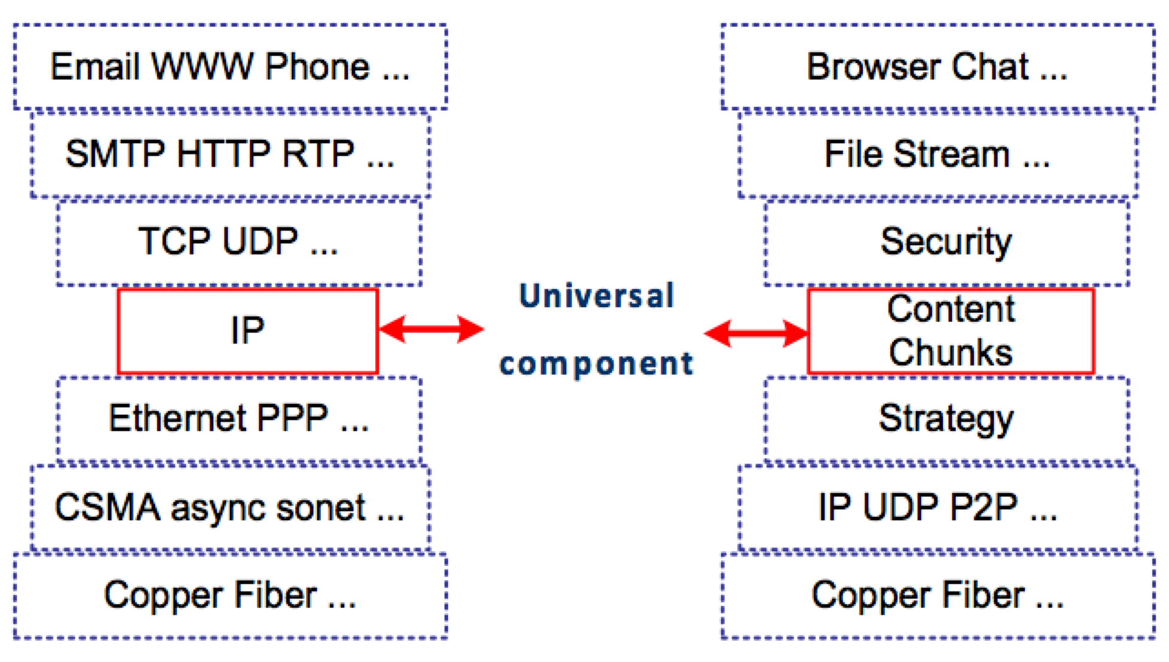
A Consideration on Congestion Control in CCN

