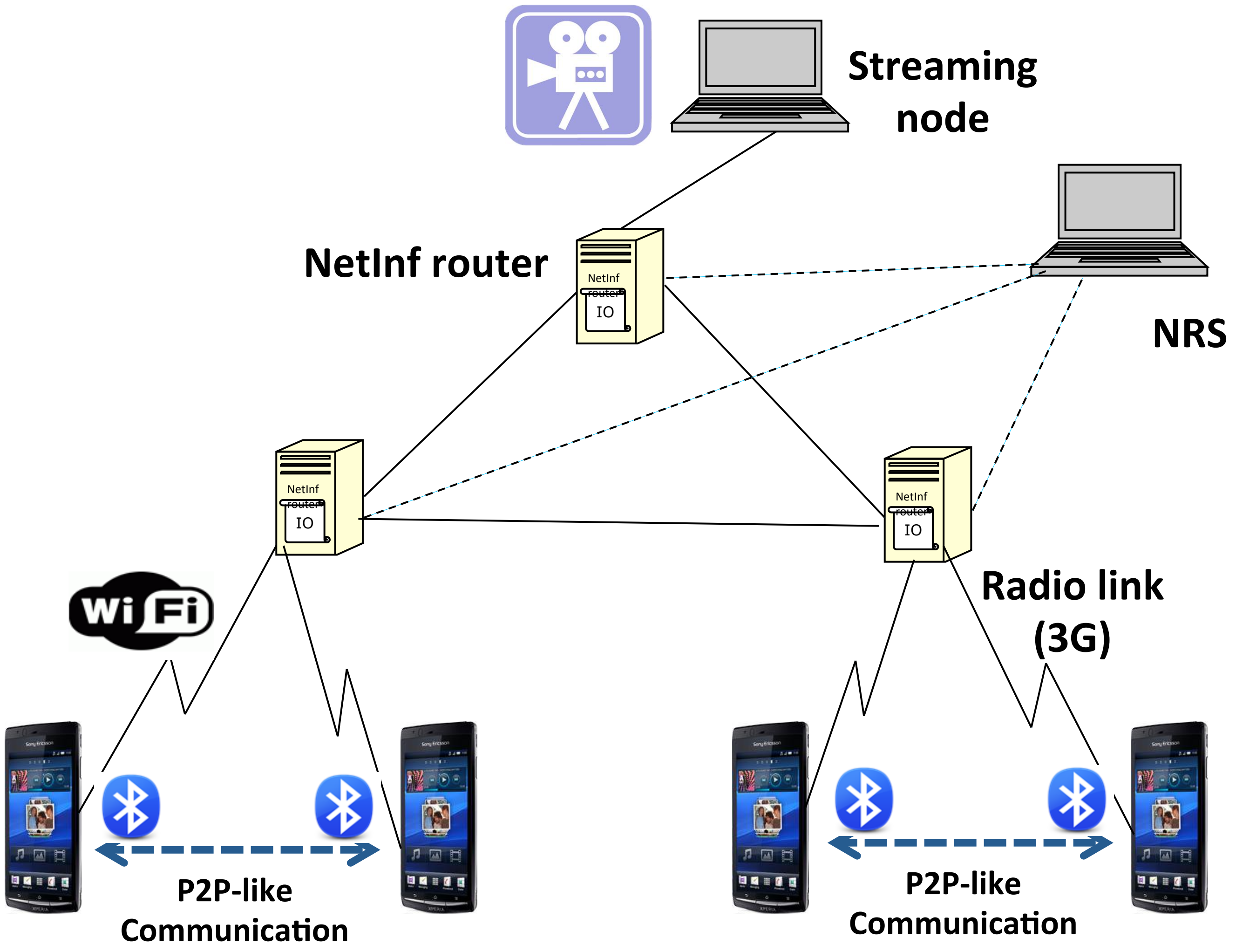
NetInf streaming

ICNRG Interim, Stockholm, Feb 14-15th 2013

Börje Ohlman

NetInf Streaming demo setup



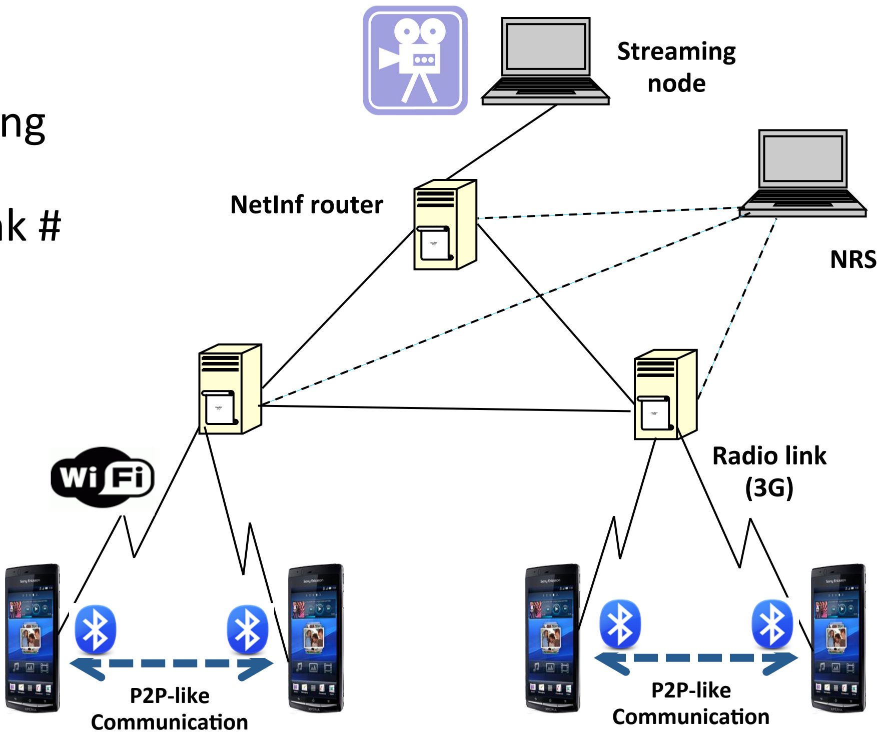
How streaming works in NetInf

To publish a stream:

- Create the NDO ID for the stream by hashing the Stream name
- Publish the Stream NDO with current chunk # in metadata

For a user to connect to a stream:

- Request the stream NDO
- Decide where to start playing the stream.
  - Live: chunk=current
