

# Enforcing “Sticky” Security Policies throughout a Distributed Application

David Chadwick, Stijn Lievens

University of Kent

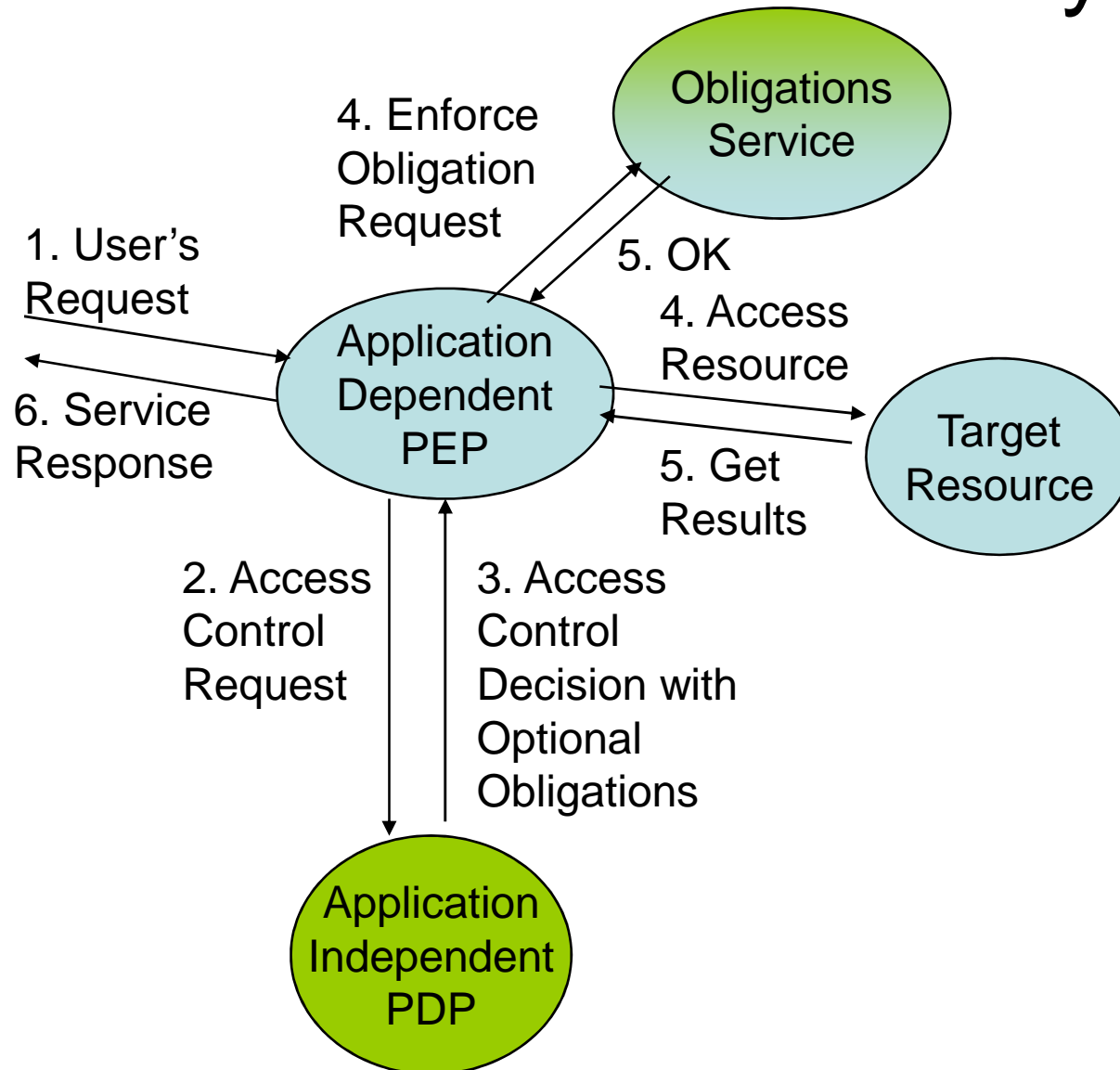
# Acknowledgements

- The work being presented here is current research supported by the EC FP7 TAS3 project
- Thanks to all in the project team who contributed to this
- We are only 11 months into a 4 year project so this work will evolve and develop in the coming years
  - and you can contribute to it now 😊