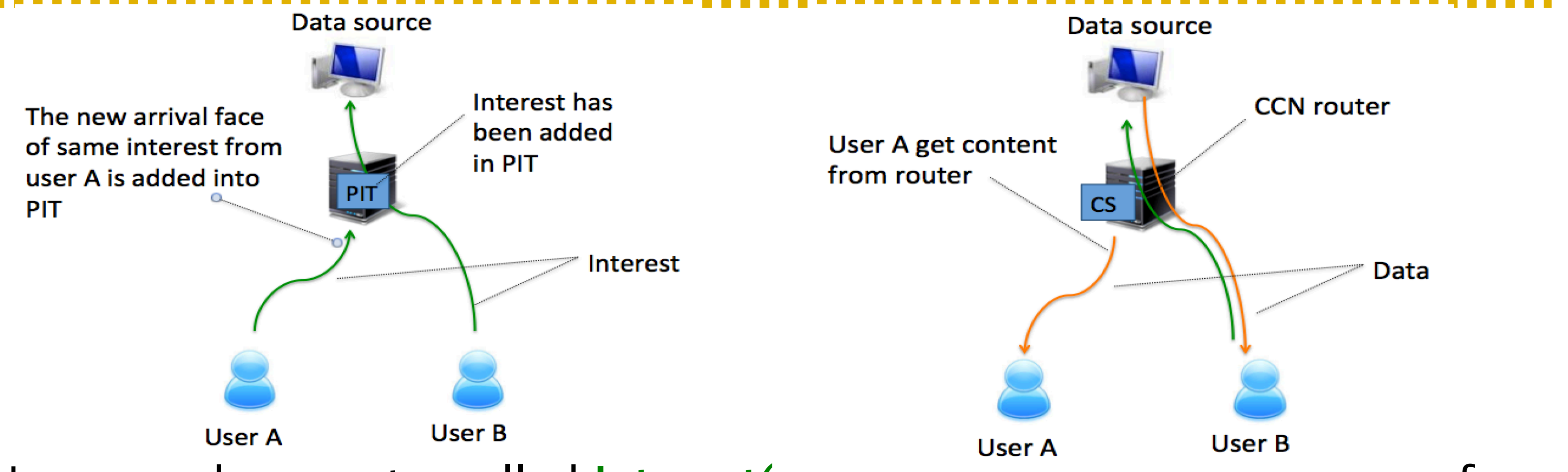
Yu Wang, Takeshi Muto, Zhou Su, Jiro Katto and Suphakit Awiphan
Graduate School of Fundamental Science and Engineering, Waseda University, Tokyo, Japan, E-mail: wang@katto.comm.waseda.ac.jp

CCN(Content Centric Network)

