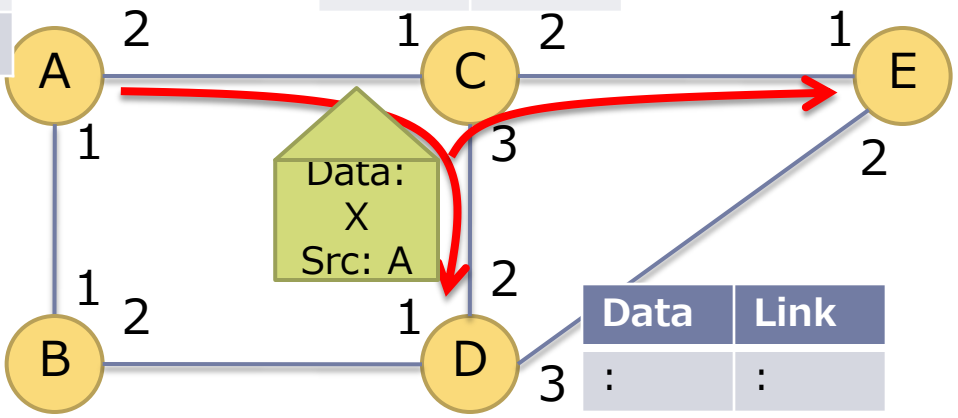
- Create a routing table on each router
  - The table has several entries for *data (groups)*
- Route messages based on the tables
  - e.g., message X sent from A to group of C, D, and E

A's routing table

Data	Link
X	2
Y	2
:	:

Data	Link
X	2, 3
:	:

Data	Link
:	:



Data	Link
:	:

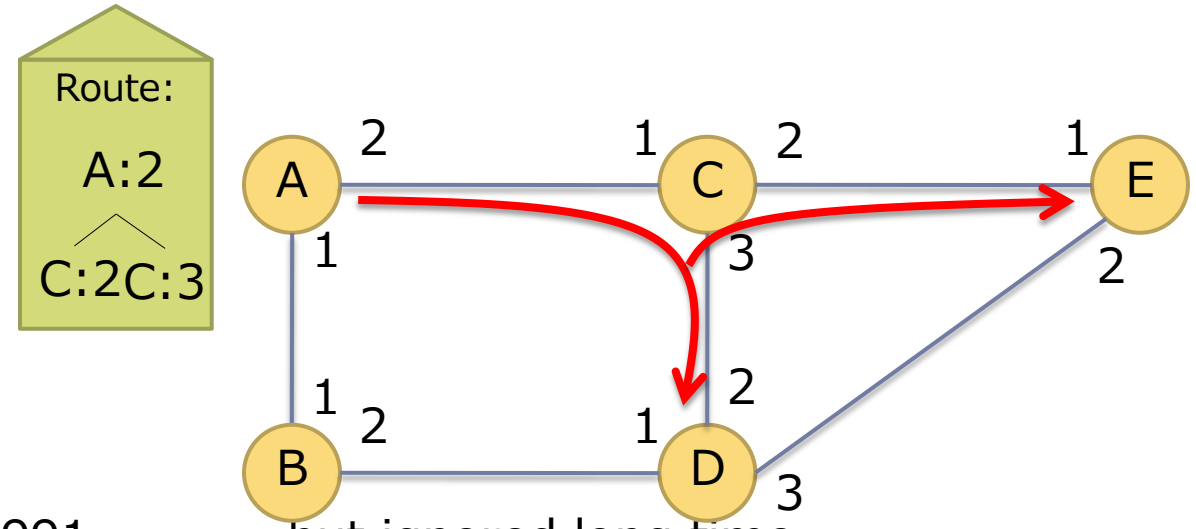
Data	Link
:	:

Difficult to aggregate entries

# of routing entries = # of data (too many)

Stateless multicast routing

- Create no routing table
- Encode a route (a set of links) into message header



- Proposed in 1991 [Peter91], but ignored long time
  - It seems impossible to send messages to *large* groups (header space is limited)
- Gathering attention recently
  - Can support a huge number of data
  - Efficient encoding scheme that represents large groups was invented