# Contents

- Policy Enforcement in Existing Systems
- Sticky policies
- Sticking policies to data
- Enforcing sticky policies
- Transferring sticky policies throughout a distributed system

# Policy Enforcement in Existing State of the Art Systems

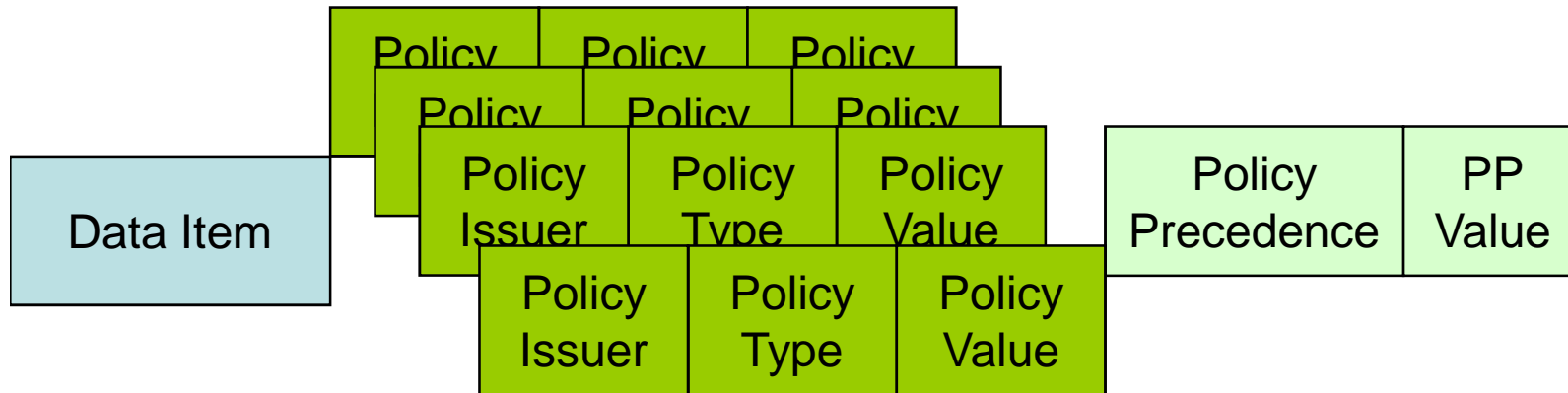


- Still many issues about obligation enforcement
  - Before, With, or After resource access
  - Two phase commit
  - Obligation language
  - Application independent or dependent

# Sticky Policies

- Policies that control how data is to be accessed and used, and that accompany data throughout an entire distributed system
- Policies may be access control policies, privacy policies, obligation policies etc
- Policies may be written in a variety of policy languages
- Adds a new level of complexity

# Envisaged Data Structure

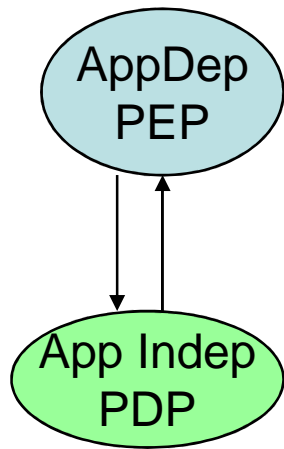


- Policy Type determines which PDP can evaluate it. Must be globally unique e.g. URN
- When multiple policies/PDPs give different decision values, what is the overall access decision?
- Policy Precedence, set by Authoritative Source, will determine the policy combining rule to use e.g. trust policy over-rides, Govt policy over-rides, data subject's policy decides, any deny takes precedence, etc.
  - Active ongoing research to determine data structure and precedence values