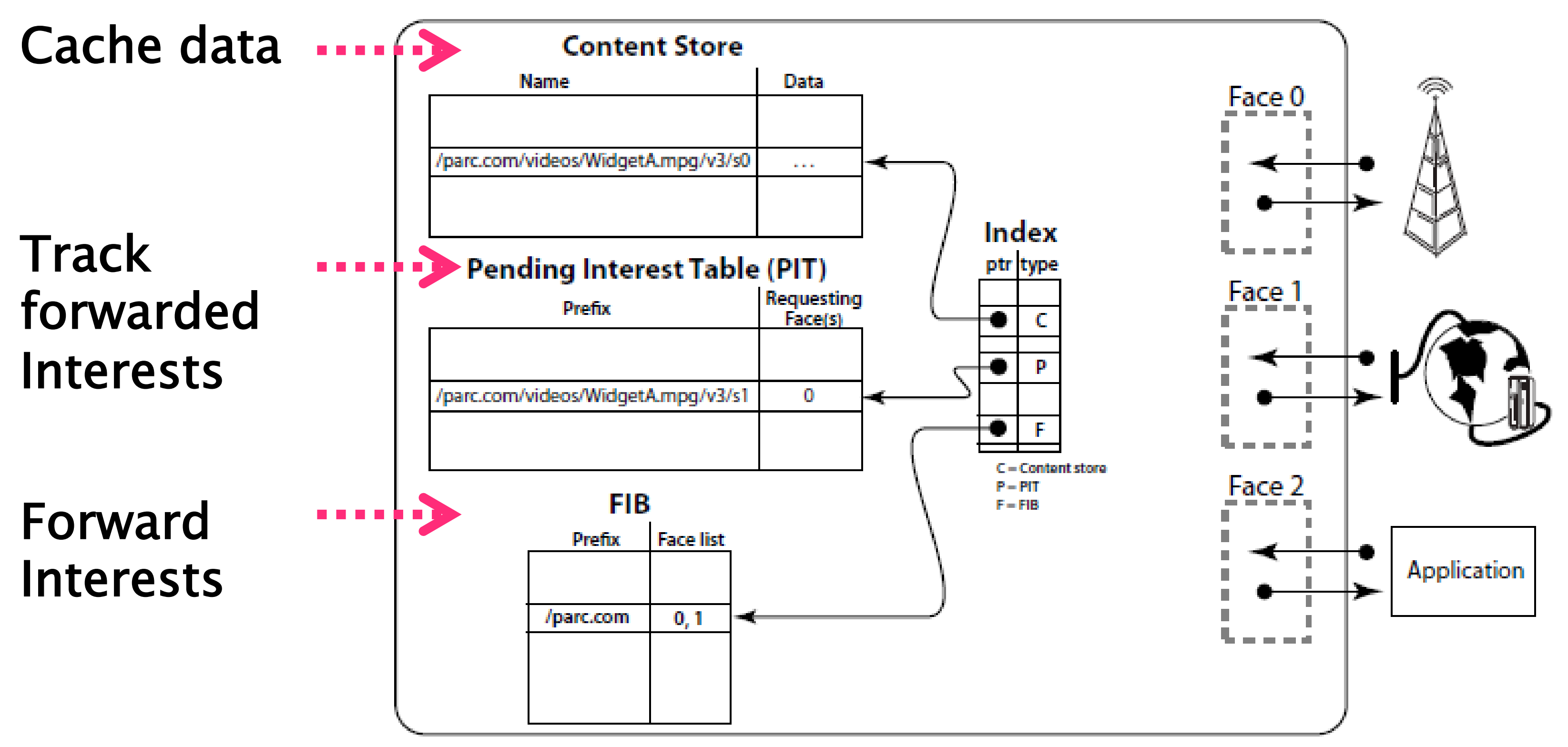


Protocol stacks for IP and CCN network

CCN[1] allows users to focus on the data's names rather than the location where that data is to be retrieved. This is realized through replacing current forwarding mechanism of IP addresses with a mechanism based on named contents


Data source
The new arrival face of same interest from user A is added into PIT
Interest has been added in PIT
PIT
Interest
User A
User B
Data source
CCN router
User A get content from router
CS
Data
User A
User B

- Users send requests, called Interest(->), which contains the name of data
- Any node hearing the Interest and having data can respond with Data(->)



Congestion control of CCN:

Congestion Control:

Congestion happens when a link or node is carrying so much data, and it can cause delay and packet loss. This affects the bandwidth badly. So congestion control helps to avoid or reduce congestions.

Basic rules[1]:

- One interest retrieves at most one data packet
- Interest packets serve the role of window advertisements in TCP. Receivers can dynamically vary the window size by varying the interests that it issues to realize congestion control

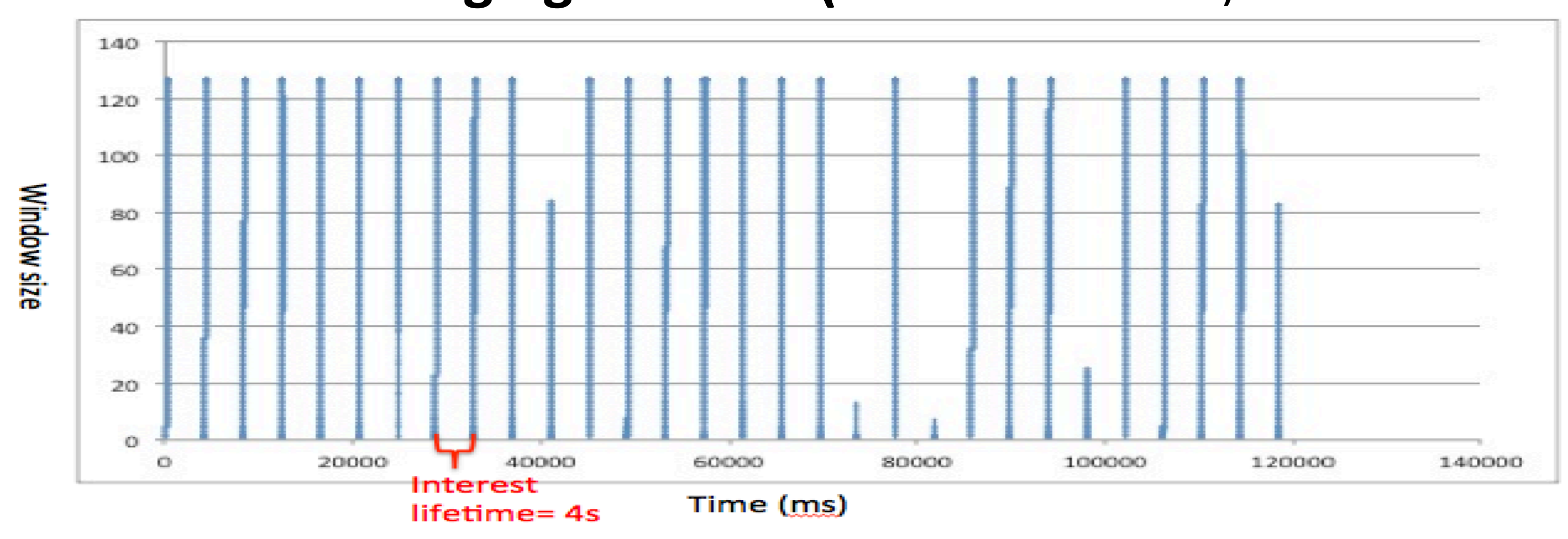
Congestion control algorithm in CCNx[4] (ccncatchunks2)