We're elaborating this approach

[Peter91] T. Peter et al., "Multicast source routing in packet-switched networks," IN

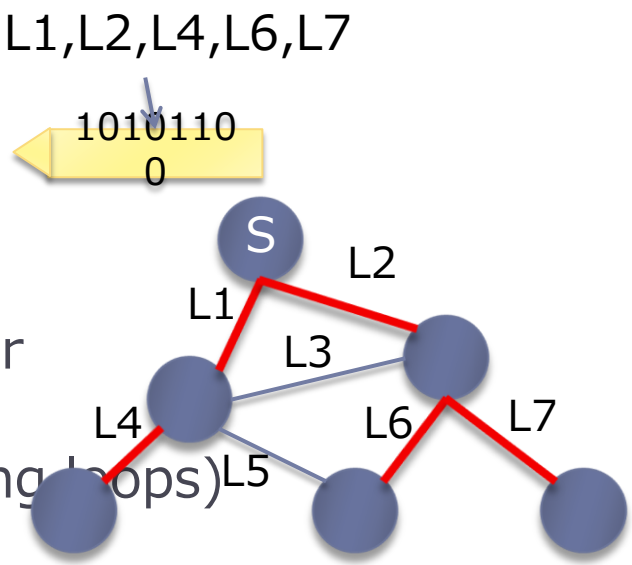
How to compress a set of links?

- Efficient set representation is a key technology for large groups
  - Small headers reduce header overheads
  - Simple decompression leads to fast forwarding processing

- Bitmaps [Chiang94]
  - Set bits according to destinations
    - Bits are assigned to nodes in advance
  - Forward it to destinations
  - Efficient but just for static topology
- Bloom filters [Ratnasamy06]
  - Encode links consisting a tree to filter
  - Forward it to links encoded
  - Efficient but with extra traffic (causing loops)

ZDD's cousin

- Another improvement is needed

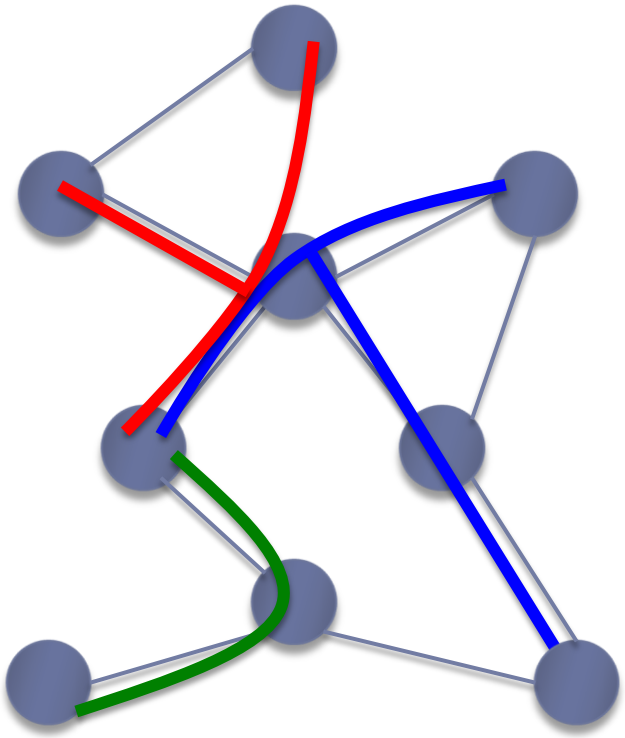


Multicasting for Bloom filters

[Chiang94] C. Chiang et al., "Multi-address Encoding for Multicast," PC
[Ratnasamy06] S. Ratnasamy et al., "Revisiting IP multicast," SIGCOM

What to be encoded to headers?