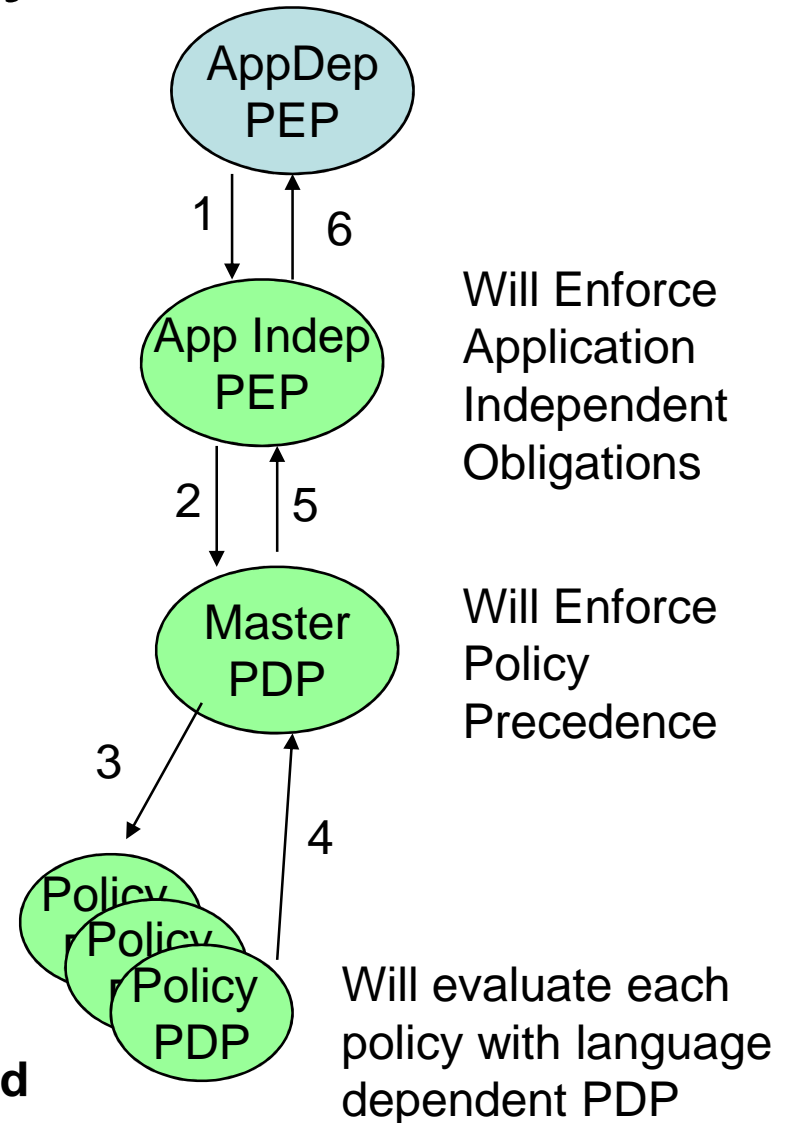
# Other Issues

- How are data items with sticky policies to be aggregated. How is a combined sticky policy created? Sometimes a new policy for the aggregated data might be needed that is not simply a combination of the original policies
  - E.g. If medical anonymised data that has been joined from multiple sources, it may become personally identifying and therefore need a more restrictive policy than that for the constituent parts
- How are data items with sticky policies to be disaggregated? How is a fractional sticky policy created?