Algorithm: Congestion control by adjusting window size

```
At Up-Call arrival from the daemon:
If ( it signals an EXPIRED interest)
    { current_window = 1;
      re-sent the expired interest;}
elseif ( it signals an OUT-OF-ORDER Data)
    current_window --;
elseif ( it is a regualr Data && current_window < MAXIMUM_WINDOW)
    current_winodw ++;
```

**EXPIRED time out is default set to 4 seconds, MAXIMUM_WINDOW <= 127*
**This congestion control of CCNx is realized in Application layer. That is, Packets are tunneled within TCP or UDP flows running over IP*

Window size changing for CCNx(ccncatchunks2)



Window size changing with UDP connection between two nodes in LAN (loss rate 0.1%)

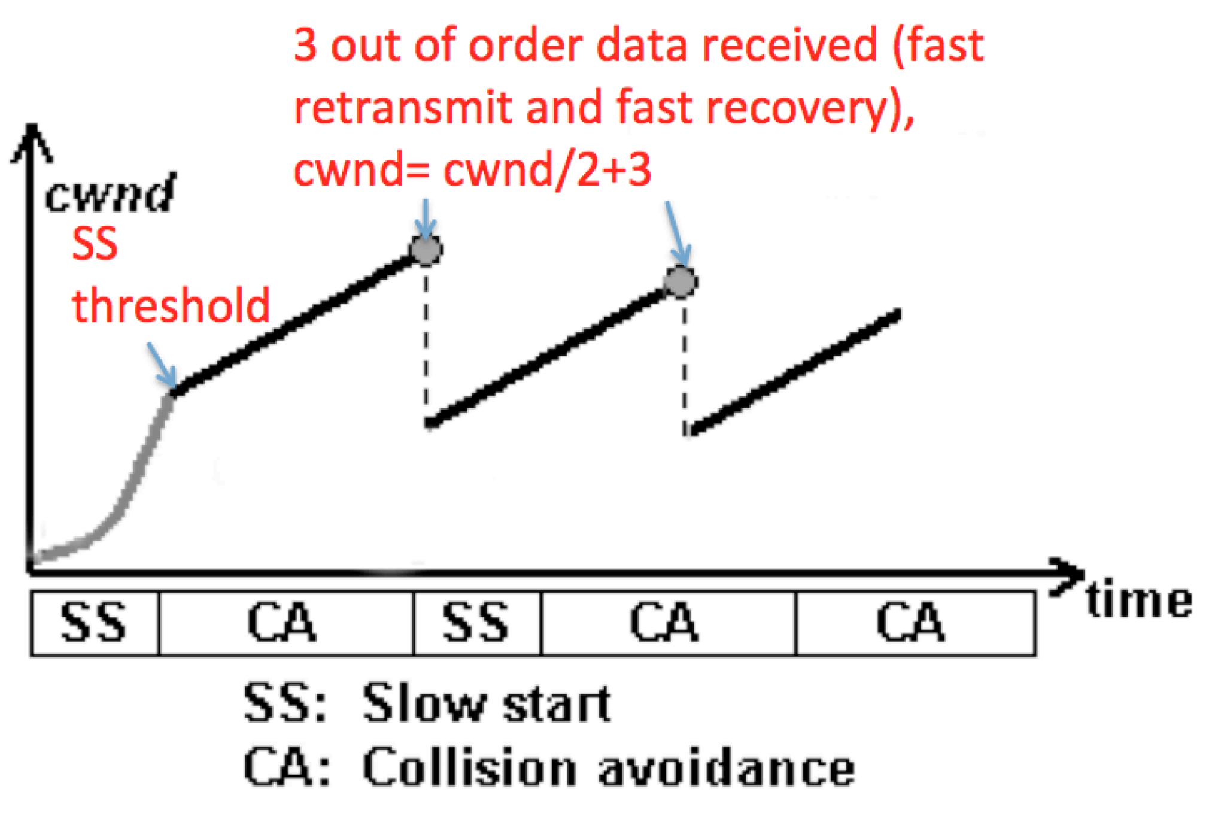
Analysis:

the average transmission bit rate is very low due to roughly changing window size and 4 seconds' untimely reaction of timeouts

Acknowledgment: This paper has been carried out in the framework of the FP7/NICT EU-JAPAN GreenICN project.

Proposal 1: Receiver-driven TCP-Reno like congestion control

This algorithm is TCP-Reno like algorithm which contains slow-start, congestion avoidance, fast retransmit, and fast recovery, but receiver driven.

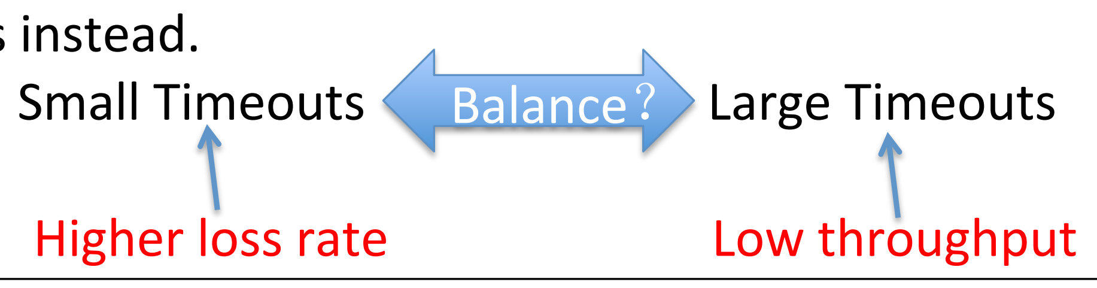


Fast Retransmit and Fast Recovery:
immediately retransmits after 3 out-of-order Data received;
ssthresh = cwnd/2 +3
Interest for missing Data retransmitted immediately

Slow Start:	**Congestion Avoidance:**
Start with cwnd = 1	Start when cwnd >= ssthreshold
On receiving ordered data cwnd ++	On receiving ordered data cwnd = cwnd + 1/cwnd
Exponential growth of cwnd	Linear growth of cwnd