- Fewer elements to be encoded are desired
  - Fewer elements need a smaller header (smaller overhead)
- Encoding combinations of links, not links themselves
  - How to choose an optimum combinations?
    - There are billions of possible combinations
  - Minimum combinations covering frequently used trees are required
    - Because they are maintained by routers
  - Does BDD/ZDD help to choose them?

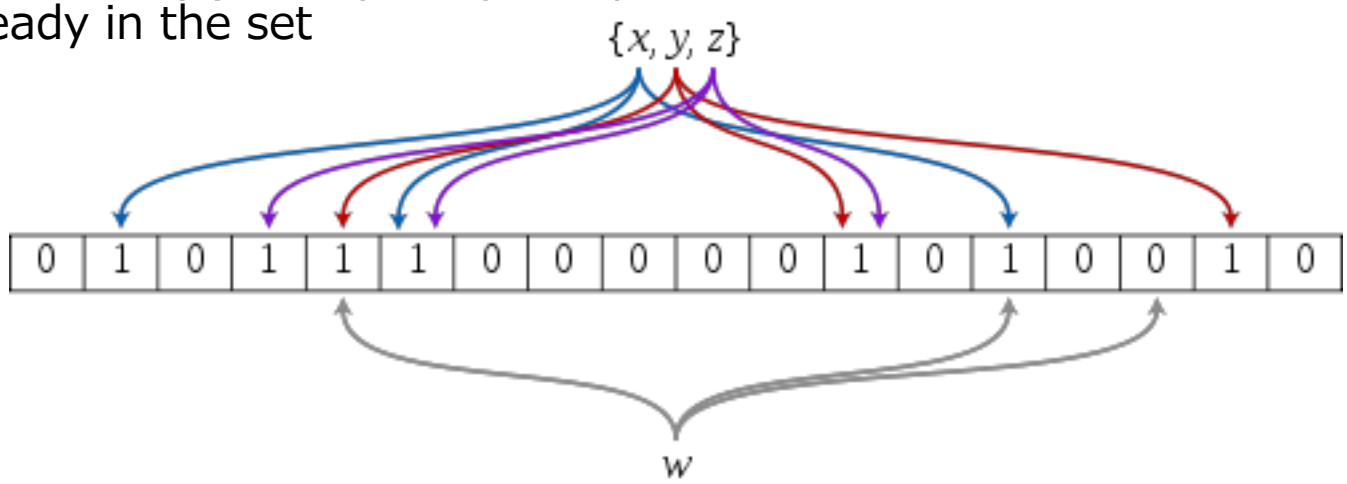


Only 3 elements constitutes a t
while 8 links for traditional met

Appendices

What's Bloom filters?

- From wikipedia.org
  - The Bloom filter, conceived by Burton Howard Bloom in 1970, is a space-efficient probabilistic data structure that is used to test whether an element is a member of a set
  - False positives are possible, but false negatives are not
  - While risking false positives, Bloom filters have a strong space advantage over other data structures for representing sets
  - Bloom filters also have the unusual property that the time needed to either add items or to check whether an item is in the set is a fixed constant, O(k), completely independent of the number of items already in the set



An example of a Bloom filter, representing the set {x, y, z}

What's Bloom filters?

- Variables

Var.	Description
BF	Bloom filter (bit string of m-bit)
EF	Empty filter (all-zero)
h_i	i-th hash function (0 < I <= k)

Range of hash functions is [0,m-1]

- To initiali
  - BF = EF
- To add an element
  - BF[$h_i(x)$] = 1
- To query for an element
  - F = EF
  - F[$h_i(x)$] = 1
  - BF & F == F
- Constant time, O(k), needed to add or query
  - Independent of the number of items
  - Can be implemented in hardware and executed in para

For a Bloom filter of m=8 and k=2, add elements X and Y to, and query for some elements

```
00000000
```
Add X
```
01001000
```
Add Y
```
01001100
```

X and Y are found
Z is not found
W is found (false positi

```
X: 01001000
Y: 01000100
Z: 10000100
W: 00001100
```