# Structure of Application Independent Policy Enforcement

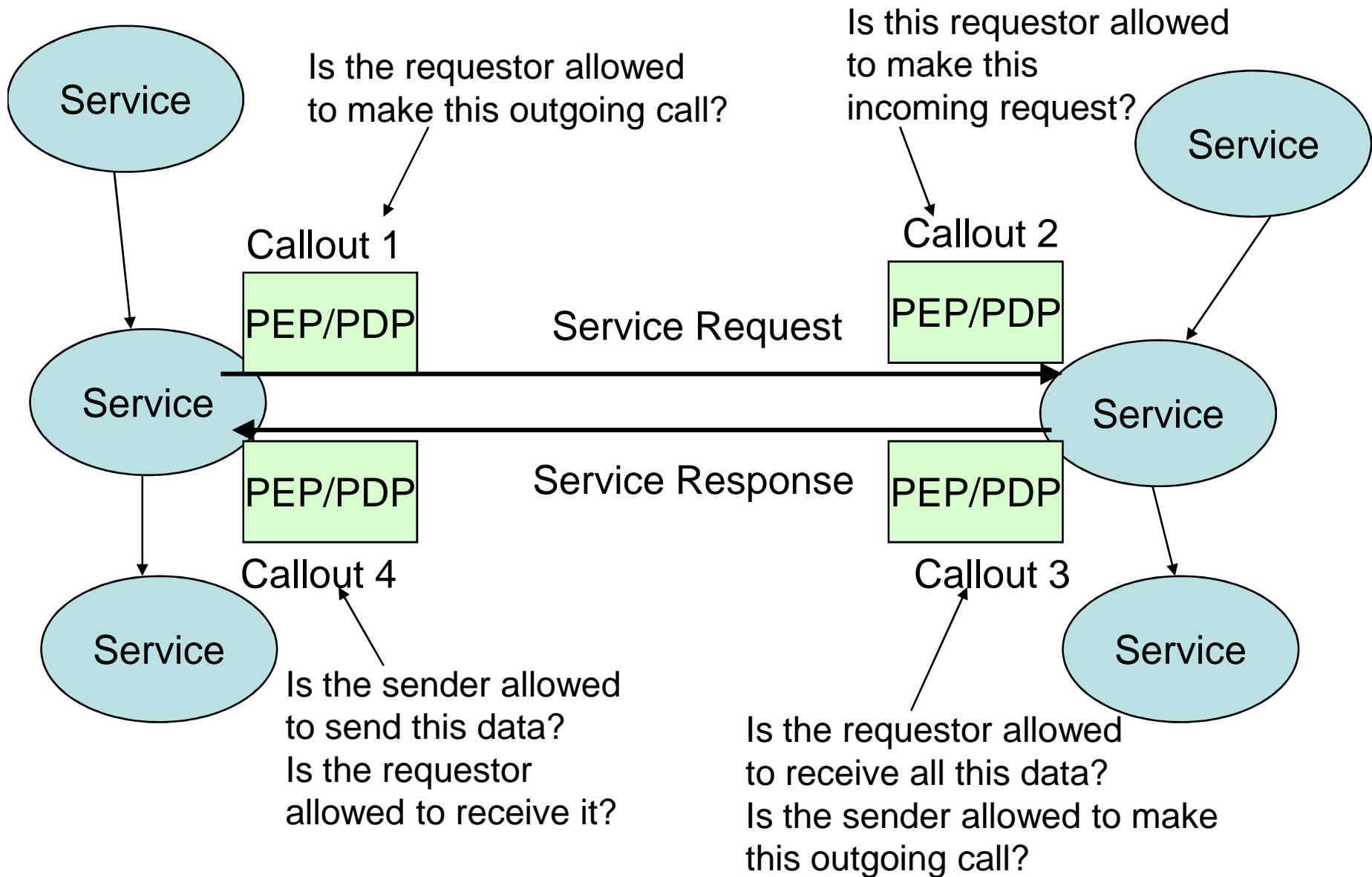


**Existing Systems**



**Proposed System**

# When are Sticky Policies Enforced?



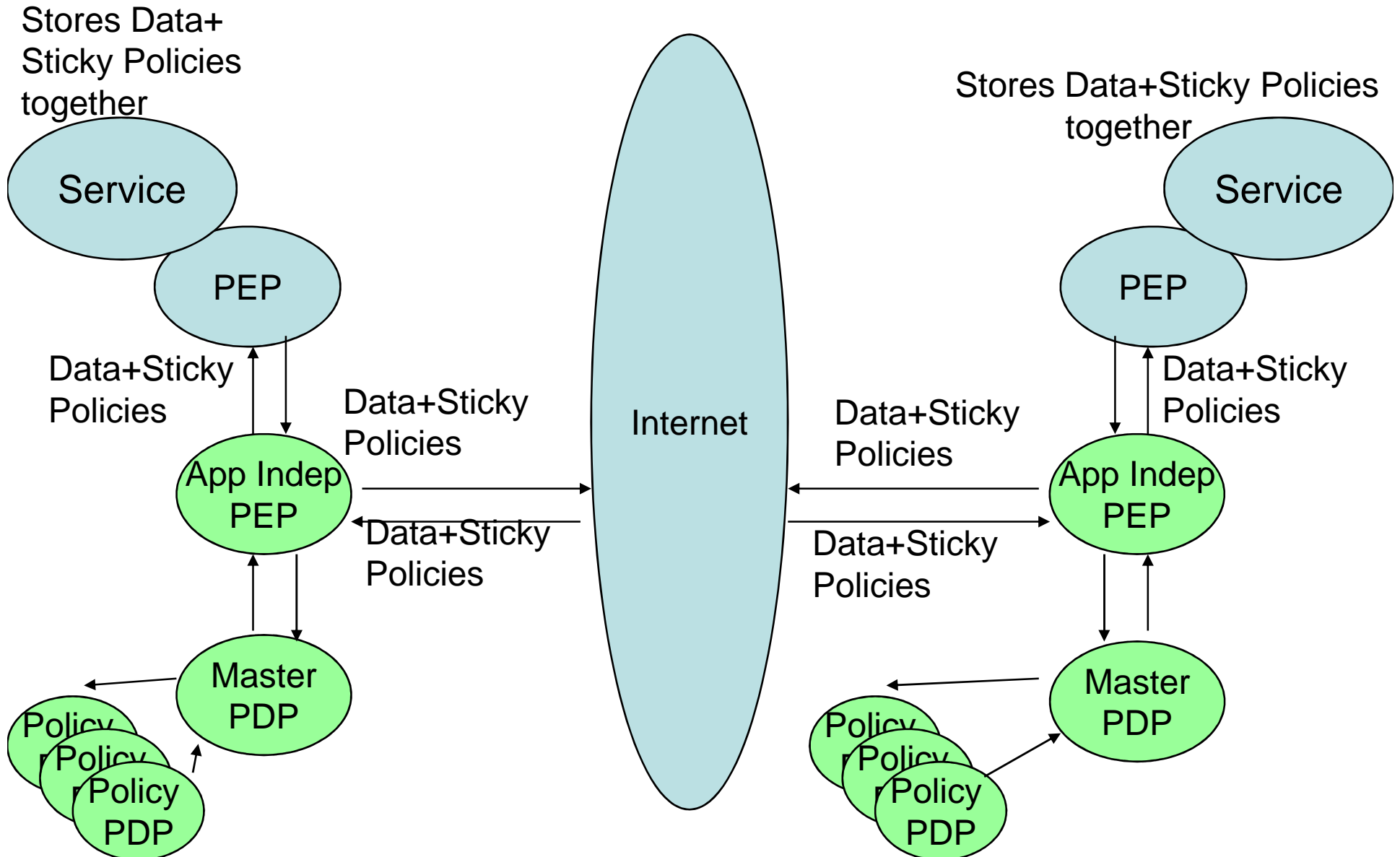
# How are Sticky Policies to be Transferred?

- Mont et al<sup>1</sup> propose to stick policies to application data by using Identity Based Encryption
    - Use policy as encryption key for the data
  - But what about existing (legacy) applications?
  - What is the trust/security model?
    - Don't know if recipient is trusted or not
    - Requires too much trust in TTP/TA
    - If confidentiality in transfer is required then can use SSL (IBE allows TTP to read it as well)
  - So we are proposing more conventional means of transfer that can cater for legacy applications as well as new ones
- <sup>1</sup>Mont, M.C.; Pearson, S.; Bramhall, P. "Towards accountable management of identity and privacy: sticky policies and enforceable tracing services". Proc 14th Int Workshop on Database and Expert Systems Applications, 1-5 Sept 2003. Page(s): 377 - 382