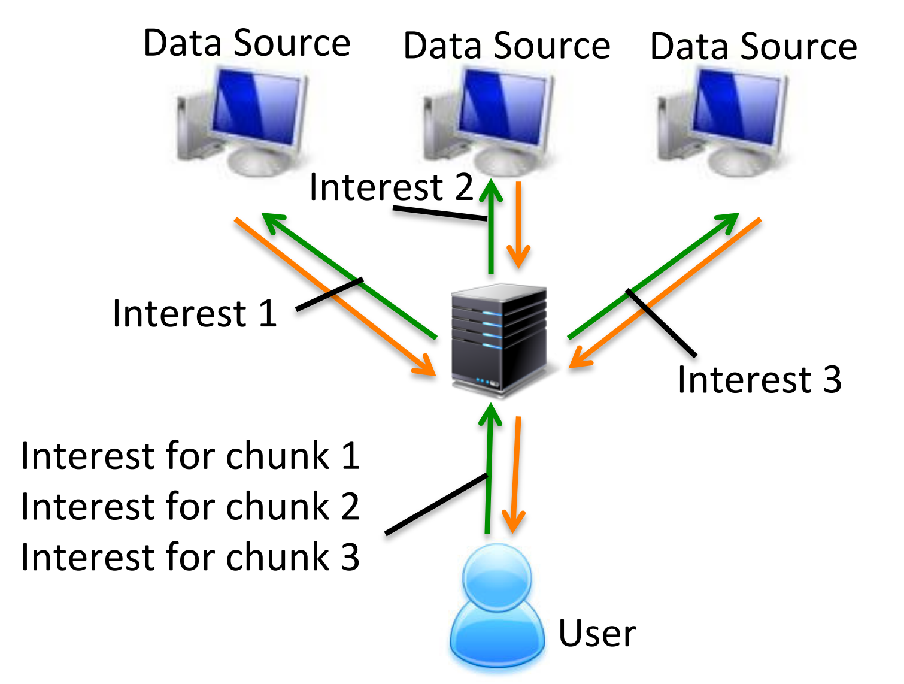
Proposal 2: Adaptive timeout

We want to use eRTT(estimated round trip time) as an adaptive parameter to decide timeouts of each interest and has a fast retransmission. Ccncatchunks2 uses a fixed 4s timeouts instead.



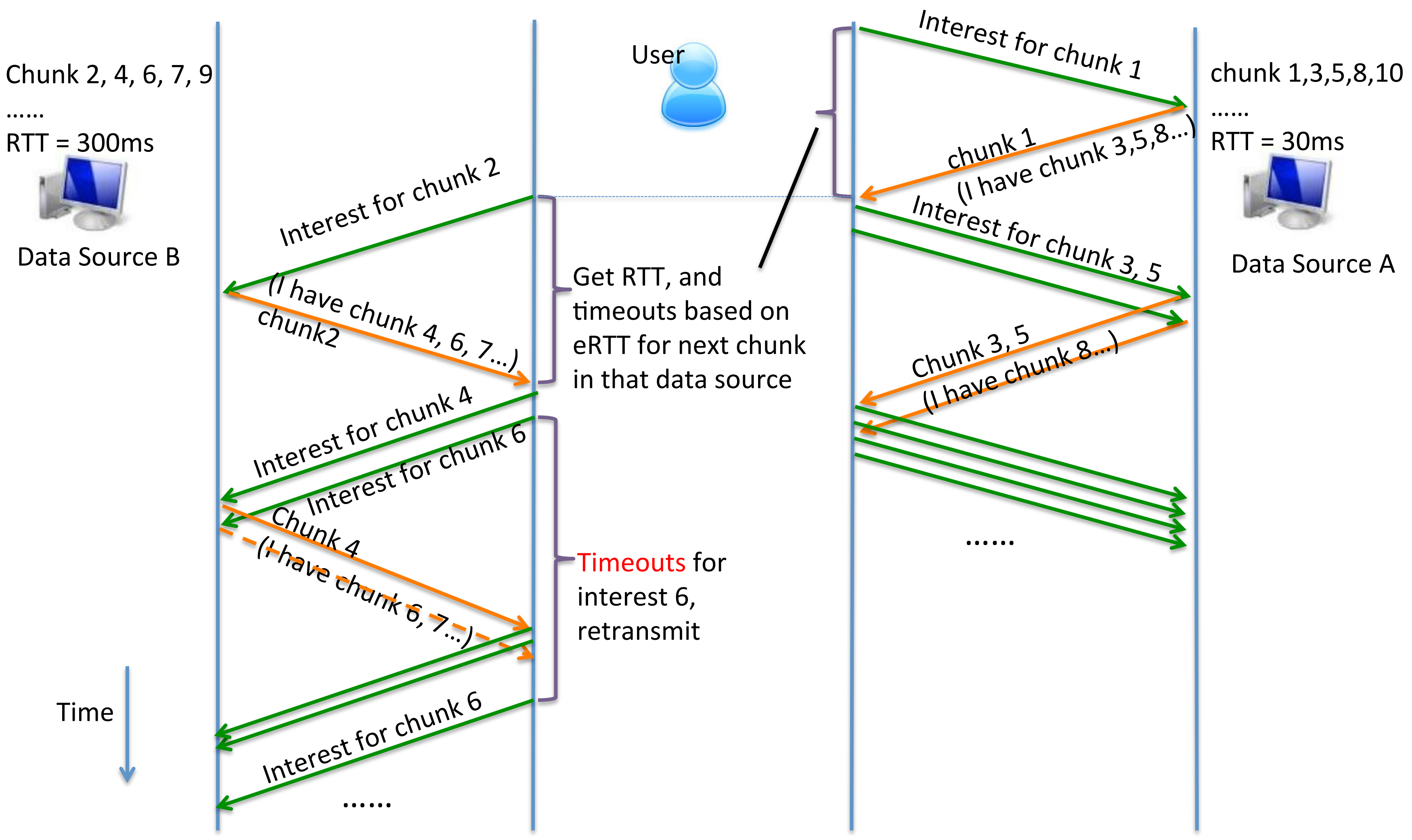
Proposal 3: Content based congestion protocol