  - Start: chunk=1
  - Starting from minute x: chunk=x*(chunklength/min)
- Request subsequent chunks



Responding to a stream request:

- When responding to a GET request for the stream NDO, that NDO MUST be marked as non-cacheable.
- When responding to a GET request for the stream-chunk NDO, that NDO MUST NOT be marked as non-cacheable.

All nodes MUST understand the non-cacheable marking.

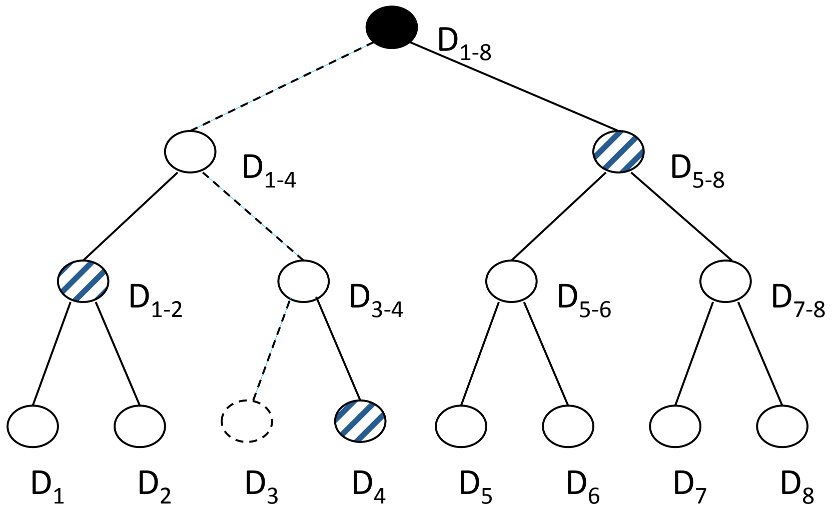
Issue: Name-data-integrity lost with sequential chunk numbering