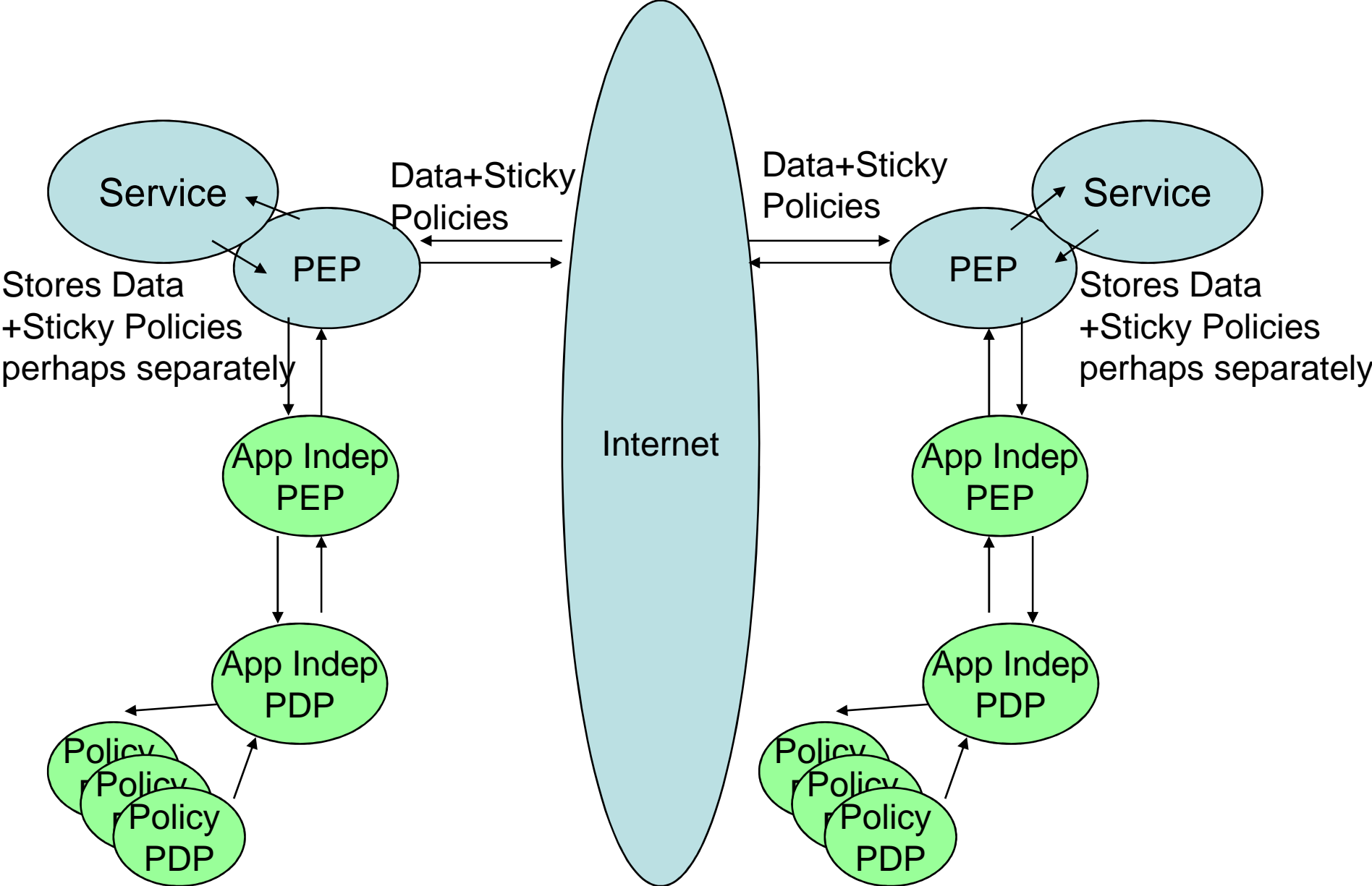
# Three Models are Proposed

- Encapsulating Security Layer for new applications that can store data and sticky policies
- Application Protocol Enhancement for existing applications that have flexibility in their protocol to carry sticky policies
- Back Channel for legacy applications that cannot store or transfer policies