Proposal 1 and proposal 2 are all based on point to point communication, so it may not appropriate for CCN.



CCN users may retrieve Data chunk from a number of different nodes/ cache [1]

Timeouts which base on eRTT and fast retransmission in TCP-Reno are not appropriate because data source change frequently.



User wants to retrieve a content segmented in 10 chunks which stored in two separate Date Sources

In this topology:

- In the chunk data, information about remaining chunks in a particular Data source is shown
- Each data source will have its own eRTT, and this is used for setting timeouts of next chunk in that Data Source
- Congestion controls are realized separately and simultaneously in different Data sources containing different chunks of a required content.
- This makes proposal 2 more applicable to CCN

Conclusion: We propose the possibilities to use TCP-Reno like congestion control and adaptive timeouts in CCN. According to the multi-sources of CCN, we propose multi-thread congestion control in CCN. And later, we will try to carry out our experiments.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking Named Content", Proc. ACM CoNEXT 2009, pp.1-12, 2009.
[2] Suphakit Awiphan, Takeshi Muto, Yu Wang, Zhou Su and Jiro Katto"Video Streaming over Content Centric Networking, Experimental Studies on PlanetLab",ComComAP, Hong Kong, China, April. 2013
[3]Paolo VIOTTI, "Caching and Transport Protocols Performance in Content-Centric Networks", Master Thesis, September 2010
[4] CCNx Project, [Online], http://www.ccnx.org
[5] Stefano Salsano, Andrea Detti, Matteo Cancellieri, Matteo Pomposini, Nicola Blefari-Melazzi, "Transport-Layer Issues in Information Centric Networks", ACM SIGCOMM Workshop on Information-Centric Networking (ICN-2012), Helsinki, Finland , August 2012.
[6] L. Saino, C. Cocora, G. Pavlou, "CCTCP: A Scalable Receiver-driven Congestion Control Protocol for Content Centric Networking", In IEEE ICC 2013 - Next-Generation Networking Symposium (ICC'13 NGN), Budapest, Hungary, June 2013