- Resolution alternatives looked at:
  - Sign each chunk
  - TESLA
  - eFFS
  - Algorithmic identifiers

Verification in NetInf streaming

Verification alternatives:

1. The chunks are grouped into blocks that are signed. The block size is recorded in the metadata of the NDO identified by the stream id. The metadata of each chunk NDO contains the signed block digest and the digests of the other chunks in the block. This allows for verification of each chunk independently immediately when received. For details, see [1].
2. In the future in many scenarios signing individual chunks might be feasible (we like IETF PPSP WG stays open for both these options).
3. For applications that have its own security mechanisms at higher layers signing might not be needed, e.g. like distribution of broadcast TV with dedicated set top boxes.



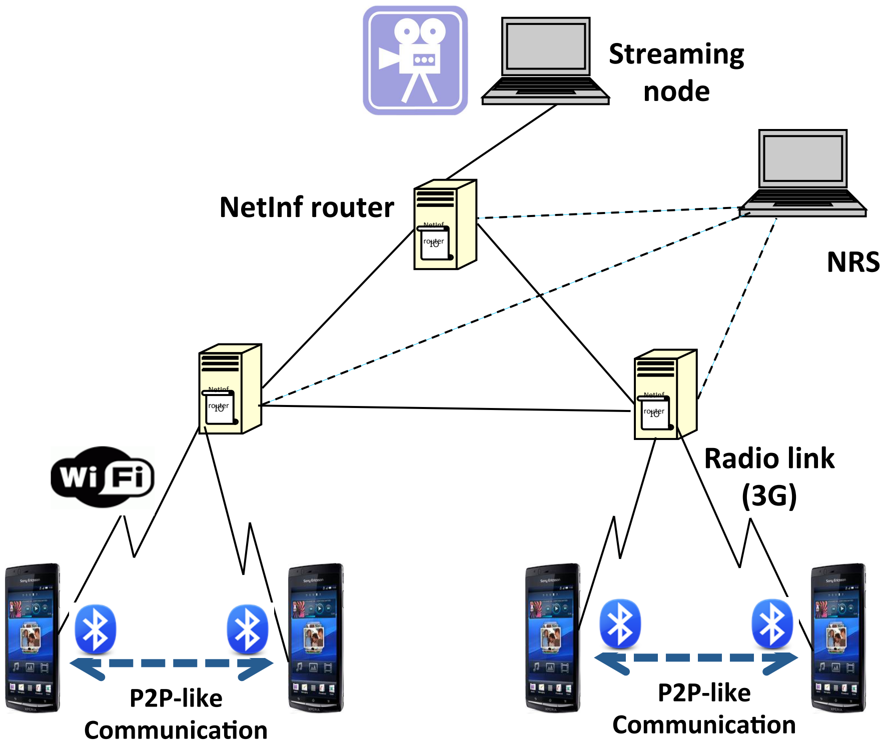
[1] C.Wong, S. Lam, *Digital Signatures for Flows and Multicasts*, IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 7, NO. 4, AUGUST 1999].

Why TESLA is not a good choice for NetInf streaming

- TESLA relies on time syncronization between sender and receiver.
- For this to be a reasonable assumption TESLA require **one** homogeneous transport network with predictable transport delays.
- This is a problem in a multiaccess scenario with multihomed devices as the key disclosure information can reach an attacker faster than expected on an alternative network connection and thus make an attack in another network possible.
- It seems that TESLA is a mechanism only works for the live streaming case. It would be nice with a mechanism that also works for later playback and DTN use cases so that not an additional mechanism is needed for these cases.

Final notes

- For concurrent viewers request aggregation is essential.
- Currently NetInf is caching a full object before forwarding it, this is obviously causing long delays for large objects.
  - Cut through forwarding is needed.


Streaming node
NetInf router
NRS
Radio link (3G)
P2P-like Communication
P2P-like Communication