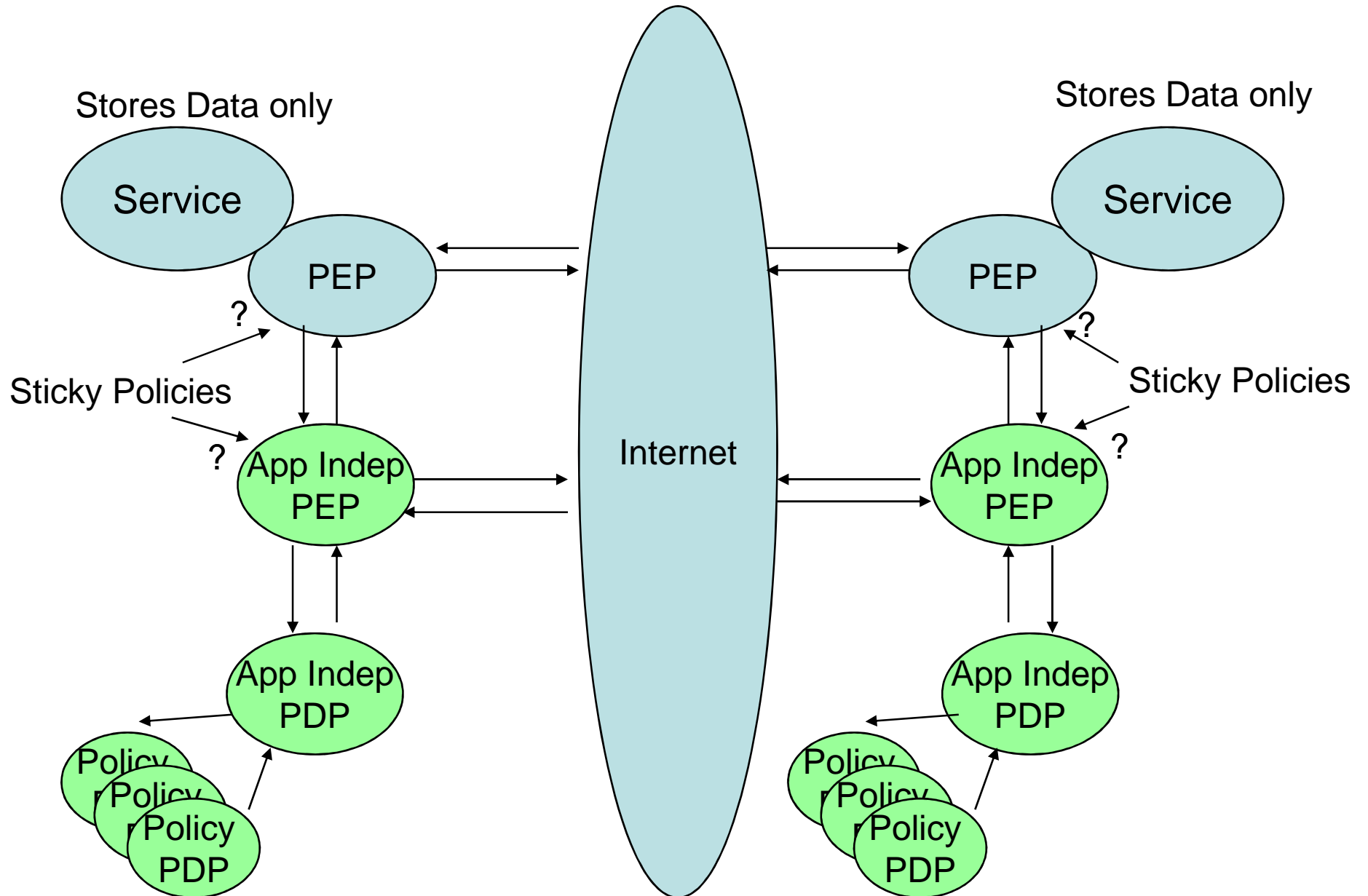
# The Encapsulating Security Layer Model for New Applications



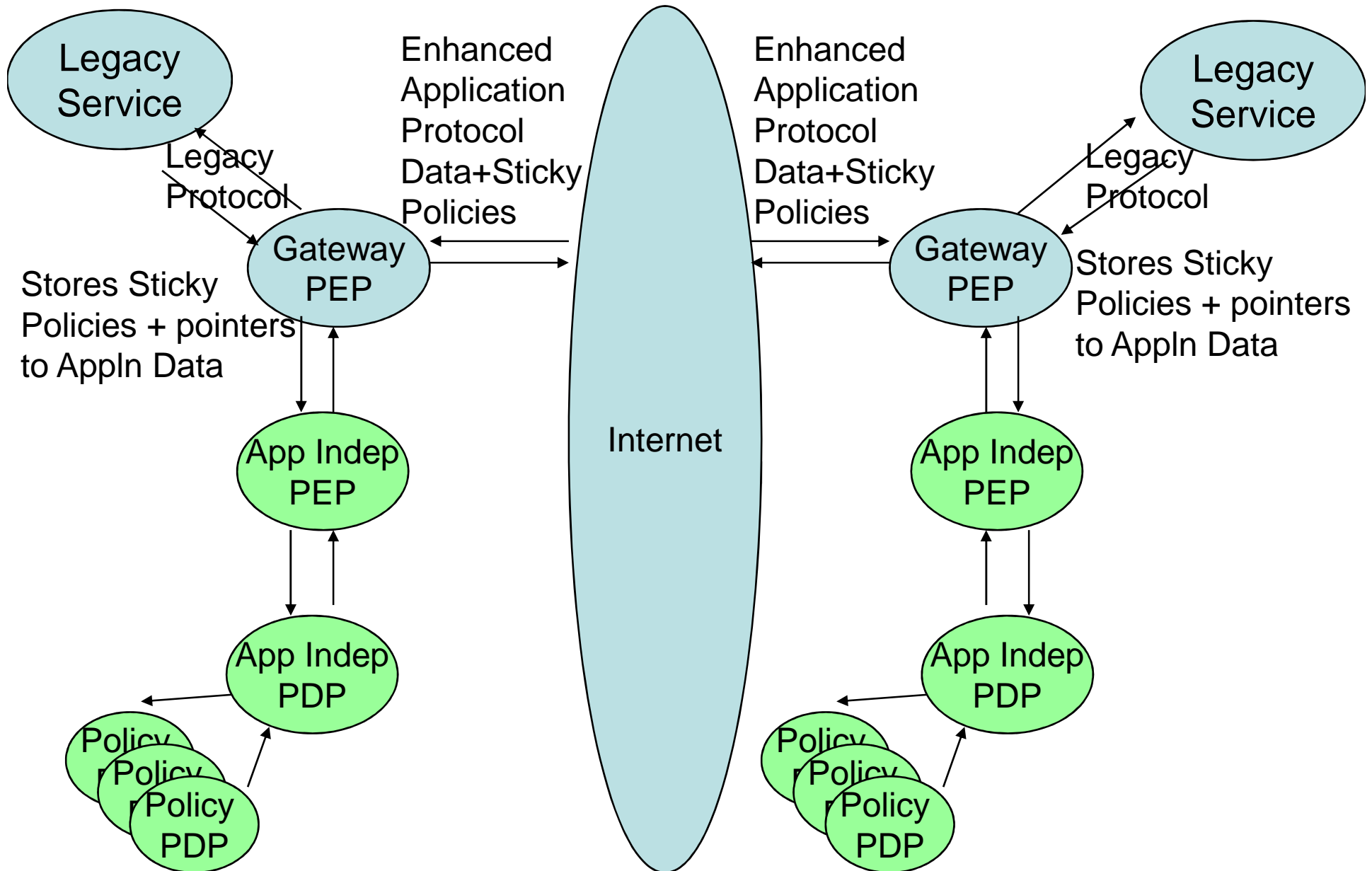
# The Application Protocol Enhancement Model



# The Back Channel Model



# One Legacy Implementation Approach



# Comparison of 3 Models

Feature	Appln prot enhancement	Encapsulat'g security layer	Back channel
Requires modification to the application layer protocol	Yes, since extra policy information needs to be carried	No	No
Requires trust in the PEP	Yes, to parse the message into security blocks and to carry the sticky policy between systems	Yes, to parse the message into security blocks	Yes, to parse the message into security blocks
Requires changes to the PEP interface	Yes somewhat, since policies are being transferred across the interface	Yes significantly, since the AIPEP will be transporting the application messages	No
Has control over distributed system security	Yes somewhat, but has to rely on application more	Yes complete control	Yes, significant control
Implementation effort for application	Easier	Most complex	Easiest (no changes needed)
Implementation effort for AIPEP	Easiest	Most complex	Medium difficulty
Supports integrated trust ne	No	Yes	No

# Trust Negotiation

- How does the sending system know if the receiving system is willing or able to enforce the sticky policies
- Trust Negotiation is one mechanism in which this can be determined
- Trust negotiation is the term used to refer to the gradual release of credentials by either party in order to mutually establish and build up trust between them
- May be carried out by a trust negotiation service on behalf of the application

# Difference Between Sticky Policies and DRM

- DRM assumes the recipient is a bad guy who wants to rip off the content owner
  - Recipient is restricted in what they can do with the data
- Sticky policies (esp. privacy ones) assume recipient wants to enforce the sticky policies (e.g. user's consent) but lacks mechanisms to automate this
  - Applications that want to allow back door access to data and/or remove sticky policies from data are not constrained in this respect

Any Questions?